WIKIPEDIA

Ruby (programming language)

Ruby is an interpreted, high-level, general-purpose programming language. It was designed and developed in the mid-1990s by Yukihiro "Matz" Matsumoto in Japan.

Ruby is dynamically typed and uses garbage collection and just-in-time compilation. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. According to the creator, Ruby was influenced by Perl, Smalltalk, Eiffel, Ada, BASIC, and Lisp.[11][3]

Contents

History

Early concept

Matsumoto has said that Ruby was conceived in 1993. In a 1999 post to the *ruby-talk* mailing list, he describes some of his early ideas about the language:[12]

Ruby

Paradigm	Multi-paradigm: functional, imperative, object-oriented, reflective
Designed by	Yukihiro Matsumoto
Developer	Yukihiro Matsumoto, et al.
First appeared	1995
Stable release	3.0.3[1] / 24 November 2021
Typing discipline	Duck, dynamic, strong
Scope	Lexical, sometimes dynamic
Implementation language	C
OS	Cross-platform
License	Ruby License
Filename extensions	.rb
Website	www.ruby-lang.org (https://www.ruby-lang.org/)
Major implementations	
Ruby MRI, TruffleRuby, YARV, Rubinius, MagLev, JRuby, MacRuby, RubyMotion, Mruby, IronRuby	
Influenced by	
Ada,[2] Basic,[3] C++,[2] CLU,[4] Dylan,[4] Eiffel,[2] Lisp,[4] Lua, Perl,[4] Python,[4] Smalltalk[4]	

> 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.

Influenced
Clojure, CoffeeScript, Crystal, D, Elixir, Groovy, Ioke,[5] Julia,[6] Mirah, Nu,[7] Ring,[8] Rust,[9] Swift[10]
Ruby Programming at Wikibooks

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]

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]

Early releases

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]

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* (オブジェクト指向スクリプト言語

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 non-Japanese speaking world.

Ruby 1.8 and 1.9

Ruby 1.8 was initially released 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][26]

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.[27]

Effective with Ruby 1.9.3, released October 31, 2011,[28] 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.[29] 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. Examples include:[30]

- 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 }`
- an additional Hash literal syntax using colons for symbol keys: `{symbol_key: "value"} == {:symbol_key => "value"}`
- per-string character encodings are supported
- new socket API (IPv6 support)
- `require_relative` import security

Ruby 2

Ruby 2.0 was 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.[31] 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.[32]

Starting with 2.1.0, Ruby's versioning policy changed to be more similar to semantic versioning.[33]

Ruby 2.2.0 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. Since version 2.2.1,[34] Ruby MRI performance on PowerPC64 was improved.[35][36][37] 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.[38]

Ruby 2.3.0 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.[39] Other notable changes in include:

- The ability to mark all string literals as frozen by default with a consequently large performance increase in string operations.[40]
- 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' } } }`, 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.

Ruby 2.4.0 includes performance improvements to hash table, Array#max, Array#min, and instance variable access.[41] Other notable changes include:

- Binding#irb: Start a REPL session similar to binding.pry
- Unify *Fixnum* and *Bignum* into *Integer* class
- String supports Unicode case mappings, not just ASCII
- A new method, Regexp#match?, which is a faster boolean version of Regexp#match
- Thread deadlock detection now shows threads with their backtrace and dependency

A few notable changes in Ruby 2.5.0 include *rescue* and *ensure* statements automatically use a surrounding *do-end* block (less need for extra *begin-end* blocks), method-chaining with *yield_self*, support for branch coverage and method coverage measurement, and easier Hash transformations with *Hash#slice* and *Hash#transform_keys* On top of that come a lot of performance improvements like faster block passing (3 times faster), faster Mutexes, faster ERB templates and improvements on some concatenation methods.

A few notable changes in Ruby 2.6.0 include an experimental just-in-time compiler (JIT), and *RubyVM::AbstractSyntaxTree* (experimental).

A few notable changes in Ruby 2.7.0 include pattern Matching (experimental), REPL improvements, a compaction GC, and separation of positional and keyword arguments.

Ruby 3

Ruby 3.0.0 was released on Christmas Day in 2020.[42] It is known as Ruby 3x3. One of its main aims was to switch the interpreter to a Just-In-Time Compiler, to make programs faster.

Ruby 3.1 is planned to be released on Christmas Day in 2021.[43]

Table of versions

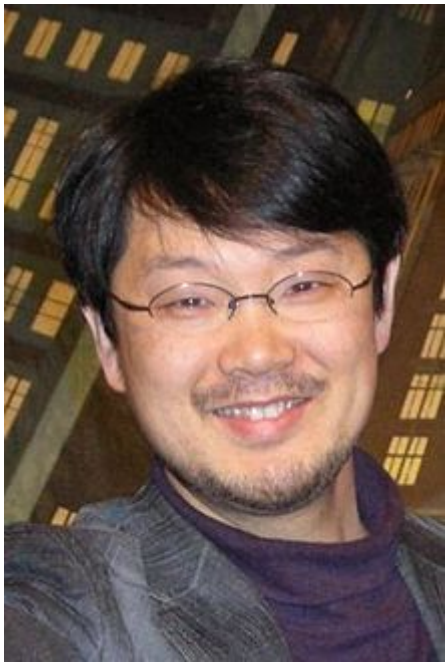
Version	Latest teeny version	Initial release date	End of support phase	End of security maintenance phase
1.0	NA	1996-12-25[44]	NA	NA
1.8	1.8.7-p375[45]	2003-08-04[46]	2012-06[47]	2014-07-01[48]
1.9	1.9.3-p551[49]	2007-12-25[50]	2014-02-23[51]	2015-02-23[52]
2.0	2.0.0-p648[53]	2013-02-24[54]	2015-02-24[53]	2016-02-24[53]
2.1	2.1.10[55]	2013-12-25[56]	2016-03-30[57][58]	2017-03-31[59][60]
2.2	2.2.10[61]	2014-12-25[62]	2017-03-28[63]	2018-03-31[60]
2.3	2.3.8[64]	2015-12-25[65]	2018-06-20[66]	2019-03-31[66]
2.4	2.4.10[67]	2016-12-25[68]	2019-04-01[69]	2020-04-01[69]
2.5	2.5.9[70]	2017-12-25[71]	2021-04-05[70]	2021-04-05[70]
2.6	2.6.8[72]	2018-12-25[73]	2021-04-05[72]	2022-04-05[72]
2.7	2.7.4[74]	2019-12-25[75]	TBA	TBA
3.0	3.0.2[76]	2020-12-25[77]	TBA	TBA
3.1	3.1.0[43]	2021-12-25[43]	TBA	TBA

Legend: Old version | Older version, still maintained | **Latest version** | Future release

Semantics and philosophy

Matsumoto has said that Ruby is designed for programmer productivity and fun, following the principles of good user interface design.[78] 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."[79] He stresses that systems design needs to emphasize human, rather than computer, needs:[80]

> 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.



Yukihiro Matsumoto, the creator of Ruby

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 (POLA) to the design of Ruby;[80] 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.[81]

Matsumoto defined it this way in an interview:[80]

> 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.

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 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.[82]

Features

- Thoroughly object-oriented with inheritance, mixins and metaclasses[83]
- Dynamic typing and duck typing
- Everything is an expression (even statements) and everything is executed imperatively (even declarations)
- Succinct and flexible syntax[84] that minimizes syntactic noise and serves as a foundation for domain-specific languages[85]
- Dynamic reflection and alteration of objects to facilitate metaprogramming[86]
- Lexical closures, iterators and generators, with a block syntax[87]

- 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[88]
- 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)
- Support for Unicode and multiple character encodings.
- Native plug-in API in C
- Interactive Ruby Shell, an interactive command-line interpreter that can be used to test code quickly (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[89]
- Just-in-time compilation

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 defined by either 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.[90][91] 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 from 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 trade-off 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.

Implementations

Matz's Ruby interpreter

The original Ruby interpreter is often referred to as 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.

Alternate implementations

As of 2018, there are a number of alternative implementations of Ruby, including JRuby, Rubinius, and mruby. Each takes a different approach, with JRuby and Rubinius providing just-in-time compilation and mruby also providing ahead-of-time compilation.

Ruby has three major alternate implementations:

- JRuby, a mixed Java and Ruby implementation that runs on the Java virtual machine. JRuby currently targets Ruby 2.5.
- TruffleRuby, a Java implementation using the Truffle language implementation framework with GraalVM
- 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 2.3.1.

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
- 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
- julializer, a transpiler (partial) from Ruby to Julia. It can be used for a large speedup over e.g. Ruby or JRuby implementations (may only be useful for numerical code).[92]
- Topaz, a Ruby implementation written in Python
- Opal, a web-based interpreter that compiles Ruby to JavaScript

Other now defunct Ruby implementations were:

- MacRuby, a Mac OS X implementation on the Objective-C runtime. Its iOS counterpart is called RubyMotion
- 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
- HotRuby, a JavaScript and ActionScript implementation of the Ruby programming language

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, JRuby, and Rubinius[93] are all able to run Rails unmodified in a production environment.

Platform support

Matsumoto originally developed Ruby on the 4.3BSD-based Sony NEWS-OS 3.x, but later migrated his work to SunOS 4.x, and finally to Linux.[94][95]

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.[96]

Modern Ruby versions and implementations are available on many operating systems, such as Linux, BSD, Solaris, AIX, macOS, Windows, Windows Phone,[97] Windows CE, Symbian OS, BeOS, OpenVMS, and IBM i.

Ruby programming language is supported across a number of cloud hosting platforms like Jelastic, Heroku, Google Cloud Platform and others.

Repositories and libraries

RubyGems is Ruby's package manager. A Ruby package is called a "gem" and can be installed via the command line. Most gems are libraries, though a few exist that are applications, such as IDEs.[98] There are over 10,000 Ruby gems hosted on RubyGems.org (http://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.[99]

See also

- Comparison of programming languages
- Metasploit Project
- Why's (poignant) Guide to Ruby
- XRuby
- Crystal (programming language)

References

1. https://www.ruby-lang.org/en/news/2021/11/24/ruby-3-0-3-released/.
2. Cooper, Peter (2009). *Beginning Ruby: From Novice to Professional*. Beginning from Novice to Professional (2nd ed.). Berkeley: APress. p. 101. ISBN 978-1-4302-2363-4. "To a lesser extent, Python, LISP, Eiffel, Ada, and C++ have also influenced Ruby."
3. "Reasons behind Ruby" (https://confreaks.tv/videos/rubyconf2008-reasons-behind-ruby). *Ruby Conference 2008*. Confreaks TV. Retrieved 2019-06-25.

4. Bini, Ola (2007). *Practical JRuby on Rails Web 2.0 Projects: Bringing Ruby on Rails to Java* (https://archive.org/details/practicaljrubyon0000bini/page/3). Berkeley: APress. p. 3 (https://archive.org/details/practicaljrubyon0000bini/page/3). ISBN 978-1-59059-881-8. "It draws primarily on features from Perl, Smalltalk, Python, Lisp, Dylan, and CLU."
5. Bini, Ola. "Ioke" (https://web.archive.org/web/20110721091046/http://www.ioke.org/). *Ioke.org*. Archived from the original (http://ioke.org/) on 2011-07-21. Retrieved 2011-07-21. "inspired by Io, Smalltalk, Lisp and Ruby"
6. "Julia 1.0 Documentation: Introduction" (https://docs.julialang.org/en/stable/). Retrieved 6 October 2018.
7. Burks, Tim. "About Nu™" (http://programming.nu/about). *Programming Nu™*. Neon Design Technology, Inc. Retrieved 2011-07-21.
8. Ring Team (3 December 2017). "Ring and other languages" (http://ring-lang.sourceforge.net/doc1.6/introduction.html#ring-and-other-languages). *ring-lang.net*. ring-lang.
9. "The Rust Reference" (https://doc.rust-lang.org/reference/influences.html). Retrieved 16 November 2019.
10. Lattner, Chris (2014-06-03). "Chris Lattner's Homepage" (http://nondot.org/sabre/). 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" (https://www.ruby-lang.org/en/about/). Retrieved 15 February 2020.
12. Shugo Maeda (17 December 2002). "The Ruby Language FAQ" (https://www.ruby-doc.org/docs/ruby-doc-bundle/FAQ/FAQ.html). Retrieved 2 March 2014.
13. Matsumoto, Yukihiro (13 February 2006). "Re: Ruby's lisp features" (http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/179642). Retrieved 15 February 2020.
14. "History of Ruby" (http://blog.nicksieger.com/articles/2006/10/20/rubyconf-history-of-ruby).
15. "[FYI: historic] The decisive moment of the language name Ruby. (Re: [ANN] ruby 1.8.1)" (http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/88819) (E-mail from Hiroshi Sugihara to ruby-talk).
16. "1.3 Why the name 'Ruby'?" (https://www.ruby-doc.org/docs/ruby-doc-bundle/FAQ/FAQ.html). *The Ruby Language FAQ*. Ruby-Doc.org. Retrieved April 10, 2012.
17. Yukihiro Matsumoto (June 11, 1999). "Re: the name of Ruby?" (http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/394). *Ruby-Talk* (Mailing list). Retrieved April 10, 2012.
18. "More archeolinguistics: unearthing proto-Ruby" (https://web.archive.org/web/20151106023204/http://eigenclass.org/hiki/ruby+0.95). Archived from the original (http://eigenclass.org/hiki/ruby+0.95) on 6 November 2015. Retrieved 2 May 2015.
19. "[ruby-talk:00382] Re: history of ruby" (http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/382). Retrieved 2 May 2015.
20. "[ruby-list:124] TUTORIAL — ruby's features" (http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-list/124). Retrieved 2 May 2015.
21. "An Interview with the Creator of Ruby" (http://www.linuxdevcenter.com/pub/a/linux/2001/11/29/ruby.html).
22. Yukihiro Matsumoto (October 2000). "Programming Ruby: Forward" (http://ruby-doc.com/docs/ProgrammingRuby/html/foreword.html). Retrieved 5 March 2014.
23. "We retire Ruby 1.8.7" (https://www.ruby-lang.org/en/news/2013/06/30/we-retire-1-8-7/). Retrieved 2 May 2015.
24. "IPA 独立行政法人 情報処理推進機構：プレス発表 プログラム言語RubyのJIS規格（JIS X 3017）制定について" (http://www.ipa.go.jp/about/press/20110322.html). Retrieved 2 May 2015.

25. "IPA 独立行政法人 情報処理推進機構：プレス発表 プログラム言語Ruby、国際規格として承認" (http://www.ipa.go.jp/about/press/20120402_2.html). Retrieved 2 May 2015.
26. "ISO/IEC 30170:2012" (https://www.iso.org/standard/59579.html). Retrieved 2017-03-10.
27. Web Development: Ruby on Rails (http://www.devarticles.com/c/a/Ruby-on-Rails/Web-Development-Ruby-on-Rails/). Devarticles.com (2007-03-22). Retrieved on 2013-07-17.
28. "Ruby 1.9.3 p0 is released" (https://www.ruby-lang.org/en/news/2011/10/31/ruby-1-9-3-p0-is-released/). ruby-lang.org. October 31, 2011. Retrieved February 20, 2013.
29. "v1_9_3_0/NEWS" (https://svn.ruby-lang.org/repos/ruby/tags/v1_9_3_0/NEWS). *Ruby Subversion source repository*. ruby-lang.org. September 17, 2011. Retrieved February 20, 2013.
30. Ruby 1.9: What to Expect (http://slideshow.rubyforge.org/ruby19.html). slideshow.rubyforge.org. Retrieved on 2013-07-17.
31. Endoh, Yusuke. (2013-02-24) Ruby 2.0.0-p0 is released (http://www.ruby-lang.org/en/news/2013/02/24/ruby-2-0-0-p0-is-released/#label-8). Ruby-lang.org. Retrieved on 2013-07-17.
32. Endoh, Yusuke. (2013-02-24) Ruby 2.0.0-p0 is released (https://www.ruby-lang.org/en/news/2013/02/24/ruby-2-0-0-p0-is-released/). Ruby-lang.org. Retrieved on 2013-07-17.
33. "Semantic Versioning starting with Ruby 2.1.0" (https://www.ruby-lang.org/en/news/2013/12/21/semantic-versioning-after-2-1-0/). December 21, 2013. Retrieved December 27, 2013.
34. Gustavo Frederico Temple Pedrosa, Vitor de Lima, Leonardo Bianconi (2015). "Ruby 2.2.1 Released" (https://www.ruby-lang.org/en/news/2015/03/03/ruby-2-2-1-released). Retrieved 12 July 2016.
35. Gustavo Frederico Temple Pedrosa, Vitor de Lima, Leonardo Bianconi (2015). "v2.2.1 ChangeLog" (https://svn.ruby-lang.org/repos/ruby/tags/v2_2_1/ChangeLog). Retrieved 12 July 2016.
36. Gustavo Frederico Temple Pedrosa, Vitor de Lima, Leonardo Bianconi (2014). "Specifying non volatile registers for increase performance in ppc64" (https://bugs.ruby-lang.org/issues/9997). Retrieved 12 July 2016.
37. Gustavo Frederico Temple Pedrosa, Vitor de Lima, Leonardo Bianconi (2014). "Specifying MACRO for increase performance in ppc64" (https://bugs.ruby-lang.org/issues/10081). Retrieved 12 July 2016.
38. "ruby/NEWS at v2_2_0 · ruby/ruby · GitHub" (https://github.com/ruby/ruby/blob/v2_2_0/NEWS). *GitHub*. Retrieved 2 May 2015.
39. "Ruby/NEWS at v.2_3_0 - ruby/ruby" (https://github.com/ruby/ruby/blob/v2_3_0/NEWS). *GitHub*. Retrieved 25 December 2015.
40. "Ruby 2.3.0 changes and features" (http://dev.mensfeld.pl/2015/11/ruby-2-3-0-changes-and-features/#frozen). *Running with Ruby*. dev.mensfeld.pl. 14 November 2015.
41. "Ruby 2.4.0 Released" (https://www.ruby-lang.org/en/news/2016/12/25/ruby-2-4-0-released/). *www.ruby-lang.org*. Retrieved 2016-12-30.
42. "Ruby 3.0.0 Released" (https://www.ruby-lang.org/en/news/2020/12/25/ruby-3-0-0-released/). *Ruby Programming Language*. 2020-12-25. Retrieved 2020-12-25.
43. "Ruby 3.1" (https://bugs.ruby-lang.org/versions/37). *bugs.ruby-lang.org*.
44. "The Ruby Community's Christmas Releases" (http://www.rubyinside.com/the-ruby-communitys-christmas-releases-4118.html). *www.rubyinside.com*.
45. "A Patch in Time: Securing Ruby" (https://blog.heroku.com/archives/2013/12/5/a_patch_in_time_securing_ruby).
46. "ruby-1.8.0 released!" (https://www.ruby-lang.org/en/news/2003/08/04/ruby-180-released/). *www.ruby-lang.org*.
47. "Plans for 1.8.7" (https://www.ruby-lang.org/en/news/2011/10/06/plans-for-1-8-7/). *www.ruby-lang.org*.

48. "EOL for Ruby 1.8.7 and 1.9.2" (https://www.ruby-lang.org/en/news/2014/07/01/eol-for-1-8-7-and-1-9-2/). *www.ruby-lang.org*.
49. "Ruby 1.9.3-p551 Released" (https://www.ruby-lang.org/en/news/2014/11/13/ruby-1-9-3-p551-is-released/). *www.ruby-lang.org*.
50. "Ruby 1.9.0 Released" (https://www.ruby-lang.org/en/news/2007/12/25/ruby-1-9-0-released/). *www.ruby-lang.org*.
51. "Support for Ruby version 1.9.3 will end on February 23, 2015" (https://www.ruby-lang.org/en/news/2014/01/10/ruby-1-9-3-will-end-on-2015/). *www.ruby-lang.org*.
52. "Support for Ruby 1.9.3 has ended" (https://www.ruby-lang.org/en/news/2015/02/23/support-for-ruby-1-9-3-has-ended/). *www.ruby-lang.org*.
53. "Ruby 2.0.0-p648 Released" (https://www.ruby-lang.org/en/news/2015/12/16/ruby-2-0-0-p648-released/). *www.ruby-lang.org*.
54. "Ruby 2.0.0-p0 is released" (https://www.ruby-lang.org/en/news/2013/02/24/ruby-2-0-0-p0-is-released/). *www.ruby-lang.org*.
55. "Ruby 2.1.10 Released" (https://www.ruby-lang.org/en/news/2016/04/01/ruby-2-1-10-released/). *www.ruby-lang.org*.
56. "Ruby 2.1.0 is released" (https://www.ruby-lang.org/en/news/2013/12/25/ruby-2-1-0-is-released/). *www.ruby-lang.org*.
57. "Support plans for Ruby 2.0.0 and Ruby 2.1" (https://www.ruby-lang.org/en/news/2016/02/24/support-plan-of-ruby-2-0-0-and-2-1/). *www.ruby-lang.org*.
58. "Ruby 2.1.9 Released" (https://www.ruby-lang.org/en/news/2016/03/30/ruby-2-1-9-released/). *www.ruby-lang.org*.
59. "Ruby Issue Tracking System" (https://bugs.ruby-lang.org/projects/ruby/wiki/ReleaseEngineering). *bugs.ruby-lang.org*.
60. "Support of Ruby 2.1 has ended" (https://www.ruby-lang.org/en/news/2017/04/01/support-of-ruby-2-1-has-ended/). *www.ruby-lang.org*.
61. "Ruby 2.2.10 Released" (https://www.ruby-lang.org/en/news/2018/03/28/ruby-2-2-10-released/). *www.ruby-lang.org*.
62. "Ruby 2.2.0 Released" (https://www.ruby-lang.org/en/news/2014/12/25/ruby-2-2-0-released/). *www.ruby-lang.org*.
63. "Ruby 2.2.7 Released" (https://www.ruby-lang.org/en/news/2017/03/28/ruby-2-2-7-released/). *www.ruby-lang.org*.
64. "Ruby 2.3.8 Released" (https://www.ruby-lang.org/en/news/2018/10/17/ruby-2-3-8-released/). *www.ruby-lang.org*.
65. "Ruby 2.3.0 Released" (https://www.ruby-lang.org/en/news/2015/12/25/ruby-2-3-0-released/). *www.ruby-lang.org*.
66. "Support of Ruby 2.2 has ended" (https://www.ruby-lang.org/en/news/2018/06/20/support-of-ruby-2-2-has-ended/). *www.ruby-lang.org*.
67. "Ruby 2.4.10 Released" (https://www.ruby-lang.org/en/news/2020/03/31/ruby-2-4-10-released/). *Ruby Programming Language*. 2020-03-31. Retrieved 2020-04-01.
68. "Ruby 2.4.0 Released" (https://www.ruby-lang.org/en/news/2016/12/25/ruby-2-4-0-released/). *www.ruby-lang.org*.
69. "Support of Ruby 2.4 has ended" (https://www.ruby-lang.org/en/news/2019/04/01/ruby-2-4-6-released/). *www.ruby-lang.org*.
70. "Ruby 2.5.9 Released" (https://www.ruby-lang.org/en/news/2021/04/05/ruby-2-5-9-released/). *Ruby Programming Language*. 2021-04-05. Retrieved 2021-04-05.
71. "Ruby 2.5.0 Released" (https://www.ruby-lang.org/en/news/2017/12/25/ruby-2-5-0-released/). *www.ruby-lang.org*.
72. "Ruby 2.6.8 Released" (https://www.ruby-lang.org/en/news/2021/07/07/ruby-2-6-8-released/). *Ruby Programming Language*. 2021-07-07. Retrieved 2021-08-15.

73. "Ruby 2.6.0 Released" (https://www.ruby-lang.org/en/news/2018/12/25/ruby-2-6-0-released/). *www.ruby-lang.org*.
74. "Ruby 2.7.4 Released" (https://www.ruby-lang.org/en/news/2021/07/07/ruby-2-7-4-released/). *Ruby Programming Language*. 2021-07-07. Retrieved 2021-08-15.
75. "Ruby 2.7.0 Released" (https://www.ruby-lang.org/en/news/2019/12/25/ruby-2-7-0-released/). *www.ruby-lang.org*.
76. "Ruby 3.0.2 Released" (https://www.ruby-lang.org/en/news/2021/07/07/ruby-3-0-2-released/). *Ruby Programming Language*. 2021-07-07. Retrieved 2021-08-15.
77. "Ruby 3.0.0 Released" (https://www.ruby-lang.org/en/news/2020/12/25/ruby-3-0-0-released/). *www.ruby-lang.org*.
78. "The Ruby Programming Language" (http://www.informit.com/articles/article.aspx?p=18225). Retrieved 2 May 2015.
79. Google Tech Talks – Ruby 1.9 (https://www.youtube.com/watch?v=oEkJvvGEtB4) on YouTube
80. Bill Venners. "The Philosophy of Ruby" (http://www.artima.com/intv/ruby4.html). Retrieved 2 May 2015.
81. "Welcome to RUBYWEEKLYNEWS.ORG" (https://web.archive.org/web/20170704073422/http://www.rubyweeklynews.org/20050529). 4 July 2017. Archived from the original on 4 July 2017.
82. "The Ruby Language FAQ: How Does Ruby Stack Up Against...?" (http://www.rootr.net/rubyfaq-2.html). Retrieved 2 May 2015.
83. Bruce Stewart (29 November 2001). "An Interview with the Creator of Ruby" (https://linuxdevcenter.com/pub/a/linux/2001/11/29/ruby.html). O'Reilly Media. Retrieved 2 May 2015.
84. Bill Venners. "Dynamic Productivity with Ruby" (http://www.artima.com/intv/tuesday3.html). Retrieved 2 May 2015.
85. "Language Workbenches: The Killer-App for Domain Specific Languages?" (http://martinfowler.com/articles/languageWorkbench.html). *martinfowler.com*. Retrieved 2 May 2015.
86. "Ruby – Add class methods at runtime" (http://www.codeproject.com/useritems/Ruby_Dynamic_Methods.asp).
87. Bill Venners. "Blocks and Closures in Ruby" (http://www.artima.com/intv/closures.html). Retrieved 2 May 2015.
88. "Methods" (https://www.ruby-lang.org/en/documentation/faq/7/). *Official Ruby FAQ*.
89. Britt, James. "Ruby 2.0.0 Standard Library Documentation" (https://www.ruby-doc.org/stdlib-2.0.0/). Retrieved 2013-12-09.
90. "[ruby-talk:01120] Re: The value of while..." (http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/1120) "In Ruby's syntax, statement is just a special case of an expression that cannot appear as an argument (e.g. multiple assignment)."
91. "[ruby-talk:02460] Re: Precedence question" (http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/2460). "statement [...] can not be part of expression unless grouped within parentheses."
92. "remove/virtual_module: Born to make your Ruby Code more than 3x faster. Hopefully" (https://github.com/remore/virtual_module). *GitHub*. 21 February 2020.
93. Peter Cooper (2010-05-18). "The Why, What, and How of Rubinius 1.0's Release" (http://www.rubyinside.com/the-why-what-and-how-of-rubinius-1-0-s-release-3261.html).
94. Maya Stodte (February 2000). "IBM developerWorks – Ruby: a new language" (https://web.archive.org/web/20000818164241/http://www-4.ibm.com/software/developer/library/ruby.html). Archived from the original (http://www-4.ibm.com/software/developer/library/ruby.html) on August 18, 2000. Retrieved 3 March 2014.
95. Yukihiro Matsumoto (August 2002). "lang-ruby-general: Re: question about Ruby initial development" (http://osdir.com/ml/lang-ruby-general/2002-08/msg02494.html). Retrieved 3 March 2014.

96. Yukihiro Matsumoto (5 January 1999). "ruby-talk: Re: hah, check these errors" (http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/170). Retrieved 3 March 2014.
97. "Iron Ruby on Windows Phone 7" (http://msdn.microsoft.com/en-us/magazine/ff960707.aspx).
98. "The Ruby Toolbox" (https://www.ruby-toolbox.com). Retrieved 2015-04-04.
99. "We retire raa.ruby-lang.org" (https://www.ruby-lang.org/en/news/2013/08/08/rip-raa/). 2013-08-08. Retrieved 2016-01-03.

Further reading

- Black, David; Leo, Joseph (March 15, 2019), *The Well-Grounded Rubyist* (Third ed.), Manning Publications, p. 584, ISBN 978-1617295218
- Metz, Sandi (August 22, 2018), *Practical Object-Oriented Design: An Agile Primer Using Ruby* (https://www.informit.com/store/practical-object-oriented-design-an-agile-primer-using-9780134456478) (Second ed.), Addison-Wesley Professional, p. 288, ISBN 978-0-13-445647-8
- Cooper, Peter (July 12, 2016), *Beginning Ruby: From Novice to Professional* (Third ed.), Apress, p. 492, ISBN 978-1484212790
- Carlson, Lucas; Richardson, Leonard (April 3, 2015), *Ruby Cookbook: Recipes for Object-Oriented Scripting* (http://oreilly.com/catalog/9781449373719) (Second ed.), O'Reilly Media, p. 963, ISBN 978-1449373719
- Fulton, Hal; Arko, André (March 2, 2015), *The Ruby Way: Solutions and Techniques in Ruby Programming* (https://www.informit.com/store/ruby-way-solutions-and-techniques-in-ruby-programming-9780321714633) (Third ed.), Addison-Wesley Professional, p. 816, ISBN 978-0-321-71463-3
- Thomas, Dave; Fowler, Chad; Hunt, Andy (July 7, 2013), *Programming Ruby 1.9 & 2.0: The Pragmatic Programmers' Guide* (Fourth ed.), Pragmatic Bookshelf, p. 888, ISBN 978-1937785499
- McAnally, Jeremy; Arkin, Assaf (March 28, 2009), *Ruby in Practice* (First ed.), Manning Publications, p. 360, ISBN 978-1933988474
- Flanagan, David; Matsumoto, Yukihiro (January 25, 2008), *The Ruby Programming Language* (https://archive.org/details/rubyprogrammingl00davi/page/446) (First ed.), O'Reilly Media, p. 446 (https://archive.org/details/rubyprogrammingl00davi/page/446), ISBN 978-0-596-51617-8
- Baird, Kevin (June 8, 2007), *Ruby by Example: Concepts and Code* (https://nostarch.com/rubyex) (First ed.), No Starch Press, p. 326, ISBN 978-1593271480
- Fitzgerald, Michael (May 14, 2007), *Learning Ruby* (https://archive.org/details/learningruby0000fitz/page/255) (First ed.), O'Reilly Media, p. 255 (https://archive.org/details/learningruby0000fitz/page/255), ISBN 978-0-596-52986-4

External links

- Official website (https://www.ruby-lang.org/en/)
- Ruby documentation (https://www.ruby-doc.org)
- Ruby (https://curlie.org/Computers/Programming/Languages/Ruby) at Curlie