The Art of Coding
A slideshow of some beautiful features of *Ruby*

Benoit Daloze

26 octobre 2010

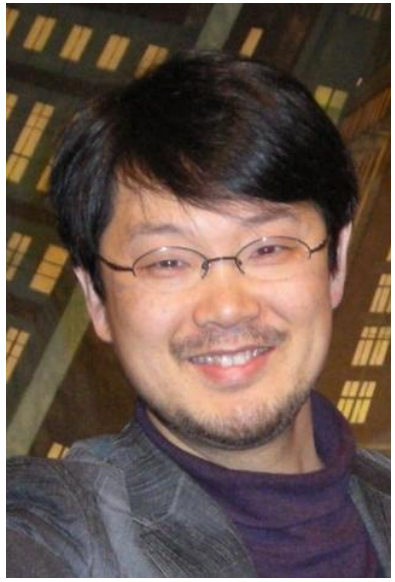
Who am I ?

- Benoit Daloze
- github.com/eregon (@eregontp)
- Rubyist since 2006
- I learned C, C++, Java, PHP, Oz, Haskell, Python and Ruby
- co-author of the symbolic gem (symbolic math)
- won a few Ruby challenges
  (Broadsides, Interactive Fiction, Game Of Life)

What is Ruby ?

Ruby is ... “a dynamic, open source programming language with a focus on simplicity and productivity. It has an elegant syntax that is natural to read and easy to write.”

History



- Ruby was created on February 24, 1993 by Yukihiro Matsumoto who wished to create a new language that balanced *functional* programming with *imperative* programming.
- "I wanted a scripting language that was *more powerful than Perl*, and *more object-oriented than Python*."
- Ruby 1.0 was released on December 25, 1996
- Current Ruby version is 1.9.2 (released on September 18, 2010)

Who use Ruby ?[1]

- NASA Langley Research Center uses Ruby to conduct simulations
- Google SketchUp is a 3D modeling application that uses Ruby for its macro scripting API
- Ruby On Rails is one of the best and most innovative web application framework (GitHub, Twitter, Basecamp, Scribd, Geni, Redmine, Lighthouse, Urban Dictionary, White Pages, Next Sprocket, spitzer.caltech.edu (NASA), Go vs Go, ...)
- Ruby was used to write the central data collection portion of Level 3 Communications Unix Capacity and Planning system that gathers performance statistics from over 1700 Unix servers scattered around the globe

1. http ://www.ruby-lang.org/en/documentation/success-stories/

Interpreters

- MRI : Matz's Ruby Interpreter : C
- JRuby : Java
- Rubinius : Ruby on top of LLVM with C++
- MacRuby : Objective-C, bridge with Cocoa
- IronRuby : Open Source implementation for .NET

Main Principles

- Beautiful and natural syntax, no useless () {} ; ...
- Everything is an Object : Number, Class, Method, Binding, nil ...
- Dynamic typing and Duck typing
- Succinct and flexible syntax
- Reflection and metaprogramming
- Functional programming with Blocks/closures

Syntax

```ruby
array = [1, [2.0, 'c'], :d, (5..67)]
obj.method(*params) { block }
# or obj.m(*params) do block end

class MyClass
  def my_method(*args)
    body
    this_last_statement_is_the_returned_value
  end

  def clever_params(a, b = a *2, c = @ivar**2)
  end
end

you, (can, splat), *an = array
```

Simplicity

```
vowels = %w[a e i o u]

alphabet = ('a'..'z').to_a

consonants = alphabet - vowels
```

Expressiveness

```ruby
5.times { puts 'Hello Ruby' }

"This is Ruby".length # => 12

3.even? # => false

[1,2,3].include? 2 # => true

ary.shuffle! until ary.sorted? # Bogosort
```

Productivity

```ruby
class Person
 # def name
 #   @name
 # end
 # def name= name
 #   @name = name
 # end
 # ...
 attr_accessor :name, :age
end

john = Person.new
john.name = 'John'
john.age = 25
# or
Person = Struct.new(:name, :age)
```

Flexibility

Core classes can be modified, even Fixnum#+

```
class Array
  def shuffle
    sort_by { rand }
  end
end

[1, 2, 3].shuffle # => [2, 3, 1], ...
```

Dynamism

```
Padawan = Class.new
class Jedi
  def train(padawan)
    def padawan.control_the_force
      puts "Now i'm ready to become a Jedi!"
    end
  end
end

Skywalker, Yoda = Padawan.new, Jedi.new
Skywalker.respond_to? :control_the_force # => false

Yoda.train Skywalker
Skywalker.respond_to? :control_the_force # => true
Skywalker.control_the_force
# => Now i'm ready to become a Jedi!
```

Conciseness

```
require 'sinatra'

get '/' do
  'Hello, World!'
end

$ curl localhost:4567 # => Hello, World!
```

Beautiful API

```ruby
open('ruby-conf.rb') { |f| f.write 'New Orleans' }

# URI work too !
require 'open-uri'

puts open 'http://www.ruby-lang.org/en/', &:read
```

Simplify loops

```ruby
def factorial n
  fact = 1
  for i in (1..n)
    fact *= i
  end
  return fact
end
factorial(5) # => 120

# becomes
class Integer
  def !
    (1..self).inject(1) { |fact, i| fact * i }
  end
end
5.! # => 120
# can even be: (1..self).inject(1, :*)
```

Chaining

```
[450, 1000, 675].sort.take(2).map { |p| "$#{p}" }
# => ["$450", "$675"]

# Find the char with the most occurrences in a
    String
adn = "ATTGCCATATCC".chars.to_a
adn.uniq.sort.max_by { |c| adn.count(c) } # => "C"
```

Enumerators

```ruby
# has a gap after 4 and 7!
numbers = [1, 2, 3, 4, 6, 7, 9, 10]

gaps = []
numbers.each_cons(2) do |x, y|
  gaps << x unless y == x + 1
end
gaps #=> [4, 7]

# or

numbers.each_cons(2)
  .reject { |x,y| x + 1 == y }.map(&:first)
```

Caching

```ruby
# Caching is easy: we have the ||= operator
# which assign only if it is nil or false

def my_method
  @cache ||= some_extensive_computation
end
```

Closures

```
def create_get_and_set closure_value
  return lambda { closure_value },
      lambda { |x| closure_value = x }
end

getter, setter = create_get_and_set
setter.call(42)
getter.call # => 42
```

Operators

Name it as it is, operators are just methods :

```
1 + 2 == 1.+(2)
```

```
class Number
  # Same for + / % ** & ^ | << [] []= ~ < ...
  def * n
    @value * n
  end
end
```

Literals

Most useful objects are literals

```
[1, 2_345_678, 0xfe, 0b101010]
3.14
"string"
/RegExp/
:symbol
%w[array of strings]
range = (1..15)
hash = {key: value}

<<HERE
heredoc
HERE
```

String interpolation

```
"This is #{'*so* ' if $VERBOSE}useful !"

"There is real #{sleep 1}code"

"I love #{"nest#{'ing'}"}"

"It convert to #{String} by calling #to_s"
```

Rich stdlib

All core Classes have a lot of handy methods

```
Array.instance_methods # => ..., pop, push, shift,
    unshift, take, drop, insert, replace, [], []=,
    rotate, ...
String.instance_methods # => bytes, capitalize,
   center, chars, chomp, chop, codepoints,
   downcase, lines, end_with?, (g)sub(!), encode
   ...
Enumerable.instance_methods # => count, grep, min,
    min_by, sort_by, each_with_index, map,
   each_cons, each_slice, zip, take, ...
```

Scripting

```
# set_mtime_from_exiftime

#!/usr/bin/env ruby
require 'time'
Dir["**/*.{jpg,JPG}"].each do |f|
  times = `exiftime #{f}`
  .scan(/(?:\d+:\d+:\d+ ?){2}/)
  .map { |time| Time.new(*time.split(/\W/) }

  raise "Not same times: #{times}" unless times.
      all? { |time| time == times.first }
  File.utime(File.atime(f), times.first, f)
end
```

RSpec

```
describe BlogPost do
  subject { BlogPost.new 'foo', 'bar' }
  it { should be_invalid }
end

describe Array do
  its(:length) { should == 0 }
end

expect do
  foo.bar
end.to change { baz.quux }.by(1)

mock.should_receive(:method).once.with(args).
    and_return(answer)
```

A quote of Matz

"[...] The computers don't care. We humans care about the effort we pay. Often people, especially computer engineers, focus on the machines. They think, 'By doing this, the machine will run faster. 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."[2]

2. http ://www.artima.com/intv/ruby4.html

Credits

- http ://ruby-lang.org
- Wikipedia : Ruby, Yukihiro Matsumoto
- ruby-talk : “The beauty of Ruby through examples”
- Pure RSpec : http ://pure-rspec-scotruby.heroku.com
- RSpec : http ://rspec.info