

The Ruby Programming Language

Draft 2007-11-01

**David Flanagan
Yukihiro Matsumoto**

DRAFT

The Ruby Programming Language: Draft 2007-11-01

by David Flanagan and Yukihiro Matsumoto

Copyright © 2007 O'Reilly Media Inc. All Rights Reserved.

This is an unpublished draft for review purposes only. Please do not make or distribute copies.

DRAFT

Table of Contents

About This Book	xvii
1. Introduction	1
1.1. A Tour of Ruby	1
1.1.1. Ruby is Object-Oriented	1
1.1.2. Blocks and Iterators	2
1.1.3. Expressions and Operators in Ruby	4
1.1.4. Methods	5
1.1.5. Assignment	5
1.1.6. Punctuation Suffixes and Prefixes	6
1.1.7. Regexp and Range	6
1.1.8. Classes and Modules	7
1.1.9. Ruby Surprises	10
1.2. Try Ruby	10
1.2.1. The Ruby Interpreter	10
1.2.2. Displaying Output	12
1.2.3. Interactive Ruby with irb	12
1.2.4. Viewing Ruby Documentation with ri	13
1.2.5. More Ruby Tutorials	13
1.2.6. Ruby Resources	13
1.3. About This Book	14
1.4. A Sudoku Solver in Ruby	15
2. The Structure and Execution of Ruby Programs	23
2.1. Lexical Structure	23
2.1.1. Comments	23
2.1.2. Literals	25
2.1.3. Punctuation	26
2.1.4. Identifiers	26
2.1.5. Keywords	27
2.1.6. White Space	28
2.1.7. Characters and Encodings	30
2.2. Syntactic Structure	32
2.2.1. Block Structure in Ruby	34
2.3. File Structure	35
2.4. Program Execution	35
3. Datatypes and Objects	37
3.1. Numbers	37
3.1.1. Integer Literals	38
3.1.2. Floating-Point Literals	38
3.1.3. Arithmetic in Ruby	39
3.1.4. Binary Floating-Point and Rounding Errors	40
3.2. Text	41
3.2.1. String Literals	42
3.2.2. Character Literals	49
3.2.3. String Operators	49

3.2.4. Accessing Characters and Substrings	51
3.2.5. Iterating Strings	53
3.3. Arrays	53
3.4. Hashes	56
3.4.1. Hash Literals	57
3.4.2. Hash Codes, Equality, and Mutable Keys	57
3.5. Ranges	58
3.5.1. Testing Membership in a Range	59
3.6. Symbols	60
3.7. True, False, and Nil	62
3.8. Objects	62
3.8.1. Object References	62
3.8.2. Object Lifetime	63
3.8.3. Object Identity	64
3.8.4. Object Class and Object Type	64
3.8.5. Object Equality	66
3.8.6. Object Order	68
3.8.7. Object Conversion	69
3.8.8. Copying Objects	72
3.8.9. Marshaling Objects	73
3.8.10. Freezing Objects	73
3.8.11. Tainting Objects	74
4. Expressions and Operators	75
4.1. Literals and Keyword Literals	75
4.2. Variable References	76
4.2.1. Uninitialized Variables	76
4.3. Constant References	77
4.4. Method Invocations	79
4.5. Assignments	81
4.5.1. Assigning to Variables	83
4.5.2. Assigning to Constants	83
4.5.3. Assigning to Attributes and Array Elements	84
4.5.4. Abbreviated Assignment	85
4.5.5. Parallel Assignment	87
4.6. Operators	90
4.6.1. Unary + and -	93
4.6.2. Exponentiation: **	93
4.6.3. Arithmetic: +, -, *, /, and %	93
4.6.4. Shift and Append: << and >>	94
4.6.5. Complement, Union, Intersection: ~, &, , and ^	94
4.6.6. Comparison: <, <=, >, >=, and <=>	95
4.6.7. Equality: ==, !=, =~, !~, ===	96
4.6.8. Boolean Operators: &&, , !, and, or, not	97
4.6.9. Ranges and Flip-Flops: .. and	99
4.6.10. Conditional: ?	101
4.6.11. Assignment Operators	103

4.6.12. defined?	103
4.6.13. Statement modifiers	104
4.6.14. Non-Operators	105
5. Statements and Control Structures	107
5.1. Conditionals	107
5.1.1. if	108
5.1.2. if as a Modifier	110
5.1.3. unless	112
5.1.4. case	113
5.1.5. The ?: Operator	116
5.2. Loops	116
5.2.1. while and until	117
5.2.2. while and until as modifiers	117
5.2.3. for/in	119
5.3. Iterators and Enumerable Objects	120
5.3.1. Numeric Iterators	121
5.3.2. Enumerable Objects	121
5.3.3. Writing Custom Iterators	123
5.3.4. Enumerators	125
5.3.5. External Iterators	126
5.3.6. Iteration and Concurrent Modification	129
5.4. Blocks	130
5.4.1. Block Syntax	130
5.4.2. The Value of a Block	131
5.4.3. Blocks and Variable Scope	132
5.4.4. Passing Arguments to a Block	133
5.5. Altering Control Flow	135
5.5.1. return	136
5.5.2. break	138
5.5.3. next	139
5.5.4. redo	141
5.5.5. retry	142
5.5.6. throw and catch	143
5.6. Exceptions and Exception Handling	144
5.6.1. Exception Classes and Exception Objects	145
5.6.2. Raising Exceptions with raise	147
5.6.3. Handling Exceptions with rescue	148
5.6.4. The else Clause	152
5.6.5. The ensure Clause	153
5.6.6. rescue with Method, Class, and Module Definitions	154
5.6.7. rescue as a statement modifier	155
5.7. BEGIN and END	155
5.8. Threads, Fibers and Continuations	157
5.8.1. Threads for Concurrency	157
5.8.2. Fibers for Coroutines	158
5.8.3. Continuations	162

6. Methods, Procs, Lambdas, and Closures	165
6.1. Defining Simple Methods	166
6.1.1. Method Return Value	166
6.1.2. Methods and Exception Handling	167
6.1.3. Invoking a Method on an Object	168
6.1.4. Defining Singleton Methods	168
6.1.5. undefining Methods	169
6.2. Method Names	169
6.2.1. Operator Methods	170
6.2.2. Method Aliases	171
6.3. Methods and Parentheses	172
6.3.1. Optional Parentheses	172
6.3.2. Required Parentheses	173
6.4. Method Arguments	174
6.4.1. Parameter Defaults	175
6.4.2. Variable-Length Argument Lists and Arrays	176
6.4.3. Mapping Arguments to Parameters	177
6.4.4. Hashes for Named Arguments	178
6.4.5. Block Arguments	179
6.5. Procs and Lambdas	182
6.5.1. Creating Procs	182
6.5.2. Invoking Procs and Lambdas	185
6.5.3. The Arity of a Proc	186
6.5.4. Proc Equality	187
6.5.5. How Lambdas Differ from Procs	187
6.6. Closures	190
6.6.1. Closures and Shared Variables	191
6.6.2. Closures and Bindings	192
6.7. Method Objects	193
6.7.1. Unbound Method Objects	194
6.8. Functional Programming	195
6.8.1. Applying a Function to an Enumerable	195
6.8.2. Function Composition	197
6.8.3. Function Currying	198
6.8.4. Function Memoization	199
6.8.5. Symbols, Methods, and Procs	199
7. Classes and Modules	203
7.1. Defining a Simple Class	204
7.1.1. Creating the Class	204
7.1.2. Instantiating a Point	204
7.1.3. Initializing a Point	205
7.1.4. Defining a to_s Method	206
7.1.5. Accessors and Attributes	207
7.1.6. Defining Operators	209
7.1.7. Array and Hash Access with []	211
7.1.8. Enumerating Coordinates	211

7.1.9. Point Equality	212
7.1.10. Ordering Points	215
7.1.11. A Mutable Point	215
7.1.12. Quick and Easy Mutable Classes	216
7.1.13. A Class Method	218
7.1.14. Constants	219
7.1.15. Class Variables	220
7.1.16. Class Instance Variables	221
7.2. Method Visibility: Public, Protected, Private	222
7.3. Subclassing and Inheritance	224
7.3.1. Inheriting Methods	226
7.3.2. Overriding Methods	226
7.3.3. Augmenting Behavior by Chaining	228
7.3.4. Inheritance of Class Methods	229
7.3.5. Inheritance and Instance Variables	229
7.3.6. Inheritance of Constants	230
7.3.7. Inheritance of Class Variables	231
7.4. Object Creation and Initialization	232
7.4.1. new, allocate and initialize	232
7.4.2. Factory Methods	233
7.4.3. dup, clone and initialize_copy	233
7.4.4. marshal_dump and marshal_load	235
7.4.5. The Singleton Pattern	236
7.5. Modules	238
7.5.1. Modules as Namespaces	238
7.5.2. Modules as Mixins	240
7.5.3. Includeable Namespace Modules	242
7.6. Loading and Requiring Modules	243
7.6.1. The Load Path	244
7.6.2. Executing Loaded Code	245
7.6.3. Autoloading Modules	246
7.7. Singleton Methods and The Eigenclass	246
7.8. Method Lookup	248
7.8.1. Class Method Lookup	249
7.9. Constant Lookup	251
8. Reflection and Metaprogramming	255
8.1. Types, Classes and Modules	255
8.1.1. Ancestry and Modules	256
8.1.2. Defining Classes and Modules	257
8.2. Evaluating Strings and Blocks	258
8.2.1. Bindings and eval	258
8.2.2. instance_eval and class_eval	259
8.2.3. instance_exec and class_exec	260
8.3. Variables and Constants	260
8.3.1. Querying, Setting, and Testing Variables	260
8.4. Methods	262

8.4.1. Listing and Testing For Methods	262
8.4.2. Obtaining Method Objects	263
8.4.3. Invoking Methods	263
8.4.4. Defining, undefining and aliasing Methods	264
8.4.5. Handling Undefined Methods	265
8.4.6. Setting Method Visibility	266
8.5. Hooks	267
8.6. Tracing	269
8.7. ObjectSpace and GC	270
8.8. Custom Control Structures	271
8.8.1. Delaying and Repeating Execution: after and every	271
8.8.2. Thread-safety with synchronized Blocks	272
8.9. Missing Methods and Missing Constants	274
8.9.1. Unicode Codepoint Constants with const_missing	274
8.9.2. Tracing Method Invocations with method_missing	275
8.9.3. Synchronized Objects by Delegation	277
8.10. Dynamically Creating Methods	277
8.10.1. Defining Methods with class_eval	278
8.10.2. Defining Methods with define_method	279
8.11. Alias Chaining	280
8.11.1. Tracing Files Loaded and Classes Defined	281
8.11.2. Chaining Methods for Thread Safety	282
8.11.3. Chaining Methods for Tracing	285
8.12. Domain-Specific Languages	286
8.12.1. Simple XML Output with method_missing	287
8.12.2. Validated XML Output with Method Generation	289
9. The Ruby Platform	293
9.1. Strings	293
9.1.1. Formatting Text	297
9.1.2. Packing and Unpacking Binary Strings	298
9.1.3. Multi-byte Characters in Ruby 1.8	299
9.2. Regular Expressions	300
9.2.1. Regexp Literals	300
9.2.2. Regexp Factory Methods	301
9.2.3. Regular Expression Syntax	302
9.2.4. Pattern Matching with Regular Expressions	305
9.3. Numbers and Math	310
9.3.1. Numeric Methods	310
9.3.2. The Math Module	311
9.3.3. Decimal Arithmetic	312
9.3.4. Complex Numbers	313
9.3.5. Rational Numbers	313
9.3.6. Vectors and Matrices	313
9.3.7. Random Numbers	314
9.4. Dates and Times	314
9.5. Collections	317

9.5.1. Enumerable Objects	317
9.5.2. Arrays	324
9.5.3. Hashes	331
9.5.4. Sets	336
9.6. Input/Output and Files	341
9.6.1. Obtaining IO Objects	341
9.6.2. Reading from a Stream	342
9.6.3. Writing to a Stream	345
9.6.4. Random Access Methods	346
9.6.5. Closing, Flushing and Testing Streams	347
9.6.6. Reading and Writing Files	348
9.6.7. Working with Files and Directories	349
9.7. Networking	355
9.7.1. A Very Simple Client	355
9.7.2. A Very Simple Server	356
9.7.3. Datagrams	356
9.7.4. A More Complex Client	358
9.7.5. A Multiplexing Server	359
9.7.6. Fetching Web Pages	360
9.8. Threads and Concurrency	361
9.8.1. Thread Lifecycle	363
9.8.2. Threads and Variables	364
9.8.3. Thread Scheduling	366
9.8.4. Thread States	367
9.8.5. Listing Threads and Thread Groups	368
9.8.6. Threading Examples	369
9.8.7. Thread Exclusion and Deadlock	371
9.8.8. Queue and SizedQueue	374
9.8.9. Condition Variables and Queues	375
10. The Ruby Environment	377
10.1. Invoking the Ruby Interpreter	377
10.1.1. Common Options	378
10.1.2. Warnings and Information Options	379
10.1.3. Text Processing Options	379
10.1.4. Miscellaneous Options	380
10.2. The Toplevel Environment	381
10.2.1. Predefined Modules and Classes	381
10.2.2. Toplevel Constants	382
10.2.3. Global Variables	383
10.2.4. Pre-Defined Global Functions	387
10.2.5. User-Defined Global Functions	389
10.3. Practical Extraction and Reporting Shortcuts	389
10.3.1. Input Functions	390
10.3.2. Deprecated Extraction Functions	390
10.3.3. Reporting Functions	390
10.3.4. One Line Script Shortcuts	391

10.4. Calling the OS 391
 10.4.1. Invoking OS Commands 392
 10.4.2. Forking and Processes 392
 10.4.3. Trapping Signals 394
 10.4.4. Terminating Programs 395
10.5. Security 395
 10.5.1. Tainted Data 395
 10.5.2. Restricted Execution and Safe Levels 396

DRAFT

List of Figures

3.1. Numeric Class Hierarchy	37
5.1. An iterator yielding to its invoking method	120
5.2. The return statement in a block	137
5.3. The break statement in a block	138
5.4. The next statement in a block	140
5.5. The Ruby Exception Class Hierarchy	145

DRAFT

DRAFT

List of Tables

3.1. Backslash escapes in double-quoted strings	44
4.1. Abbreviated assignment pseudo-operators	85
4.2. Ruby Operators, by precedence (high to low), with arity (N), associativity (A), and define-ability (M)	92
4.3. Return values of the defined? operator	103
9.1. Regular expression modifier characters	300
9.2. Regular expression syntax	303
9.3. Special Global Regular Expression Variables	307

DRAFT

DRAFT

List of Examples

1.1. A Sudoku solver in Ruby	15
5.1. Parallel iteration with external iterators	128
7.1. Constant Name Resolution	252
8.1. The after and every methods	272
8.2. Simple synchronized blocks	273
8.3. Unicode codepoint constants with const_missing	274
8.4. Tracing method invocations with method_missing	275
8.5. Synchronizing methods with method_missing	277
8.6. Attribute methods with class_eval	278
8.7. Attribute methods with define_method	279
8.8. Tracing files loaded and classes defined	281
8.9. Alias chaining for thread safety	283
8.10. Chaining with singleton methods for tracing	285
8.11. A simple DSL for generating XML output	288
8.12. A DSL for validated XML output	290

DRAFT

DRAFT

About This Book

This book is an updated version of Matz's book *Ruby in a Nutshell* which has been expanded well beyond the *Nutshell* format. As its new title implies, this book covers the Ruby programming language and aspires to do so comprehensively while still being accessible to any experienced programmer who takes the time to read it carefully. This first edition of the book covers Ruby 1.8 and Ruby 1.9.

Ruby blurs the distinction between language and platform, and so the coverage of the language includes a detailed overview of the core Ruby API. But this book is not an API reference and does not attempt to document every class and every method of the core library. Also, this is not a book about Ruby frameworks (like Rails) nor a book about Ruby tools (like *rake* and *gem*). This book does not attempt to teach OO programming, or any kind of programming methodology. And although this book documents Ruby authoritatively, it is not intended as a specification for the language: language implementers will need more than this book to correctly implement Ruby.

This draft of the book is almost complete: the string encoding changes pending for 1.9 have not been documented yet, since they are still in flux. This draft has not yet been edited, and there may be some rough patches that the professionals at O'Reilly will smooth out. You can expect to see this book in stores in early 2008.

David Flanagan
<http://www.davidflanagan.com/>
November 1, 2007

DRAFT