Strolling through Ruby 2.0

Amir Friedman
@newartriot

A bit of history...

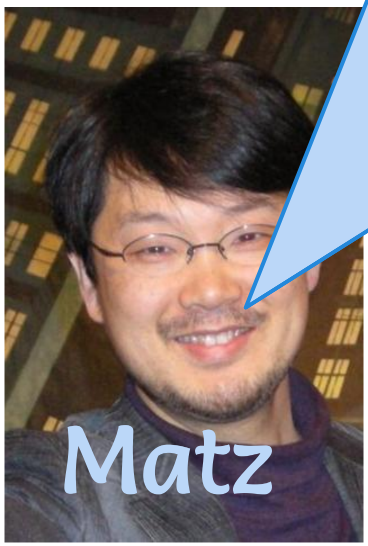
1993/02 - Ruby is born
1995/12 - 0.95, 1st public release
1996/12 - 1.0
2000/09 - 1.6, symbols
2003/08 - 1.8, rails
2007/12 - 1.9.0, still a development release
2010/08 - 1.9.2, production release
2011/10 - 1.9.3
2013/02 - 2.0, happy birthday!

Ruby 2.0 goals

- Compatibility
- Usability
- Performance

A huge leap?

The version number goes up to 2.0 but the changes are rather small. Smaller than the ones we made in 1.9.



Features

- Keyword arguments
- Enumerator#lazy
- Module#prepend
- Refinements
- Symbol Array: %i and %I
- Regex engine changed to Onigmo (/\R/)
- ...and *more*

Keyword Arguments

The old way:

```
def something(foo, bar, baz, options = {})
  puts foo, bar, baz
  puts options
end


pry(main)> something 1, 2, 3, hello: "world"
1
2
3
{:hello=>"world"}
=> nil
```

handling defaults:

```
def something(options = { name: "Anon", address: "NA" })
  options
end
```

```
pry(main)> something name: "Herbert"
=> {:name=>"Herbert"} # => Oops.
```

```
def something(options = {})
  default_options = { name: "Anon", address: "NA" }
  options = default_options.merge(options)
end
=> {:name=>"herbert", :address=>"NA"}
```

Keyword Arguments ~ Ruby 2.0 style

This:

```ruby
def my_details(options = {})
  default_options = { name: "Anon", address: "NA" }
  options = default_options.merge(options)
  puts options[:name], options[:address]
end
```

Turns to *this:*

```ruby
def my_details(name: "Anon", address: "NA")
  puts name, address
end
```

Splat operator

```
def my_details(name, *rest)
  [name, *rest]
end

a_method("Amir", "Ruby", "Underground")
=> ["Amir", "Ruby", "Underground"]
```

```
def my_details(name: "Anon", **address)
  address
end
something name: "Amir", city: "Tel Aviv"
=> {:city=>"Tel Aviv"}
```

Enumerator#lazy

Returns a lazy enumerator and enumerate values only on an as-needed basis.

If a given to #zip or #cycle, the values will be calculated immediately

```
e = (1..Float::INFINITY).select { |num| num % 5 == 0 }
=> [5,
 10,
 15,
 20,
 25,
 ...]
```

```
e = (1..Float::INFINITY).lazy.select { |num| num % 5 == 0 }
=> #<Enumerator::Lazy: ...>
e.next
=> 5
e.next
=> 10
```

Can be forced to finish enumeration:

```
e.force
=> # the rest.
```

Reads a line at a time:

```
File.open(filename).lazy.detect { |line| line =~ /login/ }
```

Module#prepend

Prepends a module to the ancestors chain

```
module Bar
  def foo ; puts "inside Bar" ; super ; end
end

class Foo
  prepend Bar
  def foo ; puts "inside Foo" ; end
end

Foo.ancestors
=> [Bar, Foo, Object, PP::ObjectMixin, Kernel, BasicObject]

my_obj = Foo.new
=> #<Foo:0x007faf1d1436a0>
my_obj.foo
inside Bar
inside Foo
=> nil
```

```ruby
module Bar
  def foo
    puts "inside foo"
    super
  end

  def self.prepended(klass)
    puts "Module prepended"
  end

  def self.included(klass)
    puts "Module included"
  end
end

class Foo
  prepend Bar
end

# prints "Module prepended"
```

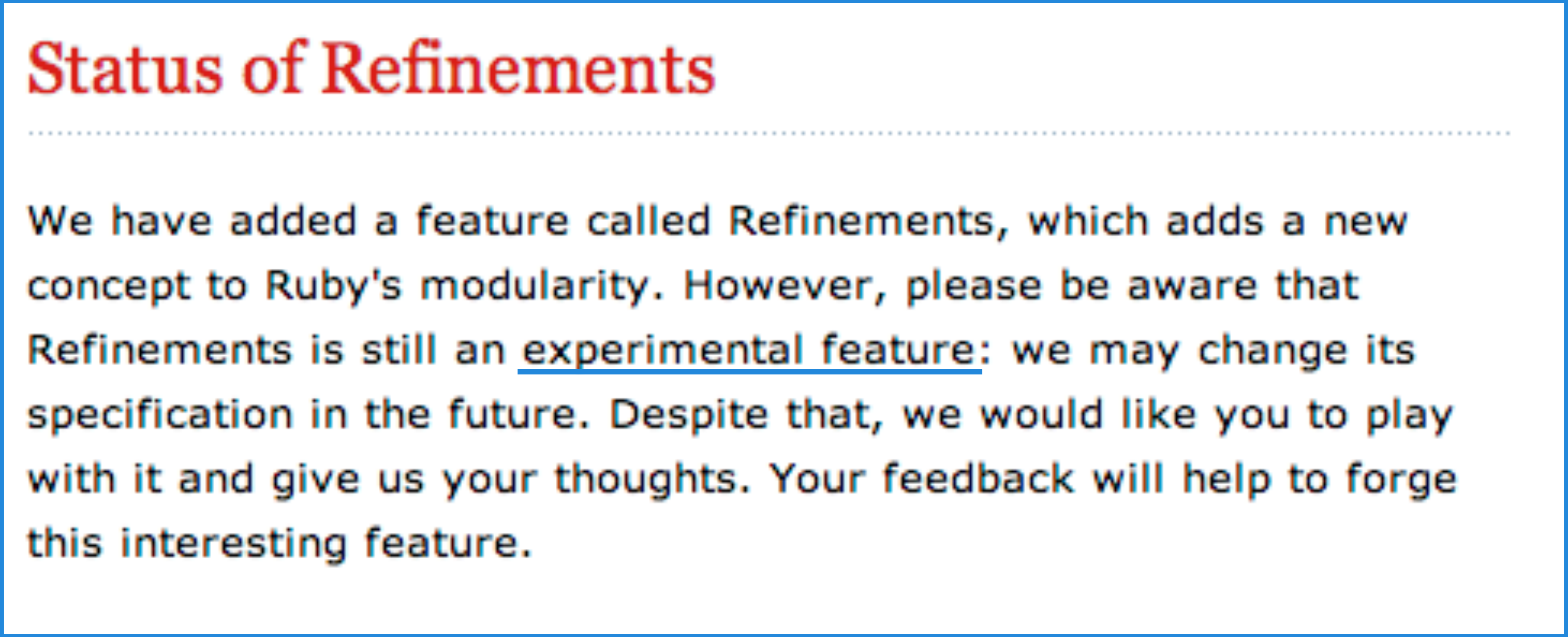
use prepend_features to prepend dynamically

```
module Bar
  def foo
    puts "inside Bar"
  end

  def self.prepend_to(klass)
    prepend_features klass
  end
end

class Foo
  def foo
    puts "inside Foo"
  end
end

my_obj = Foo.new
=> #<Foo:0x007fdb26e612d0>
my_obj.foo
inside Foo
=> nil
Bar.prepend_to Foo
=> Bar
my_obj.foo
inside Bar
```

Refinements

Refinements provide a way to extend classes locally.

See the refinements spec at:
http://bugs.ruby-lang.org/projects/ruby-trunk/wiki/RefinementsSpec



Status of Refinements

We have added a feature called Refinements, which adds a new concept to Ruby's modularity. However, please be aware that Refinements is still an experimental feature: we may change its specification in the future. Despite that, we would like you to play with it and give us your thoughts. Your feedback will help to forge this interesting feature.

Source: http://www.ruby-lang.org/en/news/2013/02/24/ruby-2-0-0-p0-is-released/

What it could have been

```ruby
module StringLength
  refine String do
    def long?
      self.length > 5 ? true : false
    end
  end
end
# warning: Refinements are experimental, and the behavior may change
in future versions of Ruby!

class StringStuff
  using StringLength
  def do_something(string)
    if string.long?
      puts "String too long"
    else
      puts "all good"
      string << "yippy"
    end
  end
end
```

How it really is...

```
module StringLength
  refine String do
    def long?
      self.length > 5 ? true : false
    end
  end
end

using StringLength
=> main

class StringStuff
  ~~# using StringLength~~
  def do_something(string)
    if string.long?
      puts "String too long"
    else
      puts "all good"
      string << "yippy"
    end
  end
end
```

%i{ symbol array }

```
=> [:symbol, :array]
```

Regex Engine is now Onigmo

Conditional, Keep (\K), newlines (\R), Further reading:

- http://perldoc.perl.org/perlre.html
- https://github.com/k-takata/Onigmo

Misc. Stuff

#to_h

converting convention to Hash: #to_h

- ENV.to_h, nil.to_h, etc.

```
ENV.class # => Object
ENV.to_h.class # => Hash
nil.to_h # => {}
```

- Kernel#Hash() function

```
Hash(arg) # => calls arg.to_h
```

- Struct supports to_h

UTF-8 is the default encoding!

```
# encoding: utf-8
# not necessary anymore
```

__dir__

Returns the source file's directory

```
File.dirname(File.realpath(__FILE__)) == __dir__
```

IO Deprecations

IO#lines, #bytes, #chars and #codepoints are deprecated

```
File.open('/Users/amirf/projects/ruby2/Bar.rb').lines
(pry):15: warning: IO#lines is deprecated; use #each_line instead
```

String#chars, String#lines return Array

Method Transplanting

Module#define_method now accepts an UnboundMethod from a Module

```ruby
module MyModule
  def pick_me
    "thanks"
  end
end
```

```ruby
define_method :pick_me, MyModule.instance_method(:pick_me)
```

```ruby
puts pick_me
```

Module#const_get

Can get nested objects

```
module ThisModule
  module IsVery
    module Deep; end;
  end
end
Object.const_get("ThisModule::IsVery::Deep")
=> ThisModule::IsVery::Deep
```

Range#size

Array#bsearch, Range#bsearch

Must be ordered.

find-minimum mode - The block needs to return true/false:
returns false for any element whose value is less than x
returns true for any element whose value is greater than or equal to x

```
[11, 23, 33, 55, 62, 70, 80, 100, 101].bsearch { |e| puts e ; e >= 70 }
62
100
80
70
=> 70
```

In find-any mode (this behaves like libc's bsearch(3)), the block must return a number, and there must be two values x and y

```
[11, 23, 33, 55, 62, 70, 80, 100, 101].bsearch { |e| 100 <=> e }
```

Signal.signame

For *NIX

```
[15] pry(main)> Signal.signame
(5)
=> "TRAP"
[16] pry(main)> Signal.signame
(9)
=> "KILL"
[17] pry(main)> Signal.signame
(1)
=> "HUP"
```

String#b

Returns a copied string whose encoding is ASCII-8BIT.

main.define_method

```ruby
define_method(:wilma) { puts "Charge it!" }
```

Object#remove_instance_variable

now public

```
define_method(:wilma) { puts "Charge it!" }
```

Array#values_at

now returns nil for each value out-of-range

```
[26] pry(main)> a = [1, 2]
=> [1, 2]
[27] pry(main)> a.values_at(0..6)
=> [1, 2, nil, nil, nil, nil, nil]
```

YAML now completely depends on libyaml being installed

Syck has been removed.

STDLib Changes

https://github.com/ruby/ruby/blob/trunk/doc/standard_library.rdoc

Thread#thread_variable_get
Thread#thread_variable_set
Thread#thread_variables
Thread#thread_variable?
for getting thread local variables

Mutex#owned?

THANKS