RUBY.

Специален език за хората, които
не могат да казват “P”

Юлиан Кунторов
@jkuntorov
I ревизия - октомври 2012г.

МАЛКО ПРЕДИСТОРИЯ

- Ruby е създаден от Юкихиро "Matz" Мацумото в средата на 90-те години, с цел да увеличи продуктивността на програмистите, запазвайки простотата на езика.
- В книгата Programming Ruby 1.9 на Pragmatic Programmers от април 2009г., Ruby е окачествен по следния начин:

  ★ **Ruby stays out of your way. You can concentrate on solving the problem at hand, instead of struggling with compiler and language issues.**

- Текущата стабилна версия на езика е 1.9.2

HELLO, WORLD

```
# hello.rb
1 def hello_world (name)
2     “Hello, #{name}”
3 end
4
5 puts hello_world “John”
6 puts hello_world “Jane”

ruby hello.rb
Hello, John
Hello, Jane
```

RUBY Е ОБЕКТНО-ОРИЕНТИРАН

- Всичко, с което работите в Ruby е обект, включително и резултата, който получавате.
- Обектите се създават чрез конструктори - извикване на метода `new` на даден клас.

```
1 first_product = Product.new
2 first_product.quantity = 2
3 first_product.description.length
4 first_product.remove_from_stock!
5 first_product.in_stock?
```

RUBY СПОДЕЛЯ

- В някои обектно-ориентирани езици (като C++) се поддържа наследяването на повече от един обект. Други езици, като Java и C#, поддържат наследяване само от един единствен обект.
- Но: една топка едновременно е **подскачащо** и **сферично** тяло, например.
- Ruby предлага интересен компромис: наследяването може да стане само от един клас-родител, но този клас може да бъде миксиран с множество т.нар. **mixins**. Те приличат на частична дефиниция на клас:

```
1 module Trig
2   PI = 3.141592654
3   def Trig.sin(x)
4     #
5   end
6   def Trig.cos(x)
7     #
8   end
9 end
```

```
1 module Moral
2   VERY_BAD = 0
3   BAD = 1
4   VERY_GOOD = 10
5
6   def Moral.sin(badness)
7     #
8   end
9 end
```

```
1 require ‘trig’
2 require ‘moral’
3 y = Trig.sin(Trig::PI/4)
4 wrongdoing =
Moral.sin(Moral::VERY_BAD)
```

ПРОМЕНЛИВИ

- В Ruby има 5 вида променливи: локални и глобални, instance и class и константи. Всеки от тях се изписва по различен начин, спрямо конвенцията:

```
1 local_variables, anObject, one_two
2 @instance_variables, @product
3 @@class_variables, @@_, @@Size
4 CONSTANTS, ALMOST_PI, PHI
5 $global_variables, $params, $flash
```

МЕТОДИ. МНОГО МЕТОДИ.

- Дефиницията на метод започва с ключовата дума `def` и завършва с `end`.
- Имената на методите могат да включват всякакви символи. Дори в конвенцията е препоръчително методи, които връщат булева стойност да завършват на `?`, а такива, които променят текущата инстанция на обекта, върху който са извикани, завършват на `!`.
- Към вградената в езика библиотека, към основните типове има хиляди готови методи. Само за типът `String` те са над 110.
- За reference можете да се обърнете към http://ruby-doc.org/

КОД НА БЛОКЧЕТА

В Ruby властва концепцията за предаването на блокове код, което е изключително удобно. Те се разграничават с служебните думи `do` и `end`, ако блокът е на няколко реда, или пък с къдрави скоби `{ }`, ако блокчето се събира на един ред.

```
1 array = [10,20,30,40,50]
2 array.each do |number|
3   puts number
4 end
5
6 1000.times { puts “I’m an evil genius!” }
```

ДА БЪДЕ ИЛИ ДА НЕ БЪДЕ?

- Освен стандартните if и else оператори, в Ruby се представя `unless`.
- Операторът `else if` в Ruby се казва `elsif`.
- В Ruby съществува концепцията за манипулатори на изразите:

```
1 a = 5
2 puts “a is negative” if a < 0
3 puts “a is positive” unless a < 0
4
5 puts “Hello” if Person.is_welcome?
```

7 ОСНОВНИ ТИПА ДАННИ

Numb3r	**Ran..ge**	
“String”	**:symbol**	**/Regex/**
Array[]	**{Ha:sh}**	

NUMBER

```
# numbers.rb
1 3.times { print "X " }
2 1.upto(5) {|i| print i, " " }
3 99.downto(95) do |i|
4   print i, " "
5 end
6 50.step(80, 5) do |i|
7   print i, " "
8 end

ruby numbers.rb
X X X 1 2 3 4 5 99 98 97 96 95 50 55
60 65 70 75 80
```

RANGE

```
# ranges.rb
1  # as intervals
2  (1..10) === 5  # true
3  (1..10) === 15 # false
4
5  # as sequences
6  (1..10).to_a
7  # => [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
8
9  # as conditions
10 while line = gets
11   puts line if line =~ /start/..
                  line =~ /end/
12 end
```

STRING

```ruby
# strings.rb
"Sec/day: #{24*60*60}" # => Sec/day: 86400
"#{'Ho! '*3}Hi!" # => Ho! Ho! Ho! Hi!
"This is line #$." # => This is line 3

# encoding: utf-8
def Σ(*args) # unknown number of args
  args.inject(:+)
end
puts Σ 1, 3, 5, 9
# => 18

length = 4:15
mins, secs = length.split(/:/)
# => mins = 4, secs = 15
```

SYMBOL

- Символите са просто константни имена, които не се налага да бъдат предекларирани и се гарантира, че са уникални.
- Няма нужда да присвояваме стойност на символите - езикът се грижи.
- Най-често се използват като ключове в асоциативни масиви.

```
1 # the old way
2 NORTH=1; SOUTH=2; WEST=3; EAST=4
3 walk(NORTH); look(EAST)
4
5 # the symbol way
6 :north; :south; :west; :east
7 walk(:north); look(:east)
```

REGULAR EXPRESSION

```
#  regex.rb
1  /\d\d:\d\d:\d\d/
2  # a time such as 12:34:56
3  /Ruby (Perl|Python)/
4  # Ruby, space, and either Perl or Python
5
6  if line =~ /Perl|Python/
7   puts "Scripting language mentioned:
          #{line}"
8  end
9
10 # replace first 'Perl' with 'Ruby'
11 line.sub(/Perl/, 'Ruby')
```

ARRAY

```
# arrays.rb
1 # array with three elements
2 a = [ 1, 'cat', 3.14 ]
3 puts "The first element is #{a[0]}"
4 puts "The last element is #{a[-1]}"
5
6 # set the third element
7 a[2] = nil
8 puts "The array is now #{a.inspect}"

ruby arrays.rb
The first element is 1
The last element is 3.14
The array is now [1, "cat", nil]
```

HASH

```
#  hashes.rb
1  # Ruby > 1.9
2  instruments = {
3    cello:    'string',
4    clarinet: 'woodwind'
5  }
6
7  instruments[:cello] # => 'string'
8  instruments['cello'] # => nil
9
10 # Ruby < 1.9
11 instruments = {
12   :cello:   => 'string',
13   :clarinet => 'woodwind'
14 }
```

RUBY ПРОВОКИРА TDD

- През годините са били създадени много библиотеки за unit-testing. Ruby идва с вградена библиотека - при версия 1.8 , това е Test::Unit, а при 1.9 - MiniTest на Раян Дейвис.

- Това означава, че можете паралелно да развивате и кода и тестовете, без да се налага да инсталирате външна библиотека.

```
# roman_test.rb
1 require 'roman'
2 require 'test/unit'
3 class TestRoman < MiniTest::Unit::TestCase
4   def test_simple
5     assert_equal("i", Roman.new(1).to_s)
6     assert_equal("ix", Roman.new(9).to_s)
7   end
8 end
```

БИЖУТАТА НА RUBY

- С развитието на езика, на програмистите им се налага да обменят все повече код.

- Така се появяват т.нар. Ruby Gems, което е удобна единица за разпространение на код, която поддържа версии и е силно интегрирана в Ruby.

- Ако ви трябва нещо, на адрес www.rubygems.org има огромна вероятност, че ще го откриете, из близо 46069-те джемове (към днешна дата).

RUBY ON RAILS

- Ruby on Rails е framework за web приложения, базирана на MVC архитектурата (Model View Controller), предоставяща RESTful web услуги.
- Rails има два основни принципа:
  - **DRY (Don’t Repeat Yourself)**
  - **Convention over configuration**
- Rails е agile:
  - Хората и взаимоотношенията са пред процесите и инструментите.
  - Работещият софтуер е пред пълноценната документация.
  - Работата в екип с клиента е пред преговори по договора.
  - Бързото отразяване на промените е пред следването на план.

ДОСАДНА ДОКУМЕНТАЦИЯ?

- От версия 1.8 насам, Ruby идва с вграден RDoc, което всъщност е генератор на документация.
- RDoc прави две неща:
  - Анализира Ruby и C файлове и изважда необходимата информация
  - С нея създава нещо, което може да бъде лесно прочетено, например HTML.

```
# rdoc_example.rb
1 =begin rdoc
2 Calculate the minimal-cost path though the graph.
3 =end
4 def calculate_path
5   ...
6 end
```

БЛАГИНКИ

Програма за размяна стойността на две променливи

```c
# C
1 int a = 1;
2 int b = 2;
3 int temp;
4 temp = a;
5 a = b;
6 b = temp;
```

```ruby
# Ruby
1 a = 1
2 b = 2
3 a, b = b, a
```

И ОЩЕ НЕЩО...

- операторите `case` ... `when` ...
- цикъл `for` ... `in`, който не се използва и цикъл `until`
- операторите `break`, `redo`, `next`
- Fibers (влакна, като нишки), `lambda`
- Rake - вариантът на make за Ruby
- RSpec и Cucumber - тестове като разговорната реч
- Capybara - помага ви за тестовете на web приложения, като симулира действията на реален потребител

БИБЛИОГРАФИЯ

В тази презентация е използвана информация от следните източници:

- Programming Ruby 1.9 (The Pragmatic programmer's guide) на Dave Thomas - http://pragprog.com/book/ruby3/programming-ruby-1-9

- Agile Web Development with Rails (4th edition) на Sam Ruby, Dave Thomas и David Hansson - http://pragprog.com/book/rails4/agile-web-development-with-rails

- Ruby Docs - http://ruby-doc.org/

- Ruby в Wikipedia - http://en.wikipedia.org/wiki/Ruby_(programming_language)