Ruby (programming language)

This article is about the programming language. For other uses, see Ruby (disambiguation).
Not to be confused with Ruby on Rails.

Ruby is a dynamic, reflective, object-oriented, general-purpose programming language. It was designed and developed in the mid-1990s by Yukihiro “Matz” Matsumoto in Japan.

According to its creator, Ruby was influenced by Perl, Smalltalk, Eiffel, Ada, and Lisp.[11] It supports multiple programming paradigms, including functional, object-oriented, and imperative. It also has a dynamic type system and automatic memory management.

1 History

1.1 Early concept

Ruby was conceived on February 24, 1993. In a 1999 post to the *ruby-talk* mailing list, Ruby author Yukihiro Matsumoto describes some of his early ideas about the language:[12]

> I was talking with my colleague about the possibility of an object-oriented scripting language. I knew Perl (Perl4, not Perl5), but I didn't like it really, because it had the smell of a toy language (it still has). The object-oriented language seemed very promising. I knew Python then. But I didn't like it, because I didn't think it was a true object-oriented language — OO features appeared to be add-on to the language. As a language maniac and OO fan for 15 years, I really wanted a genuine object-oriented, easy-to-use scripting language. I looked for but couldn't find one. So I decided to make it.

Matsumoto describes the design of Ruby as being like a simple Lisp language at its core, with an object system like that of Smalltalk, blocks inspired by higher-order functions, and practical utility like that of Perl.[13]

1.2 The name “Ruby”

The name “Ruby” originated during an online chat session between Matsumoto and Keiju Ishitsuka on February 24, 1993, before any code had been written for the language.[14] Initially two names were proposed: "Coral" and "Ruby". Matsumoto chose the latter in a later e-mail to Ishitsuka.[15] Matsumoto later noted a factor in choosing the name “Ruby” – it was the birthstone of one of his colleagues.[16][17]

1.3 First publication

The first public release of Ruby 0.95 was announced on Japanese domestic newsgroups on December 21, 1995.[18][19] Subsequently, three more versions of Ruby were released in two days.[14] The release coincided with the launch of the Japanese-language *ruby-list* mailing list, which was the first mailing list for the new language.

Already present at this stage of development were many of the features familiar in later releases of Ruby, including object-oriented design, classes with inheritance, mixins, iterators, closures, exception handling and garbage collection.[20]

1.4 Early releases

Following the release of Ruby 0.95 in 1995, several stable versions of Ruby were released in the following years:

- Ruby 1.0: December 25, 1996[14]
- Ruby 1.2: December 1998
- Ruby 1.4: August 1999
- Ruby 1.6: September 2000

In 1997, the first article about Ruby was published on the Web. In the same year, Matsumoto was hired by netlab.jp to work on Ruby as a full-time developer.[14]

In 1998, the Ruby Application Archive was launched by Matsumoto, along with a simple English-language homepage for Ruby.[14]

In 1999, the first English language mailing list *ruby-talk* began, which signaled a growing interest in the language outside Japan.[21] In this same year, Matsumoto and Keiju Ishitsuka wrote the first book on Ruby, *The Object-oriented Scripting Language Ruby* (⍰⍰⍰⍰⍰⍰⍰⍰⍰⍰⍰⍰⍰⍰⍰ Ruby), which was published in Japan in October 1999. It would be followed in the early 2000s by around 20 books on Ruby published in Japanese.[14]

By 2000, Ruby was more popular than Python in Japan.[22] In September 2000, the first English language book *Programming Ruby* was printed, which was later freely released to the public, further widening the adoption of Ruby amongst English speakers. In early 2002, the English-language *ruby-talk* mailing list was receiving more messages than the Japanese-language *ruby-list*, demonstrating Ruby's increasing popularity in the English-speaking world.

1.5 Ruby 1.8

Ruby 1.8 was initially released in August 2003, was stable for a long time, and was retired June 2013.[23] Although deprecated, there is still code based on it. Ruby 1.8 is only partially compatible with Ruby 1.9.

Ruby 1.8 has been the subject of several industry standards. The language specifications for Ruby were developed by the Open Standards Promotion Center of the Information-Technology Promotion Agency (a Japanese government agency) for submission to the Japanese Industrial Standards Committee (JISC) and then to the International Organization for Standardization (ISO). It was accepted as a Japanese Industrial Standard (JIS X 3017) in 2011[24] and an international standard (ISO/IEC 30170) in 2012.[25]

Around 2005, interest in the Ruby language surged in tandem with Ruby on Rails, a web framework written in Ruby. Rails is frequently credited with increasing awareness of Ruby.[26]

1.6 Ruby 1.9

Ruby 1.9 was released in December 2007. Effective with Ruby 1.9.3, released October 31, 2011,[27] Ruby switched from being dual-licensed under the Ruby License and the GPL to being dual-licensed under the Ruby License and the two-clause BSD license.[28] Adoption of 1.9 was slowed by changes from 1.8 that required many popular third party gems to be rewritten.

Ruby 1.9 introduces many significant changes over the 1.8 series.[29] Examples:

- block local variables (variables that are local to the block in which they are declared)
- an additional lambda syntax: f = ->(a,b) { puts a + b }
- per-string character encodings are supported
- new socket API (IPv6 support)
- require_relative import security

Ruby 1.9 has been obsolete since February 23, 2015,[30] and it will no longer receive bug and security fixes. Users are advised to upgrade to a more recent version.

1.7 Ruby 2.0

Ruby 2.0 added several new features, including:

- method keyword arguments,
- a new method, Module#prepend, for extending a class,
- a new literal for creating an array of symbols,
- new API for the lazy evaluation of Enumerables, and
- a new convention of using #to_h to convert objects to Hashes.[31]

Ruby 2.0 is intended to be fully backward compatible with Ruby 1.9.3. As of the official 2.0.0 release on February 24, 2013, there were only five known (minor) incompatibilities.[32]

It has been obsolete since February 22, 2016 and it will no longer receive bug and security fixes. Users are advised to upgrade to a more recent version.

1.8 Ruby 2.1

Ruby 2.1.0 was released on Christmas Day in 2013.[33] The release includes speed-ups, bugfixes, and library updates.

Starting with 2.1.0, Ruby's versioning policy is more like semantic versioning.[34] Although similar, Ruby's versioning policy is not compatible with semantic versioning:

Semantic versioning also provides additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format, not available at Ruby.

1.9 Ruby 2.2

Ruby 2.2.0 was released on Christmas Day in 2014.[35] The release includes speed-ups, bugfixes, and library updates and removes some deprecated APIs. Most notably, Ruby 2.2.0 introduces changes to memory handling – an incremental garbage collector, support for garbage collection of symbols and the option to compile directly against jemalloc. It also contains experimental support for using vfork(2) with system() and spawn(), and added support for the Unicode 7.0 specification.

Features that were made obsolete or removed include callcc, the DL library, Digest::HMAC, lib/rational.rb, lib/complex.rb, GServer, Logger::Application as well as various C API functions.[36]

PowerPC64 performance Since version 2.2.1,[37] Ruby MRI performance on PowerPC64 was improved.[38][39][40]

1.10 Ruby 2.3

Ruby 2.3.0 was released on Christmas Day in 2015. A few notable changes include:

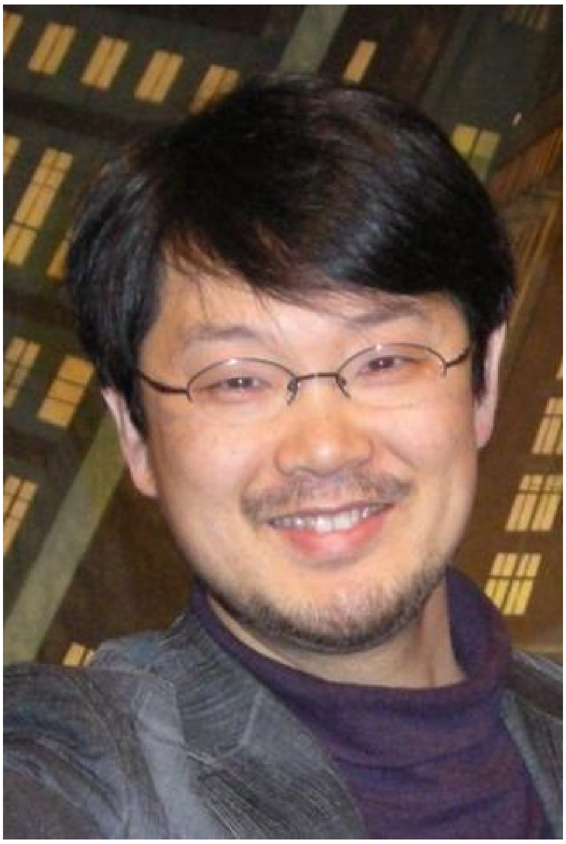
- The ability to mark all strings literals as frozen by default with consequently large performance increase in string operations.[41]
- Hash comparison to allow direct checking of key/value pairs instead of just keys.
- A new safe navigation operator &. that can ease nil handling (e.g. instead of if obj && obj.foo && obj.foo.bar, we can use if obj&.foo&.bar).
- The *did_you_mean* gem is now bundled by default and required on startup to automatically suggest similar name matches on a *NameError* or *NoMethodError*.
- *Hash#dig* and *Array#dig* to easily extract deeply nested values (e.g. given profile = { social: { wikipedia: { name: 'Foo Baz' } } }<nowiki />, the value *Foo Baz* can now be retrieved by profile.dig(:social, :wikipedia, :name)).
- .grep_v(regexp) which will match all negative examples of a given regular expression in addition to other new features.

The 2.3 branch also includes many performance improvements, updates, and bugfixes including changes to Proc#call, Socket and IO use of exception keywords, Thread#name handling, default passive Net::FTP connections, and Rake being removed from stdlib.[42]

2 Table of versions

3 Philosophy

Matsumoto has said that Ruby is designed for programmer productivity and fun, following the principles of good user interface design.[63] At a Google Tech Talk in 2008 Matsumoto further stated, “I hope to see Ruby help every programmer in the world to be productive, and to enjoy programming, and to be happy. That is the primary purpose of Ruby language.”[64] He stresses that systems design needs to emphasize human, rather than computer, needs:[65]



Yukihiro Matsumoto, the creator of Ruby

> Often people, especially computer engineers, focus on the machines. They think, “By doing this, the machine will run fast. By doing this, the machine will run more effectively. By doing this, the machine will something something something.” They are focusing on machines. But in fact we need to focus on humans, on how humans care about doing programming or operating the application of the machines. We are the masters. They are the slaves.

Ruby is said to follow the principle of least astonishment (POLA), meaning that the language should behave in such a way as to minimize confusion for experienced users. Matsumoto has said his primary design goal was to make a language that he himself enjoyed using, by minimizing programmer work and possible confusion. He has said that he had not applied the principle of least astonishment to the design of Ruby,[65] but nevertheless the phrase has come to be closely associated with the Ruby programming language. The phrase has itself been a source of surprise, as novice users may take it to mean that Ruby’s behaviors try to closely match behaviors familiar from other languages. In a May 2005 discussion on the newsgroup comp.lang.ruby, Matsumoto attempted to distance Ruby from POLA, explaining that because any design choice will be surprising to someone, he uses a personal standard in evaluating surprise. If that personal standard remains consistent, there would be few surprises for those familiar with the standard.[66]

Matsumoto defined it this way in an interview:[65]

> Everyone has an individual background. Someone may come from Python, someone else may come from Perl, and they may be surprised by different aspects of the language. Then they come up to me and say, 'I was surprised by this feature of the language, so Ruby violates the principle of least surprise.' Wait. Wait. The principle of least surprise is not for you only. The principle of least surprise means principle of least *my* surprise. And it means the principle of least surprise after you learn Ruby very well. For example, I was a C++ programmer before I started designing Ruby. I programmed in C++ exclusively for two or three years. And after two years of C++ programming, it still surprises me.

4 Features

- Thoroughly object-oriented with inheritance, mixins and metaclasses[67]
- Dynamic typing and duck typing
- Everything is an expression (even statements) and everything is executed imperatively (even declarations)
- Succinct and flexible syntax[68] that minimizes syntactic noise and serves as a foundation for domain-specific languages[69]
- Dynamic reflection and alteration of objects to facilitate metaprogramming[70]
- Lexical closures, iterators and generators, with a unique block syntax[71]
- Literal notation for arrays, hashes, regular expressions and symbols
- Embedding code in strings (interpolation)
- Default arguments
- Four levels of variable scope (global, class, instance, and local) denoted by sigils or the lack thereof
- Garbage collection
- First-class continuations
- Strict boolean coercion rules (everything is *true* except false and nil)
- Exception handling
- Operator overloading
- Built-in support for rational numbers, complex numbers and arbitrary-precision arithmetic
- Custom dispatch behavior (through method_missing and const_missing)
- Native threads and cooperative fibers (fibers are a 1.9/YARV feature)
- Initial support for Unicode and multiple character encodings (no ICU support)[72]
- Native plug-in API in C
- Interactive Ruby Shell (a REPL)
- Centralized package management through RubyGems
- Implemented on all major platforms
- Large standard library, including modules for YAML, JSON, XML, CGI, OpenSSL, HTTP, FTP, RSS, curses, zlib, and Tk[73]

5 Semantics

Ruby is object-oriented: every value is an object, including classes and instances of types that many other languages designate as primitives (such as integers, booleans, and "null"). Variables always hold references to objects. Every function is a method and methods are always called on an object. Methods defined at the top level scope become methods of the Object class. Since this class is an ancestor of every other class, such methods can be called on any object. They are also visible in all scopes, effectively serving as “global” procedures. Ruby supports inheritance with dynamic dispatch, mixins and singleton methods (belonging to, and defined for, a single instance rather than being defined on the class). Though Ruby does not support multiple inheritance, classes can import modules as mixins.

Ruby has been described as a multi-paradigm programming language: it allows procedural programming (defining functions/variables outside classes makes them part of the root, 'self' Object), with object orientation (everything is an object) or functional programming (it has anonymous functions, closures, and continuations; statements all have values, and functions return the last evaluation). It has support for introspection, reflection and metaprogramming, as well as support for interpreter-based[74] threads. Ruby features dynamic typing, and supports parametric polymorphism.

According to the Ruby FAQ, the syntax is similar to Perl and the semantics are similar to Smalltalk but it differs greatly from Python.[75]

6 Syntax

The syntax of Ruby is broadly similar to that of Perl and Python. Class and method definitions are signaled by keywords, whereas code blocks can be both defined by keywords or braces. In contrast to Perl, variables are not obligatorily prefixed with a sigil. When used, the sigil changes the semantics of scope of the variable. For practical purposes there is no distinction between expressions and statements.[76] Line breaks are significant and taken as the end of a statement; a semicolon may be equivalently used. Unlike Python, indentation is not significant.

One of the differences of Ruby compared to Python and Perl is that Ruby keeps all of its instance variables completely private to the class and only exposes them through accessor methods (attr_writer, attr_reader, etc.). Unlike the “getter” and “setter” methods of other languages like C++ or Java, accessor methods in Ruby can be created with a single line of code via metaprogramming; however, accessor methods can also be created in the traditional fashion of C++ and Java. As invocation of these methods does not require the use of parentheses, it is trivial to change an instance variable into a full function, without modifying a single line of calling code or having to do any refactoring achieving similar functionality to C# and VB.NET property members.

Python’s property descriptors are similar, but come with a tradeoff in the development process. If one begins in Python by using a publicly exposed instance variable, and later changes the implementation to use a private instance variable exposed through a property descriptor, code internal to the class may need to be adjusted to use the private variable rather than the public property. Ruby’s design forces all instance variables to be private, but also provides a simple way to declare set and get methods. This is in keeping with the idea that in Ruby, one never directly accesses the internal members of a class from outside the class; rather, one passes a message to the class and receives a response.

See the Examples section below for samples of code demonstrating Ruby syntax.

7 Differences from other languages

Some features that differ notably from languages such as C or Perl:

- The language syntax is sensitive to the capitalization of identifiers, in all cases treating capitalized variables as constants. Class and module names are constants and refer to objects derived from Class and Module.
- The sigils $ and @ do not indicate variable data type as in Perl, but rather function as scope resolution operators.
- Floating point literals must have digits on both sides of the decimal point: neither .5 nor 2. are valid floating point literals, but 0.5 and 2.0 are.

  (In Ruby, integer literals are objects that can have methods apply to them, so requiring a digit after a decimal point helps to clarify whether 1.e5 should be parsed analogously to 1.to_f or as the exponential-format floating literal 1.0e5. The reason for requiring a digit before the decimal point is less clear; it might relate either to method invocation again, or perhaps to the .. and ... operators, for example in the fragment 0.1...3.)

- Boolean non-boolean datatypes are permitted in boolean contexts (unlike in e.g. Smalltalk and Java), but their mapping to boolean values differs markedly from some other languages: 0 and “empty” (e.g. empty list, string or associative array) all evaluate to *true*, thus changing the meaning of some common idioms in related or similar languages such as Lisp, Perl and Python.

  A consequence of this rule is that Ruby methods by convention — for example, regular-expression searches — return numbers, strings, lists, or other non-*false* values on success, but nil on failure.

- Versions prior to 1.9 use plain integers to represent single characters, much like C. This may cause surprises when slicing strings: “abc"[0] yields 97 (the ASCII code of the first character in the string); to obtain “a” use “abc"[0,1] (a substring of length 1) or “abc"[0].chr.
- The notation statement until expression does not run the statement if the expression is already *true*. (The behavior is like Perl, but unlike other languages’ equivalent statements, e.g. do { statement } while (!(expression)); in C/C++/...). This is because statement until expression is actually syntactic sugar over until expression; statement; end, the equivalent of which in C/C++ is while (!(expression)) { statement; }, just as statement if expression is equivalent to if (expression) { statement; }. However, the notation begin statement end until expression in Ruby will in fact run the statement once even if the expression is already *true*, acting similarly to the do-while of other languages. (Matsumoto has expressed a desire to remove the special behavior of begin statement end until expression,[77] but it still exists as of Ruby 2.0.)
- Because constants are references to objects, changing what a constant refers to generates a warning, but modifying the object itself does not. For example, Greeting << " world!" if Greeting == “Hello”

does not generate an error or warning. This is similar to final variables in Java or a const pointer to a non-const object in C++.

- Ruby provides the functionality to freeze an object.
- The usual conjunctive and disjunctive operators for conditional expressions have the same precedence, so and does not bind tighter than or in Ruby, a behaviour similar to languages such as APL, Ada, VHDL, Mathematica, zkl and others. However, Ruby also has C-like operators || and && that work as in C-like languages.

A list of so-called gotchas may be found in Hal Fulton's book *The Ruby Way*, 2nd ed (ISBN 0-672-32884-4), Section 1.5. A similar list in the 1st edition pertained to an older version of Ruby (version 1.6), some problems of which have been fixed in the meantime. For example, retry now works with while, until, and for, as well as with iterators.

8 Interaction

See also: Interactive Ruby Shell

The Ruby official distribution also includes irb, an interactive command-line interpreter that can be used to test code quickly. The following code fragment represents a sample session using irb:

```
$ irb irb(main):001:0> puts 'Hello, World' Hello, World
=> nil irb(main):002:0> 1+2 => 3
```

9 Examples

The following examples can be run in a Ruby shell such as Interactive Ruby Shell, or saved in a file and run from the command line by typing ruby <*filename*>.

Classic Hello world example:

```
puts 'Hello World!'
```

Some basic Ruby code:

```
# Everything, including a literal, is an object, so this
works: −199.abs # => 199 'ice is nice'.length # =>
11 'ruby is cool.'.index('u') # => 1 "Nice Day Isn't
It?".downcase.split(").uniq.sort.join # => " '?acdeinsty"
```

Input:

```
print 'Please type name >' name = gets.chomp puts
"Hello #{name}."
```

Conversions:

```
puts 'Give me a number' number = gets.chomp puts
number.to_i output_number = number.to_i + 1 puts
output_number.to_s + ' is a bigger number.'
```

9.1 Strings

There are a variety of ways to define strings in Ruby.

The following assignments are equivalent:

```
a = "\nThis is a double-quoted string\n" a = %Q{\nThis
is a double-quoted string\n} a = %{\nThis is a double-
quoted string\n} a = %/\nThis is a double-quoted
string\n/ a = <<-BLOCK This is a double-quoted string
BLOCK
```

Strings support variable interpolation:

```
var = 3.14159 "pi is #{var}" => "pi is 3.14159"
```

The following assignments are equivalent and produce raw strings:

```
a = 'This is a single-quoted string' a = %q{This is a
single-quoted string}
```

9.2 Collections

Constructing and using an array:

```
a = [1, 'hi', 3.14, 1, 2, [4, 5]] a[2] # => 3.14 a.[](2)
# => 3.14 a.reverse # => [[4, 5], 2, 1, 3.14, 'hi', 1]
a.flatten.uniq # => [1, 'hi', 3.14, 2, 4, 5]
```

Constructing and using an associative array (in Ruby, called a *hash*):

```
hash = Hash.new # equivalent to hash = {} hash = {
:water => 'wet', :fire => 'hot' } # makes the previous
line redundant as we are now # assigning hash to a
new, separate hash object puts hash[:fire] # prints
"hot" hash.each_pair do |key, value| # or: hash.each
do |key, value| puts "#{key} is #{value}" end # returns
{:water=>"wet", :fire=>"hot"} and prints: # water is wet
# fire is hot hash.delete :water # deletes the pair :water
=> 'wet' and returns "wet" hash.delete_if {|key,value|
value == 'hot'} # deletes the pair :fire => 'hot' and returns
{}
```

9.3 Control structures

If statement:

```
# Generate a random number and print whether it's even
```

or odd. if rand(100) % 2 == 0 puts “It’s even” else puts “It’s odd” end

9.4 Blocks and iterators

The two syntaxes for creating a code block:

{ puts 'Hello, World!' } # note the braces # or: do puts 'Hello, World!' end

A code block can be passed to a method as an optional block argument. Many built-in methods have such arguments:

File.open('file.txt', 'w') do |file| # 'w' denotes “write mode” file.puts 'Wrote some text.' end # file is automatically closed here File.readlines('file.txt').each do |line| puts line end # => Wrote some text.

Parameter-passing a block to be a closure:

# In an object instance variable (denoted with '@'), remember a block. def remember(&a_block) @block = a_block end # Invoke the preceding method, giving it a block that takes a name. remember {|name| puts “Hello, #{name}!"} # Call the closure (note that this happens not to close over any free variables): @block.call('Jon') # => “Hello, Jon!"

Creating an anonymous function:

proc {|arg| puts arg} Proc.new {|arg| puts arg} lambda {|arg| puts arg} ->(arg) {puts arg} # introduced in Ruby 1.9

Returning closures from a method:

def create_set_and_get(initial_value=0) # note the default value of 0 closure_value = initial_value [Proc.new {|x| closure_value = x}, Proc.new { closure_value }] end setter, getter = create_set_and_get # returns two values setter.call(21) getter.call # => 21 # Parameter variables can also be used as a binding for the closure, # so the preceding can be rewritten as: def create_set_and_get(closure_value=0) [proc {|x| closure_value = x } , proc { closure_value }] end

Yielding the flow of program control to a block that was provided at calling time:

def use_hello yield “hello” end # Invoke the preceding method, passing it a block. use_hello {|string| puts string} # => 'hello'

Iterating over enumerations and arrays using blocks:

array = [1, 'hi', 3.14] array.each {|item| puts item } # prints: # 1 # 'hi' # 3.14 array.each_index {|index| puts "#{index}: #{array[index]}" } # prints: # 0: 1 # 1: 'hi' # 2: 3.14 # The following uses a (a..b) Range (3..6).each {|num| puts num } # prints: # 3 # 4 # 5 # 6 # The following uses a (a...b) Range (3...6).each {|num| puts num } # prints: # 3 # 4 # 5

A method such as inject can accept both a parameter and a block. The inject method iterates over each member of a list, performing some function on it while retaining an aggregate. This is analogous to the foldl function in functional programming languages. For example:

[1,3,5].inject(10) {|sum, element| sum + element} # => 19

On the first pass, the block receives 10 (the argument to inject) as sum, and 1 (the first element of the array) as element. This returns 11, which then becomes sum on the next pass. It is added to 3 to get 14, which is then added to 5 on the third pass, to finally return 19.

Using an enumeration and a block to square the numbers 1 to 10 (using a *range*):

(1..10).collect {|x| x*x} # => [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

Or invoke a method on each item (map is a synonym for collect):

(1..5).map(&:to_f) # => [1.0, 2.0, 3.0, 4.0, 5.0]

9.5 Classes

The following code defines a class named Person. In addition to initialize, the usual constructor to create new objects, it has two methods: one to override the <=> comparison operator (so Array#sort can sort by age) and the other to override the to_s method (so Kernel#puts can format its output). Here, attr_reader is an example of metaprogramming in Ruby: attr_accessor defines getter and setter methods of instance variables, but attr_reader only getter methods. The last evaluated statement in a method is its return value, allowing the omission of an explicit return statement.

class Person attr_reader :name, :age def initialize(name, age) @name, @age = name, age end def <=>(person) # the comparison operator for sorting @age <=> person.age end def to_s "#{@name} (#{@age})" end end group = [Person.new(“Bob”, 33), Person.new(“Chris”, 16), Person.new(“Ash”, 23)] puts group.sort.reverse

The preceding code prints three names in reverse age order:

Bob (33) Ash (23) Chris (16)

Person is a constant and is a reference to a Class object.

9.5.1 Open classes

In Ruby, classes are never closed: methods can always be added to an existing class. This applies to *all* classes, including the standard, built-in classes. All that is needed to do is open up a class definition for an existing class, and the new contents specified will be added to the existing contents. A simple example of adding a new method to the standard library's Time class:

# re-open Ruby's Time class class Time def yesterday self - 86400 end end today = Time.now # => 2013-09-03 16:09:37 +0300 yesterday = today.yesterday # => 2013-09-02 16:09:37 +0300

Adding methods to previously defined classes is often called monkey-patching. If performed recklessly, the practice can lead to both behavior collisions with subsequent unexpected results and code scalability problems.

9.6 Exceptions

An exception is raised with a raise call:

raise

An optional message can be added to the exception:

raise “This is a message”

Exceptions can also be specified by the programmer:

raise ArgumentError, “Illegal arguments!"

Alternatively, an exception instance can be passed to the raise method:

raise ArgumentError.new(“Illegal arguments!")

This last construct is useful when raising an instance of a custom exception class featuring a constructor that takes more than one argument:

class ParseError < Exception def initialize input, line, pos super “Could not parse '#{input}' at line #{line}, position #{pos}" end end raise ParseError.new(“Foo”, 3, 9)

Exceptions are handled by the rescue clause. Such a clause can catch exceptions that inherit from StandardError. Other flow control keywords that can be used when handling exceptions are else and ensure:

begin # do something rescue # handle exception else # do this if no exception was raised ensure # do this whether or not an exception was raised end

It is a common mistake to attempt to catch all exceptions with a simple rescue clause. To catch all exceptions one must write:

begin # do something rescue Exception # Exception handling code here. # Don't write only “rescue"; that only catches StandardError, a subclass of Exception. end

Or catch particular exceptions:

begin # do something rescue RuntimeError # handle only RuntimeError and its subclasses end

It is also possible to specify that the exception object be made available to the handler clause:

begin # do something rescue RuntimeError => e # handling, possibly involving e, such as “puts e.to_s” end

Alternatively, the most recent exception is stored in the magic global $!.

Several exceptions can also be caught:

begin # do something rescue RuntimeError, Timeout::Error => e # handling, possibly involving e end

9.7 Metaprogramming

Ruby code can programmatically modify, at runtime, aspects of its own structure that would be fixed in more rigid languages, such as class and method definitions. This sort of metaprogramming can be used to write more concise code and effectively extend the language.

For example, the following Ruby code generates new methods for the built-in String class, based on a list of colors. The methods wrap the contents of the string with an HTML tag styled with the respective color.

COLORS = { black: “000”, red: “f00”, green: “0f0”, yellow: “ff0”, blue: “00f”, magenta: “f0f”, cyan: “0ff”, white: “fff” } class String COLORS.each do |color,code| define_method “in_#{color}" do "<span style=\"color: ##{code}\">#{self}</span>" end end end

The generated methods could then be used like this:

“Hello, World!".in_blue => "<span style=\"color: #00f\">Hello, World!</span>"

To implement the equivalent in many other languages, the programmer would have to write each method (in_black, in_red, in_green, etc.) separately.

Some other possible uses for Ruby metaprogramming include:

- intercepting and modifying method calls
- implementing new inheritance models
- dynamically generating classes from parameters
- automatic object serialization
- interactive help and debugging

9.8 More examples

More sample Ruby code is available as algorithms in the following article:

- Exponentiating by squaring

10 Implementations

See also: Ruby MRI § Operating systems

10.1 Matz's Ruby Interpreter

The official Ruby interpreter often referred to as the Matz's Ruby Interpreter or MRI. This implementation is written in C and uses its own Ruby-specific virtual machine.

The standardized and retired Ruby 1.8 implementation was written in C, as a single-pass interpreted language.[23]

Starting with Ruby 1.9, and continuing with Ruby 2.x and above, the official Ruby interpreter has been YARV ("Yet Another Ruby VM"), and this implementation has superseded the slower virtual machine used in previous releases of MRI.

10.2 Alternate implementations

As of 2010, there are a number of alternative implementations of Ruby, including JRuby, Rubinius, MagLev, IronRuby, MacRuby (and its iOS counterpart, RubyMotion), mruby, HotRuby, Topaz and Opal. Each takes a different approach, with IronRuby, JRuby, MacRuby and Rubinius providing just-in-time compilation and MacRuby and mruby also providing ahead-of-time compilation.

Ruby has two major alternate implementations:

- JRuby, a Java implementation that runs on the Java virtual machine. JRuby currently targets Ruby 2.2,
- Rubinius, a C++ bytecode virtual machine that uses LLVM to compile to machine code at runtime. The bytecode compiler and most core classes are written in pure Ruby. Rubinius currently targets Ruby 2.1,

Other Ruby implementations include:

- MagLev, a Smalltalk implementation that runs on GemTalk Systems' GemStone/S VM
- mruby, an implementation designed to be embedded into C code, in a similar vein to Lua. It is currently being developed by Yukihiro Matsumoto and others
- Opal, a web-based interpreter that compiles Ruby to JavaScript
- RGSS, or Ruby Game Scripting System, a proprietary implementation that is used by the RPG Maker series of software for game design and modification of the RPG Maker engine
- A transpiler (partial) from Ruby to Julia, julializer is available. It can be used for a large speedup over e.g. Ruby or JRuby implementations (may only be useful for numerical code).[78]

Other now defunct Ruby implementations were:

- MacRuby, an OS X implementation on the Objective-C runtime
- IronRuby an implementation on the .NET Framework
- Cardinal, an implementation for the Parrot virtual machine
- Ruby Enterprise Edition, often shortened to *ree*, an implementation optimized to handle large-scale Ruby on Rails projects

The maturity of Ruby implementations tends to be measured by their ability to run the Ruby on Rails (Rails) framework, because it is complex to implement and uses many Ruby-specific features. The point when a particular implementation achieves this goal is called "the Rails singularity". The reference implementation (MRI), JRuby, and Rubinius[79] are all able to run Rails unmodified in a production environment.

10.3 Platform support

Matsumoto originally did Ruby development on the 4.3BSD-based Sony NEWS-OS 3.x, but later migrated his work to SunOS 4.x, and finally to Linux.[80][81]

By 1999, Ruby was known to work across many different operating systems, including NEWS-OS, SunOS, AIX, SVR4, Solaris, NEC UP-UX, NeXTSTEP, BSD, Linux, Mac OS, DOS, Windows, and BeOS.[82]

Modern Ruby versions and implementations are available on many operating systems, such as Linux, BSD, Solaris, AIX, OS X, Windows, Windows Phone,[83] Windows CE, Symbian OS, BeOS, and IBM i.

11 Repositories and libraries

RubyGems is Ruby's package manager. A Ruby package is called a "gem" and can easily be installed via the command line. Most gems are libraries, though a few exist that are applications, such as IDEs.[84] There are over 124,000 Ruby gems hosted on RubyGems.org.

Many new and existing Ruby libraries are hosted on GitHub, a service that offers version control repository hosting for Git.

The Ruby Application Archive, which hosted applications, documentation, and libraries for Ruby programming, was maintained until 2013, when its function was transferred to RubyGems.[85]

12 See also

- Comparison of programming languages
- Why's (poignant) Guide to Ruby — an online ruby textbook in graphic novel format
- Metasploit Project — the world's largest Ruby project, with over 700,000 lines of code
- XRuby

13 References

[1] Hayford, Emmanuel (2016-11-15). "Ruby 2.3.2 Released". *Ruby Programming Language*. Retrieved 2016-11-15.

[2] "[ruby] Contents of /trunk/COPYING". Retrieved 2 May 2015.

[3] "[ruby] Contents of /trunk/GPL". Retrieved 2 May 2015.

[4] "[ruby] Contents of /trunk/BSDL". Retrieved 2 May 2015.

[5] Cooper, Peter (2009). *Beginning Ruby: From Novice to Professional*. Beginning from Novice to Professional (2nd ed.). Berkeley: APress. p. 101. ISBN 1-4302-2363-4. To a lesser extent, Python, LISP, Eiffel, Ada, and C++ have also influenced Ruby.

[6] Bini, Ola (2007). *Practical JRuby on Rails Web 2.0 Projects: Bringing Ruby on Rails to Java*. Berkeley: APress. p. 3. ISBN 1-59059-881-4. It draws primarily on features from Perl, Smalltalk, Python, Lisp, Dylan, and CLU.

[7] Bini, Ola. "Ioke". *Ioke.org*. Retrieved 2011-07-21. inspired by Io, Smalltalk, Lisp and Ruby

[8] "Introduction — Julia Language 0.4.1 documentation". Retrieved 13 November 2015.

[9] Burks, Tim. "About Nu™". *Programming Nu™*. Neon Design Technology, Inc. Retrieved 2011-07-21.

[10] Lattner, Chris (2014-06-03). "Chris Lattner's Homepage". Chris Lattner. Retrieved 2014-06-03. The Swift language is the product of tireless effort from a team of language experts, documentation gurus, compiler optimization ninjas, and an incredibly important internal dogfooding group who provided feedback to help refine and battle-test ideas. Of course, it also greatly benefited from the experiences hard-won by many other languages in the field, drawing ideas from Objective-C, Rust, Haskell, Ruby, Python, C#, CLU, and far too many others to list.

[11] "About Ruby". Retrieved 2 March 2014.

[12] Shugo Maeda (17 December 2002). "The Ruby Language FAQ". Retrieved 2 March 2014.

[13] Yukihiro Matsumoto (13 February 2006), *ruby-talk: Re: Ruby's lisp features*, retrieved 2 March 2014

[14] http://blog.nicksieger.com/articles/2006/10/20/rubyconf-history-of-ruby History of Ruby

[15] http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/88819 "[FYI: historic] The decisive moment of the language name Ruby. (Re: [ANN] ruby 1.8.1)" — E-mail from Hiroshi Sugihara to ruby-talk

[16] "The Ruby Language FAQ – 1.3 Why the name 'Ruby'?". Ruby-Doc.org. Retrieved April 10, 2012.

[17] Yukihiro Matsumoto (June 11, 1999). "Re: the name of Ruby?". *Ruby-Talk* (Mailing list). Retrieved April 10, 2012.

[18] "More archeolinguistics: unearthing proto-Ruby". Retrieved 2 May 2015.

[19] "[ruby-talk:00382] Re: history of ruby". Retrieved 2 May 2015.

[20] "[ruby-list:124] TUTORIAL - ruby's features". Retrieved 2 May 2015.

[21] http://www.linuxdevcenter.com/pub/a/linux/2001/11/29/ruby.html An Interview with the Creator of Ruby

[22] Yukihiro Matsumoto (October 2000). "Programming Ruby: Forward". Retrieved 5 March 2014.

[23] "We retire Ruby 1.8.7". Retrieved 2 May 2015.

[24] "IPA ⍰⍰⍰⍰⍰⍰ ⍰⍰⍰⍰⍰⍰⍰⍰:⍰⍰⍰⍰⍰⍰⍰⍰⍰⍰⍰⍰Ruby⍰JIS⍰⍰(JIS X 3017)⍰⍰⍰⍰⍰⍰". Retrieved 2 May 2015.

[25] "IPA ⍰⍰⍰⍰⍰⍰ ⍰⍰⍰⍰⍰⍰⍰⍰:⍰⍰⍰⍰⍰⍰⍰⍰⍰⍰⍰⍰Ruby⍰⍰⍰⍰⍰⍰⍰⍰⍰⍰". Retrieved 2 May 2015.

[26] Web Development: Ruby on Rails. Devarticles.com (2007-03-22). Retrieved on 2013-07-17.

[27] "Ruby 1.9.3 p0 is released". ruby-lang.org. October 31, 2011. Retrieved February 20, 2013.

[28] "v1_9_3_0/NEWS". *Ruby Subversion source repository*. ruby-lang.org. September 17, 2011. Retrieved February 20, 2013.

[29] Ruby 1.9: What to Expect. Slideshow.rubyforge.org. Retrieved on 2013-07-17.

[30] "Support for Ruby 1.9.3 has ended". Retrieved 2 May 2015.

[31] Endoh, Yusuke. (2013-02-24) Ruby 2.0.0-p0 is released. Ruby-lang.org. Retrieved on 2013-07-17.

[32] Endoh, Yusuke. (2013-02-24) Ruby 2.0.0-p0 is released. Ruby-lang.org. Retrieved on 2013-07-17.

[33] "Ruby 2.1.0 is released". December 25, 2013. Retrieved December 26, 2013.

[34] "Semantic Versioning starting with Ruby 2.1.0". December 21, 2013. Retrieved December 27, 2013.

[35] "Ruby 2.2.0 Released". December 25, 2014. Retrieved January 4, 2015.

[36] "ruby/NEWS at v2_2_0 · ruby/ruby · GitHub". *GitHub*. Retrieved 2 May 2015.

[37] Gustavo Frederico Temple Pedrosa, Vitor de Lima, Leonardo Bianconi (2015). "Ruby 2.2.1 Released". Retrieved 12 July 2016.

[38] Gustavo Frederico Temple Pedrosa, Vitor de Lima, Leonardo Bianconi (2015). "v2.2.1 ChangeLog". Retrieved 12 July 2016.

[39] Gustavo Frederico Temple Pedrosa, Vitor de Lima, Leonardo Bianconi (2014). "Specifying non volatile registers for increase performance in ppc64". Retrieved 12 July 2016.

[40] Gustavo Frederico Temple Pedrosa, Vitor de Lima, Leonardo Bianconi (2014). "Specifying MACRO for increase performance in ppc64". Retrieved 12 July 2016.

[41] Ruby 2.3.0 changes and features – Frozen string literals

[42] "Ruby/NEWS at v.2_3_0 - ruby/ruby - Github". *GitHub*. Retrieved 25 December 2015.

[43] A Patch in Time: Securing Ruby

[44] ruby-1.8.0 released!

[45] Plans for 1.8.7

[46] EOL for Ruby 1.8.7 and 1.9.2

[47] Ruby 1.9.3-p551 Released

[48] Ruby 1.9.0 Released

[49] Support for Ruby version 1.9.3 will end on February 23, 2015

[50] Support for Ruby 1.9.3 has ended

[51] Ruby 2.0.0-p648 Released

[52] Ruby 2.0.0-p0 is released

[53] Ruby 2.1.10 Released

[54] Ruby 2.1.0 is released

[55] Support plans for Ruby 2.0.0 and Ruby 2.1

[56] Ruby 2.1.9 Released

[57] Release Engineering

[58] Ruby 2.2.5 Released

[59] Ruby 2.2.0 Released

[60] Ruby 2.3.1 Released

[61] Ruby 2.3.0 Released

[62] What's coming in Ruby 3 and Rails 5

[63] "The Ruby Programming Language". Retrieved 2 May 2015.

[64] Google Tech Talks – Ruby 1.9 on YouTube

[65] Bill Venners. "The Philosophy of Ruby". Retrieved 2 May 2015.

[66] Ruby Weekly News 23rd – 29th May 2005

[67] Bruce Stewart (29 November 2001). "An Interview with the Creator of Ruby - O'Reilly Media". Retrieved 2 May 2015.

[68] Bill Venners. "Dynamic Productivity with Ruby". Retrieved 2 May 2015.

[69] "Language Workbenches: The Killer-App for Domain Specific Languages?". *martinfowler.com*. Retrieved 2 May 2015.

[70] Ruby – Add class methods at runtime

[71] Bill Venners. "Blocks and Closures in Ruby". Retrieved 2 May 2015.

[72] "Feature #2034: Consider the ICU Library for Improving and Expanding Unicode Support - Ruby trunk - Ruby Issue Tracking System". Retrieved 2 May 2015.

[73] Britt, James. "Ruby 2.0.0 Standard Library Documentation". Retrieved 2013-12-09.

[74] Green threads

[75] "The Ruby Language FAQ: How Does Ruby Stack Up Against...?". Retrieved 2 May 2015.

[76] *In Ruby's syntax, statement is just a special case of an expression that cannot appear as an argument (e.g. multiple assignment).* http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/1120
statement [...] can not be part of expression unless grouped within parentheses. http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/2460

[77] Re: semenatics of if/unless/while statement modifiers. Blade.nagaokaut.ac.jp (2005-11-23). Retrieved on 2013-07-17.

[78] https://github.com/remore/virtual_module

[79] Peter Cooper (2010-05-18). "The Why, What, and How of Rubinius 1.0's Release".

[80] Maya Stodte (February 2000). “IBM developerWorks – Ruby: a new language”. Archived from the original on August 18, 2000. Retrieved 3 March 2014.

[81] Yukihiro Matsumoto (August 2002). “lang-ruby-general: Re: question about Ruby initial development”. Retrieved 3 March 2014.

[82] Yukihiro Matsumoto (5 January 1999). “ruby-talk: Re: hah, check these errors”. Retrieved 3 March 2014.

[83] “Iron Ruby on Windows Phone 7”.

[84] “The Ruby Toolbox”. Retrieved 2015-04-04.

[85] “We retire raa.ruby-lang.org”. 2013-08-08. Retrieved 2016-01-03.

14 Further reading

- Metz, Sandi (September 5, 2012), *Practical Object-Oriented Design in Ruby* (First ed.), Addison-Wesley, p. 272, ISBN 0-321-72133-0
- McAnally, Jeremy; Arkin, Assaf (March 28, 2009), *Ruby in Practice* (First ed.), Manning Publications, p. 360, ISBN 1-933988-47-9
- Thomas, Dave; Fowler, Chad; Hunt, Andy (April 28, 2009), *Programming Ruby 1.9: The Pragmatic Programmers’ Guide* (Third ed.), Pragmatic Bookshelf, p. 1000, ISBN 1-934356-08-5
- Black, David (June 4, 2009), *The Well-Grounded Rubyist* (First ed.), Manning Publications, p. 520, ISBN 1-933988-65-7
- Flanagan, David; Matsumoto, Yukihiro (January 25, 2008), *The Ruby Programming Language* (First ed.), O'Reilly Media, p. 446, ISBN 0-596-51617-7
- Baird, Kevin (June 8, 2007), *Ruby by Example: Concepts and Code* (First ed.), No Starch Press, p. 326, ISBN 1-59327-148-4
- Fitzgerald, Michael (May 14, 2007), *Learning Ruby* (First ed.), O'Reilly Media, p. 255, ISBN 0-596-52986-4
- Cooper, Peter (March 26, 2007), *Beginning Ruby: From Novice to Professional* (First ed.), Apress, p. 664, ISBN 1-59059-766-4
- Fulton, Hal (November 4, 2006), *The Ruby Way* (Second ed.), Addison-Wesley, p. 888, ISBN 0-596-52369-6
- Carlson, Lucas; Richardson, Leonard (July 19, 2006), *Ruby Cookbook* (First ed.), O'Reilly Media, p. 906, ISBN 0-596-52369-6

15 External links

- Official website
- Official Ruby documentation
- Ruby User Guide — by Yukihiro Matsumoto, the creator of Ruby
- A community-driven Ruby coding style guide
- Ruby From Other Languages
- Ruby Forum — gateway to the ruby-talk mailing list
- Try Ruby! — web-based Ruby REPL
- Ruby Draft Specification, September 2010
- Ruby at DMOZ

16 Text and image sources, contributors, and licenses

16.1 Text

- **Ruby (programming language)** *Source:* https://en.wikipedia.org/wiki/Ruby_(programming_language)?oldid=749658280 *Contributors:* AxelBoldt, Derek Ross, Lee Daniel Crocker, Brion VIBBER, Tarquin, Taw, Sjc, Drj, Andre Engels, Fnielsen, Youssefsan, Hajhouse, Matusz, Toby Bartels, Hannes Hirzel, Hirzel, Edward, K.lee, Robert Dober, Crenner, Wwwwolf, Graue, Wiz~enwiki, Minesweeper, Nanshu, Jikanbae, Notheruser, Александър, LittleDan, Lupinoid, Poor Yorick, Nikai, Cadr, Mxn, Drz~enwiki, Hashar, Guaka, Dcoetzee, Andrevan, Jm34harvey, Dysprosia, Tb, Jogloran, Hao2lian, Furrykef, Bevo, Mignon~enwiki, Pilaf~enwiki, JackH, Northgrove, Phil Boswell, Chuunen Baka, Robbot, Chealer, Craig Stuntz, Fredrik, R3m0t, RedWolf, Coop, Joeljkp, Carlj7, Anthony, Connelly, Centrx, Giftlite, Dinomite, Homeobocks, Fukumoto, Ds13, Georgesawyer, Ssd, Ceejayoz, Bovlb, Jorge Stolfi, AlistairMcMillan, Pne, Uzume, Dfrankow, Neilc, Stevietheman, Pgan002, Knutux, LiDaobing, Yath, Sonjaaa, Beland, Dasch, Kusunose, Apotheon, Andyabides, Oneiros, ReiniUrban, Maximaximax, Halo, Bravestar, Austin Hair, Srittau, Frau Holle, JavaTenor, Ratiocinate, Dv~enwiki, RandalSchwartz, Corti, Mernen, Shipmaster, Discospinster, Rich Farmbrough, Till Ulen, Hydrox, Wrp103, Metamatic, Lulu of the Lotus-Eaters, Antonio Cangiano, Gronky, Blade Hirato~enwiki, Bender235, Kbh3rd, LemRobotry, Nmagedman, Danakil, Spitzak, Nick Nolan, Pmcm, LordRM, Nigelj, AmosWolfe, Gunark, Robhu, Polluks, .:Ajvol:., Unquietwiki, Guido del Confuso, Richi, Mrbill, SpeedyGonsales, Storm Rider, Beinsane, Jhertel, Guy Harris, Atlant, Jeltz, Droob, Barte, Andrewpmk, Atanamir, AzaToth, Sligocki, Jackliddle, PeteVerdon, Ronark, Sphivo, Whayworth, Tony Sidaway, Gpvos, Michaelpb, Mnemo, Stygian23, Ianblair23, Zootm, MIT Trekkie, Paraphelion, HenryLi, Heyseuss, Mcsee, Isfisk, Gmaxwell, Ben White, Zudduz, Mindmatrix, Bkkbrad, Timosa, Ruud Koot, RolandH, Rickjpelleg, Robertwharvey, Slocombe, Bowman, GregorB, Kaster, CPES, Teemu Leisti, Turnstep, Marudubshinki, Dysepsion, SteveCrook, Qwertyus, Li-sung, DanielAmelang, Yurik, Rjwilmsi, Chipuni, DeadlyAssassin, MarSch, Gudeldar, CalPaterson, The wub, Mohawkjohn, Yahoolian, AlisonW, Drbrain, Yamamoto Ichiro, FlaBot, Strangnet, Crazycomputers, GünniX, Thomas Linder Puls, Fragglet, Rbonvall, Jrtayloriv, Agusti~enwiki, Rob*, Windharp, King of Hearts, Jmorgan, Daev, Chobot, Daekharel, Bobdc, Peterl, YurikBot, Tthorley, Wavelength, Hairy Dude, Masiarek, RussBot, Hyad, Kengo.nakajima, Khatharr, Van der Hoorn, Gaius Cornelius, Gustavb, Paradoja, Mipadi, DanKohn, Michalis Famelis, Derex, Francis Ocoma, Eipipuz, Kain2396, MySchizoBuddy, Zwobot, Dbfirs, EEMIV, Snarius, Jessemerriman, RyanJones, MGPCoe, Tcooke, Black Falcon, Amccaf1, Rwxrwxrwx, FF2010, Closedmouth, Cedar101, LeonardoRob0t, Donhalcon, Dublinclontarf, Mujaki~enwiki, GrEp, KKL, Owl-syme, Tom Morris, NickelShoe, Robertd, P69, SmackBot, Laughing Man, Faisal.akeel, Paolino~enwiki, Wegesrand, Allixpeeke, Renku, Anastrophe, GeneralAntilles, Scott Paeth, Aceofspades1217, Flankk, Unforgettableid, Gilliam, Ohnoitsjamie, Betacommand, Bluebot, Faya, Thumperward, Oli Filth, Al Hart, MalafayaBot, Mithaca, Nbarth, Mcaruso, Colonies Chris, Kmactane, JGXenite, Joerite, Frap, Stevemidgley, GeorgeMoney, -Barry-, James Britt, Cybercobra, SnappingTurtle, Geoffr, The Extremist, Evolve2k, A5b, Mrego, Eigensoul, BurnDownBabylon, Gobonobo, Korean alpha for knowledge, Ksn, Kokot.kokotisko, Soumyasch, Tlesher, Joshua Scott, Baator, Kopf1988, Panglossa, Tmcw, Sharcho, EdC~enwiki, Pjrm, Afkbot, Ziplux~enwiki, Ashleyjoyce, Jason.grossman, Dreftymac, Lasah, Digitalsurgeon, Sander Säde, LethargicParasite, Tawkerbot2, MonkeeSage, Ioannes Pragensis, SkyWalker, HDCase, Doceddi, Perle, CRGreathouse, PuerExMachina, Sofy.Basir, SimpleBeep, AnibalRojas, WeggeBot, HenkeB, KevinScott, NE Ent, DarkseidX, MojoX, Cydebot, Gogo Dodo, Tfgbd, Countchoc, Chasingsol, Headius, Pdq, Bayonetblaha, MicahElliott, Arb, Thijs!bot, David from Downunder, Hervegirod, Ajpeters, A3RO, Paxinum, Danshep79, Escarbot, Ianozsvald, I already forgot, Su30, Gioto, Widefox, Jeremy Devenport, Seaphoto, Gczffl, KKong, LazyEditor, Isilanes, KMeyer, Hexene, Maslin, Trappist, Mhagerman, JAnDbot, Bailey.d.r, MER-C, Mcorazao, DPoon, Sunnyoraish, VoABot II, AurakDraconian, Usien6, Galactic Hitchhiker, Jatkins, Loqi, SwiftBot, Destynova, Hu2hugo, Sp0ng, Lensboel, Herraotic~enwiki, Gwern, Radnam, Zoquero, Snood1205, Garkbit, RockMFR, J.delanoy, St.daniel, McSly, Hessammehr, RenniePet, Midnight Madness, Rwessel, Diego.viola, Cbenz, STBotD, Huw Collingbourne, DorganBot, Kvdveer, MartinRinehart, Black Walnut, Dvyjones, Mkarlesky, AlnoktaBOT, TheOtherJesse, SlurpTheo, Philip Trueman, PGSONIC, TXiKiBoT, Bspeakmon, Jazzwick, Vanished user ikijeirw34iuaeolaseriffic, Arash Hemmat, Genium, JhsBot, Busbeytheelder, Micropolygon, Rogerdpack, Synthebot, Matttemp, Wikiivan13, AlleborgoBot, SieBot, TJRC, Evolvingjerk, Malcolmxl5, BotMultichill, Igouy, Michaelneale, Duplicity, Jerryobject, WRK, FF-Wonko, Aruton, Macy, Winterheat, Pianoman320, Shadowfirebird, Stillwaterising, Vanyo, PsyberS, VanishedUser sdu9aya9fs787sads, ClueBot, Rule.rule, Kl4m, CiudadanoGlobal, Justin W Smith, Wikievil666, The Thing That Should Not Be, Kl4m-AWB, Drmies, Xavexgoem, Niceguyedc, ArlenCuss, Excirial, Justapersona, Watchduck, Anon lynx, JimJavascript, Totie, Wemagor, J.Gowers, TheRedPenOfDoom, SeanPage31, TheAstonishingBadger, Alexey Muranov, Zeljko.filipin, DumZiBoT, Supermatique, Spitfire, Koolabsol, Wertuose, Addbot, Proofreader77, Wisdomgroup, Raypereda, Scientus, TheCritisizer, Dyadron, Morning277, SpBot, LinkFA-Bot, Paulanunda, Hilltopperpete, Nonpr3, DILIN, Ricvelozo, Jarble, Luckas-bot, Yobot, The Earwig, Worm That Turned, Flankk2, AnomieBOT, Jim1138, Wickorama, Gc9580, Ulric1313, TheMathinator, ArthurBot, Xqbot, Unixphil, Jedediah Smith, Donpayette, Aphades, Flying sheep, Anna Frodesiak, Bmazuyer, Pradameinhoff, Arcnova, Mastazi, Shadowjams, WhatisFeelings?, Thehelpfulbot, FrescoBot, D'ohBot, Lantius, Sindhu sundar, Esjs, Salvan, Winterst, Fabio.kon, CodeBlock, Yserbnaayj, Jschnur, Σ, Jamieorc, Weylinp, TobeBot, Ale And Quail, Gregwebs, 777sms, Deeep.deep, Angelito7, Jfmantis, Theturtle32, LinguistManiac, Tomchen1989, Jowa fan, Mochaman69, EmausBot, Merrells, ThePCKid, Wham Bam Rock II, The Mysterious El Willstro, DSearle, BigFloppyDonkey1234, Gimlinu, Upsetspecial, D'oh!, Wes.turner, Grunny, Adambenjaminlowe, Demonkoryu, Friendocity, Benjaminoakes, Rcsprinter123, L Kensington, Jcubic, Ipsign, ChuispastonBot, ShotaFukumori, Heja2009, Mister Mormon, Mattsenate, Will Beback Auto, ClueBot NG, Neuroneutron, Catlemur, Snotbot, Srchulo, Widr, Antiqueight, WikiPuppies, Meniv, Helpful Pixie Bot, Abhishekanand25, BG19bot, KamranMackey, Hza a 9, StudioFortress, MusikAnimal, Compfreak7, Saikumarganji, Ubergeek3141, Dominikhonnef, Jc00ke, Cyberbot II, Dev65, ChrisGualtieri, Andrewgggg, Dexbot, SevanMilis, Codename Lisa, Bdscvr, Olonic, Stickmangumby, DJStarrfish, Moizk, Starflyer59, MCaecilius, Pimp slap the funk, UltimateSupreme, Icarot, Captain Conundrum, Myconix, Jdaudier, Comp.arch, Huihermit, Melody Lavender, My name is not dave, Boto-boto, Sam Sailor, JaconaFrere, AdamPro, Arbox, Itamarkalimi, Aaronbartell, Eman235, Sethxia, TerryAlex, LibertyChick1776, Gvaresh, Sigehelmus, Thetechgirl, Viam Ferream, Mapfap, Fiveismorethanfour, GustavoTemple, Lemondoge, Sashahhh, Transfat0g, -trrriple, Siaw23, CLCStudent, V975, Startledmarmot, RailsDuck, Bigpasty679, Formiganinja and Anonymous: 769

16.2 Images

- **File:8bit-dynamiclist_(reversed).gif** *Source:* https://upload.wikimedia.org/wikipedia/commons/c/cc/8bit-dynamiclist_%28reversed%29.gif *License:* CC-BY-SA-3.0 *Contributors:* This file was derived from: 8bit-dynamiclist.gif
Original artist: Seahen, User:Rezonansowy

- **File:Commons-logo.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/4/4a/Commons-logo.svg *License:* CC-BY-SA-3.0 *Contributors:* ? *Original artist:* ?

- **File:Folder_Hexagonal_Icon.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/4/48/Folder_Hexagonal_Icon.svg *License:* Cc-by-sa-3.0 *Contributors:* ? *Original artist:* ?
- **File:Free_and_open-source_software_logo_(2009).svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/3/31/Free_and_open-source_software_logo_%282009%29.svg *License:* Public domain *Contributors:* FOSS Logo.svg *Original artist:* Free Software Portal Logo.svg (FOSS Logo.svg): ViperSnake151
- **File:Portal-puzzle.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/f/fd/Portal-puzzle.svg *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:Question_book-new.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/9/99/Question_book-new.svg *License:* Cc-by-sa-3.0 *Contributors:*

  Created from scratch in Adobe Illustrator. Based on Image:Question book.png created by User:Equazcion *Original artist:*
  Tkgd2007
- **File:Ruby_logo_64x64.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/f/f1/Ruby_logo_64x64.png *License:* CC BY-SA 2.5 *Contributors:* http://rubyidentity.org/ *Original artist:* Yukihiro Matsumoto
- **File:Symbol_book_class2.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/8/89/Symbol_book_class2.svg *License:* CC BY-SA 2.5 *Contributors:* Mad by Lokal_Profil by combining: *Original artist:* Lokal_Profil
- **File:Symbol_list_class.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/d/db/Symbol_list_class.svg *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:Symbol_neutral_vote.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/8/89/Symbol_neutral_vote.svg *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:Wikibooks-logo-en-noslogan.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/d/df/Wikibooks-logo-en-noslogan.svg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* User:Bastique, User:Ramac et al.
- **File:Wikiquote-logo.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/f/fa/Wikiquote-logo.svg *License:* Public domain *Contributors:* Own work *Original artist:* Rei-artur
- **File:Wikiversity-logo.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/9/91/Wikiversity-logo.svg *License:* CC BY-SA 3.0 *Contributors:* Snorky (optimized and cleaned up by verdy_p) *Original artist:* Snorky (optimized and cleaned up by verdy_p)
- **File:Yukihiro_Matsumoto.JPG** *Source:* https://upload.wikimedia.org/wikipedia/commons/7/76/Yukihiro_Matsumoto.JPG *License:* Public domain *Contributors:* Yukihiro Matsumoto. Originally uploaded by Cep21 to English Wikipedia. *Original artist:* Cep21

16.3 Content license

- Creative Commons Attribution-Share Alike 3.0