Ruby Research Paper

Florida State University

Author: Drew Smith
Date: April 22, 2016

Summary:

The document instigates the history of the Ruby programming language, as well as who created it, why it was created, and it's purpose. It dives deep into the features that Ruby offers, the data structures it contains, and the functionality of the language as a whole. The document also describes Ruby's portability, supporting software, frameworks, and syntax that might interest someone to begin programming in Ruby. A huge counterpart of the Ruby programming language is Rails, in which the document explains how the two interacted with each other to become one of the most popular programing languages in the world.

Introduction:

Ruby is a powerful, object-oriented, open-source language with a clean and easily understandable syntax. It's high-level abstraction makes Ruby extremely simple to write, analyze, and debug, as it looks very similar to regular English. Its perfect object-oriented design allows users to effectively manipulate data structures as objects to build and execute professional-grade programs, but is also suitable for procedural and functional programming styles. Ruby's dynamic programming structure contains a complex but expressive grammar and a core class library with a rich and powerful API that works well with its popular counterpart, Rails. Ruby draws inspiration from many other poplar programming languages but uses a grammar that is easy for beginner programmers to learn.

Yukihiro Matsumoto, also known as "Matz", is the creator of Ruby, in which originated in Japan. Matz began Ruby's development in 1993 and completed its initial development in

1995, where he first began introducing the language to local Japanese industries. It was not until the year 2003 where Ruby started to become increasingly popular in English-speaking countries. With the help of its counterpart, Ruby on Rails, created in 2003 by David Heinemeier Hansson, Ruby quickly escalated to one of the world's most accepted programing languages.

The ambition towards the creation of Ruby was to design a programming language that users can use at ease. Matz often said that he is "trying to make Ruby natural, not simple" in a way that mirrors life. He emphasized that humans should not have to think like computers, as they are our slaves. Instead, programming should be a task that humans could easily relate to. Therefore, Matz began analyzing various aspects of other programming languages and concluded to combine the use of Lisp, Smalltalk, Perl, and Python. He anticipated for something more powerful than Perl, but yet more object-oriented than Python. So after two years of development, Matz finalized the creation of the Ruby programing language.

Discussion:

Ruby is a highly portable language that is capable of running on various operating systems including Windows, Linux, Unix, and OS X. Once downloaded, Ruby comes with an interactive interpreter shell that allows users to type in commands into the command line and see immediate results. This is extremely useful for beginner programmers or even experienced users that desire to test different functionality within Ruby. The interactive shell can also be used extensively to test all of Ruby's unique style and features including its

object-oriented design, interpreted language functionality, Meta programming capabilities, and exceptional syntax.

Object-Oriented Design:

In Ruby, everything is an object. Every aspect of code and information within Ruby can be given their own properties and actions, which makes it a perfect object-oriented language. Ruby's object-oriented paradigm was created to deal with the increasing complexity of large software systems, as they were often extremely difficult to maintain. It highly anticipates the use of Encapsulation, Abstraction, Inheritance, and Polymorphism.

Encapsulation:

By using Encapsulation, Ruby is able to hide pieces of functionality by making it unavailable to the user. It is a form of data protection, so that data cannot be manipulated without apparent intentions. Encapsulation is what defines the boundaries in a program and allows code to achieve higher levels of complexity. Ruby, like many other object-oriented languages, accomplishes this task by creating objects, and exposing interfaces to interact with those objects. This benefits the user by allowing the creation of objects, which permit the programmer to think on a higher level of abstraction.

Abstraction:

Abstraction in Ruby is a technique used to manage the complexity of large computer systems. It works by establishing a level of complexity on which a person interacts with the system, suppressing the more complex details below the current level. The programmer works with an idealized interface and can add additional levels of functionality that would otherwise be too complex to handle. Ruby Gems is a package manager for Ruby that

provides a standard format for distributing powerful built-in libraries called “gems” that utilizes the idea of abstraction. This allows code in Ruby to be much easier to read, write and create, as they let the gems do most of the back-end work. Gems in Ruby also closely resemble the use of Inheritance.

Inheritance:

The concept of Inheritance is used in Ruby where a class inherits the behaviors of another class, referred to as the superclass. This gives Ruby programmers the power to define basic classes with large reusability and smaller subclasses for more fine-grained, detailed behaviors. Ruby only supports the use of single inheritance, which a class can only inherit from a single other class. But with the use of modules, Ruby can ultimately eliminate the need for multiple inheritance, providing a facility called a mixin. In Ruby, a mixin is code wrapped up in a module that a class can include or extend. In other words, they can be added to one or more classes to add additional capabilities without using inheritance. In Ruby, a mixin is code wrapped up in a module that a class can include or extend.

Polymorphism:

Polymorphism is the ability for data to be represented as many different types. Ruby’s object-oriented design gives programmers flexibility to use pre-written code for new purposes. Another way to apply polymorphic structure to Ruby programs is to use a Module. Modules are similar to classes in that they contain shared behavior. However, you cannot create an object with a module. A module must be mixed in with a class using the reserved word “include”. This is called a “mixin”. After mixing in a module, the behaviors declared in that module are available to the class and its objects.

Interpreted Language:

Ruby is an interpreted language that uses dynamic programming to help provide greater flexibility to the programmer. Dynamic Programming in Ruby, also referred to as "weakly typed", means that variables are not bound to a particular type until runtime. This allows Ruby programs to completely omit the compile and linking phase and allow programmers to pass in parameters during execution. Although Ruby's interpreted language is faster to develop and provides greater flexibility, it also is less efficient to run. Therefore, Ruby is rarely used for complex algorithms and data structures.

Meta Programming:

Meta programming is also a key feature of Ruby. This allows programmers to write computer programs that are capable or writing and manipulating it's own code or even other programs. In many cases, this allows programmers to get more done in less time. This explains how immensely powerful Meta programming is and all the advantages that a programmer can benefit from utilizing it.

Syntax:

Syntax in Ruby closely resembles the use of natural language, as that was Matz's main objective towards the creation of Ruby. It highly utilizes the use of normal English in order for programmers to comprehend what exactly a program is doing. The absence of types allows Ruby language to be very clean and flexible as any type value can be passed in to any given variable. Also, Ruby neglects the use of semi-colons at the end its statements. The Ruby interpreter automatically distinguishes when a statement or function is finished with the use of powerful key words built into Ruby. Ruby programmers also do not need to use parenthesis to define or call a function, unless they wish to pass in parameters or

arguments. Ruby syntax has arguably been declared, as one of the most desirable syntax's to use within a programming language.

Ruby on Rails:

"Rails" is a popular software library that extends the Ruby programming language. It combines Ruby with HTML, CSS, and JavaScript to create web-based platforms. Since Rails runs on a web server, Rails is considered a "back end" web application development platform. In a larger sense, Rails is more than an API, as it is the central scheme of a massive community that produces software libraries to simplify the creation of complex websites. The Ruby programing language must give an enormous amount of credit to Rails for making it one of the trendiest languages in the modern day programing world.

Conclusion:

Ruby's powerful, interpreted language is increasingly becoming one of the most desirable languages today. The use of it's object-oriented style, superb syntax, and high-level abstraction makes Ruby a great programming language for all levels of programming. Also obtaining the capabilities of procedural and functional programming styles, Ruby's dynamic programming structure will continue to gain massive attention to modern day programmers.

References:

- Matz, Speaking on the Ruby-Talk Mailing List, May 12th, 2000.
- Matz, in An Interview with the Creator of Ruby, Nov. 29th, 2001.
- Matz, in Blocks and Closures in Ruby, December 22nd, 2003. Person, Yehuda Katz, "My 10 Favorite Things About the Ruby Language", 23 Aug. 2009
- Longman, Addison W. "Programming Ruby: The Pragmatic Programmer's Guide." Programming Ruby: The Pragmatic Programmer's Guide, June 2001. Web. 22 Apr. 2016.
- Thomas, David, and Andrew Hunt. "Programming Ruby." Programming Ruby. N.p., n.d. Web. 22 Apr. 2016.