Язык программирования Ruby

CS 212

30 октября 2018 г.

План

1. Описание языка
   - Характеристики языка
   - Актуальность языка
2. Ruby on Rails Doctrine
3. Синтаксис
   - Типы данных
   - Константы
   - Переменные
   - Символы
   - Строки

Историческая справка

Автор: Yukihiro "Matz"Matsumoto
Страна: Япония
Период: Середина девяностых (1993)

Формальные характеристики

1. Динамический
2. Интерпретируемый,
3. Рефлексивный,
4. Объектно-ориентированный,
5. Язык общего назначения

Динамический язык

Динамический язык — язык программирования, который позволяет определять типы данных и осуществлять синтаксический анализ и компиляцию на этапе выполнения программы. Динамическая типизация является основным, но не единственным критерием динамического языка программирования

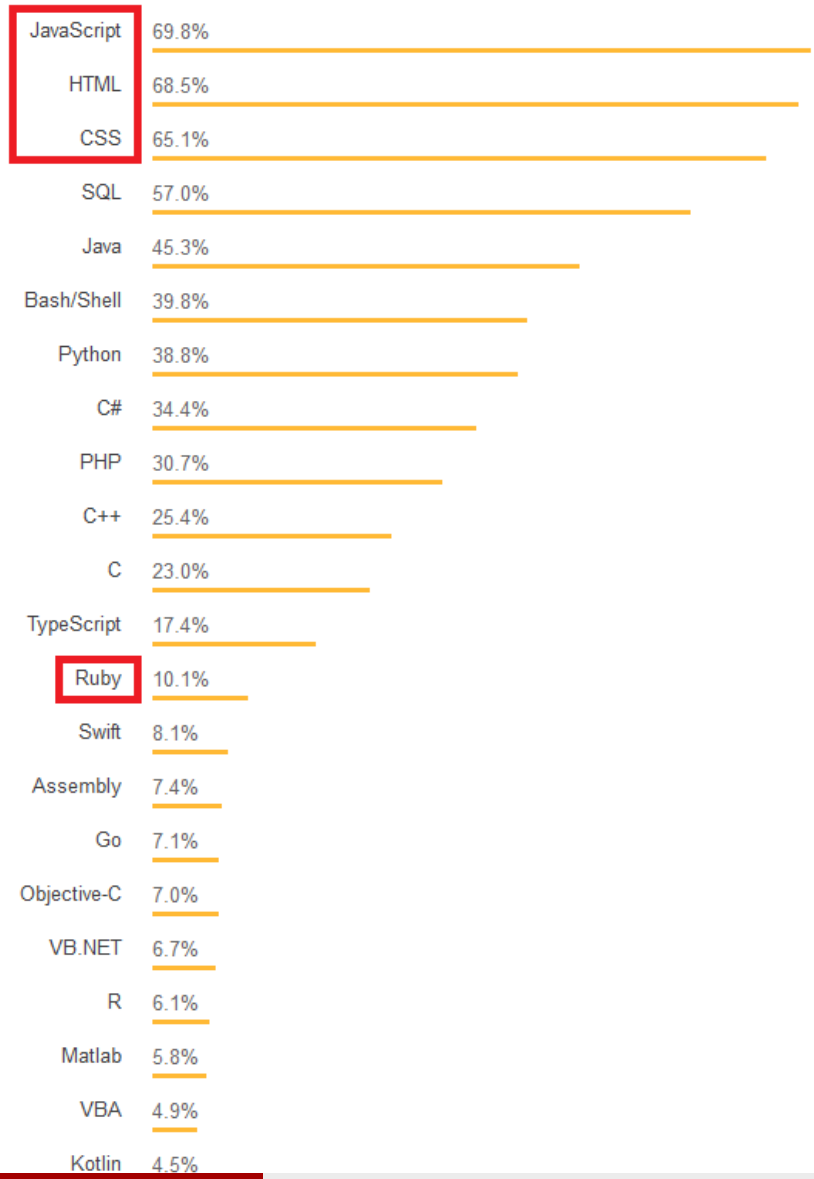
Рефлексивный язык

1. Поиск и модификация конструкций исходного кода во время выполнения
2. Изменение имён классов и функций во время выполнения
3. Анализ и выполнение строк кода, поступающих извне
4. Создание интерпретатора байткода нового языка

Объектно-ориентированный язык

1. Наследование
2. Инкапсуляция
3. Полиморфизм

Актуальность



Актуальность

Актуальность

Ruby on Rails
David Heinemeier Hansson

Optimize for programmer happiness

Python предлагает один наиболее эффективный способ - Ruby предлагает выбор Java защищает программиста от себя самого - Ruby предоставляет возможности делать что угодно.

```
1
2  irb
3  irb(main):001:0> exit
4  irb
5  irb(main):001:0> quit
6
7  python
8  >>> exit
9  Use exit() or Ctrl-D (i.e. EOF) to exit
```

Листинг 1: The Principle of Least Surprise

Convention over Configuration

Принятие полезных соглашений позволяет не принимать лишние решения. Для экспертов это экономия времени, для новичков - снижение порога вхождения. Необязательно знать всего соглашения, чтобы пользоваться их результатами. Сложнее всего принять правильное решение об отклонении от соглашений.

The menu is omakase

CoC на более высоком уровне.

1. В числе безопасность
2. Мастерство группы растет при изучении общего набора инструментов
3. Замены возможны, но не обязательны

No one paradigm

Отдельные элементы могут выполнены с использованием разных парадигм

Exalt beautiful code

```ruby
class Project < ApplicationRecord
  belongs_to :account
  has_many :participants, class_name: 'Person'
  validates_presence_of :name
end
```

Листинг 2: DSL или Ruby

Provide sharp knives

У программиста должны эффективные инструменты, даже если они потенциально опасны. Например, Monkey Patching.

2.days.ago - красивая конструкция?

Value integrated systems

Монолитная система проще в проектировании и разработке, чем система сервисов или микросервисов. Разработчики стремятся разделить свои системы до того, как это станет необходимо. Эффективный подход заключается в использовании общей базы кода в различных частях приложения.

Progress over stability

Rails живет очень долго и не собирается исчезать благодаря эволюции. В Rails Core нет пожизненных членств.

Push up a big tent

Привлечение новых людей в сообщество разработчиков, включая тех, чьи идеи отличаются от принятых большинством. Вы не обязаны отказываться от своего подхода, но должны прилагать усилия для формирования новых подходов. Несмотря на критику, идея может превратиться в новую парадигму.

Типы данных

Все сущности в Ruby являются объектами классов. Для определения класса используется метод .class Существует 8 базовых типов данных и 3 дополнительных для представления чисел.

```ruby
puts 'text!'.class # String
puts 1.class # Integer
puts 1.class.superclass # Numeric
puts 1.class.superclass.superclass # Object
puts 4.3.class # Float
puts 4.3.class.superclass # Numeric
puts nil.class # NilClass
h = { x: 1, 'really?' => true }
puts h.class # Hash
puts :symbol.class
puts [].class # Array
puts (1..8).class # Range
```

Константы

Имена констант начинаются с большой буквы.

```
1 MyConstant = 1
2 MY_CONSTANT = 2
3 MY_CONSTANT = 3
```

Листинг 4: Константы

Можно изменять значения констант, интерпретатор реагирует на это предупреждением.

```
1 (irb):5: warning: already initialized constant
    MY_CONSTANT
2 (irb):4: warning: previous definition of
    MY_CONSTANT was here
```

Переменные

Существует несколько типов переменных

```
1 $global_variable
2 @instance_variable
3 @@class_variable
4 local_variable
5 ClassName
```

Листинг 5: Переменные

Символы

Символы используются для записи неизменяемых строк и имеют только одну копию объекта

```
1 p "constant_string".object_id
2 p "constant_string".object_id
3 p :constant_string.object_id
4 p :constant_string.object_id
```

Листинг 6: Символы

Строки

Строки можно описывать в двойных или одинарных кавычках. Если строка содержит только текст, желательно использовать одинарные кавычки. Если в строке должны вычисляться выражение, то можно использовать шаблонные строки в двойных кавычках

```
1 "some text" # Bad styling
2 'also some text' # Ok
3 "easy as #{2+2}" # OK
```

Листинг 7: Строки

Операции со строками ч.1

```
1 'some text' + 'another text'
2 'repeat this' * 5
3 'Super Mario'.gsub('Mario', 'Luigi')
4 'Super Mario'.gsub 'Mario', 'Luigi'
5 '1'.to_i
6 '1'.to_f
7 'abcdef' <=> 'abfcd' # <, <=, >, >=, and between?
```

Листинг 8: Операции со строками 1

Проверка на равенство

Существует несколько операций сравнения:

1. == generic "equality"
2. === case equality
3. .eql? Hash equality
4. .equal? identity comparison

```
1 'str' == 'str'
2 'str' === 'str'
3 'str'.eql? 'str'
4 'str'.equal? 'str'
```

Листинг 9: Проверка на равенство

Операции со строками ч.2

```ruby
a = 'Hello world'
p a.length          #=> 11
p a[1]              #=> "e"
p a[2, 3]           #=> "llo"
p a[2..3]           #=> "ll"
p a[-3, 2]          #=> "rl"
p a[7..-2]          #=> "orl"
p a[-4..-2]         #=> "orl"
p a[-2..-4]         #=> ""
p a[11, 0]          #=> ""
p a[12, 0]          #=> nil
```

Листинг 10: Операции со строками 2

Операции со строками ч.3

```ruby
'hello'.capitalize  #=> 'Hello'
'hEllO'.downcase #=> 'hello'
'hEllO'.downcase! #=> 'hello'
"hEllO".swapcase!
'abcdef'.casecmp('abcde')  #=> 1
' '.empty?      #=> false
''.empty?       #=> true
```

Листинг 11: Операции со строками 3

Операции со строками ч.4

```ruby
"hello".include? "lo" #=> true
"hello".include? ?h   #=> true
"hello".index('e')          #=> 1
"hello".index('a')          #=> nil
"hello".index(?e)           #=> 1
"abcd".insert(3, 'X')  #=> "abcXd"
"abcd".insert(4, 'X')  #=> "abcdX"
"abcd".insert(-3, 'X') #=> "abXcd"
```

Листинг 12: Операции со строками 4

Документация