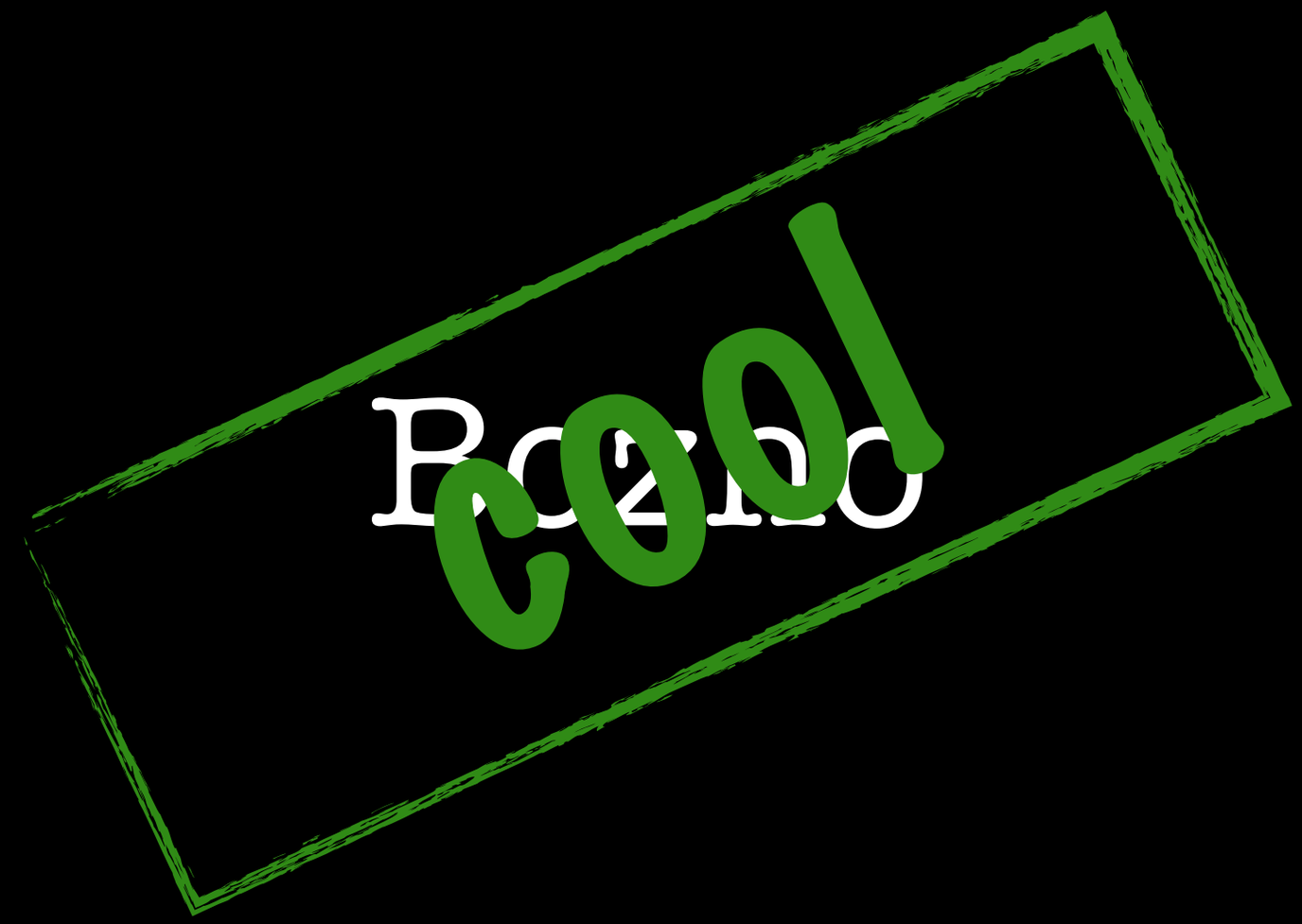
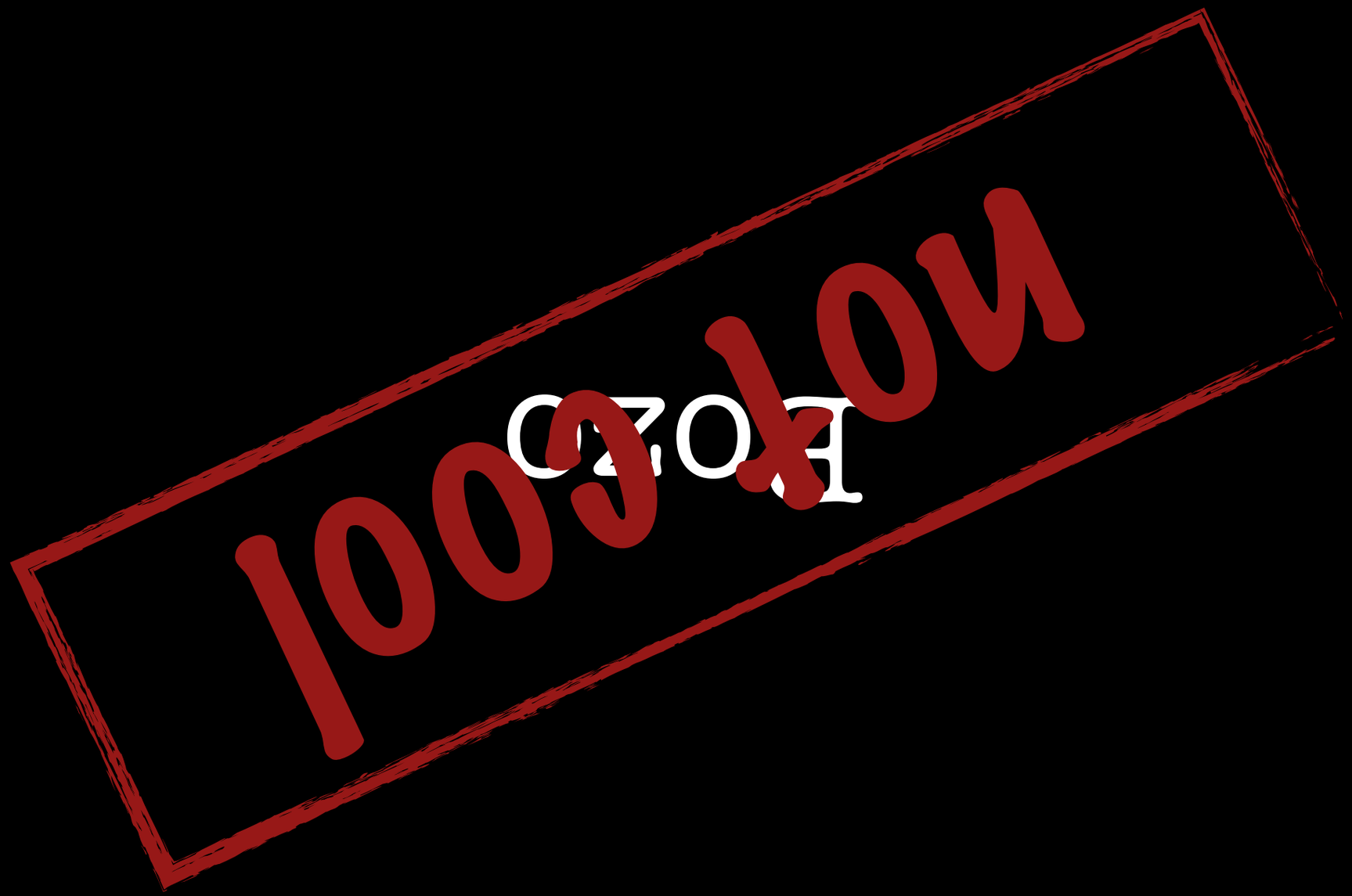
Καλησπέρα

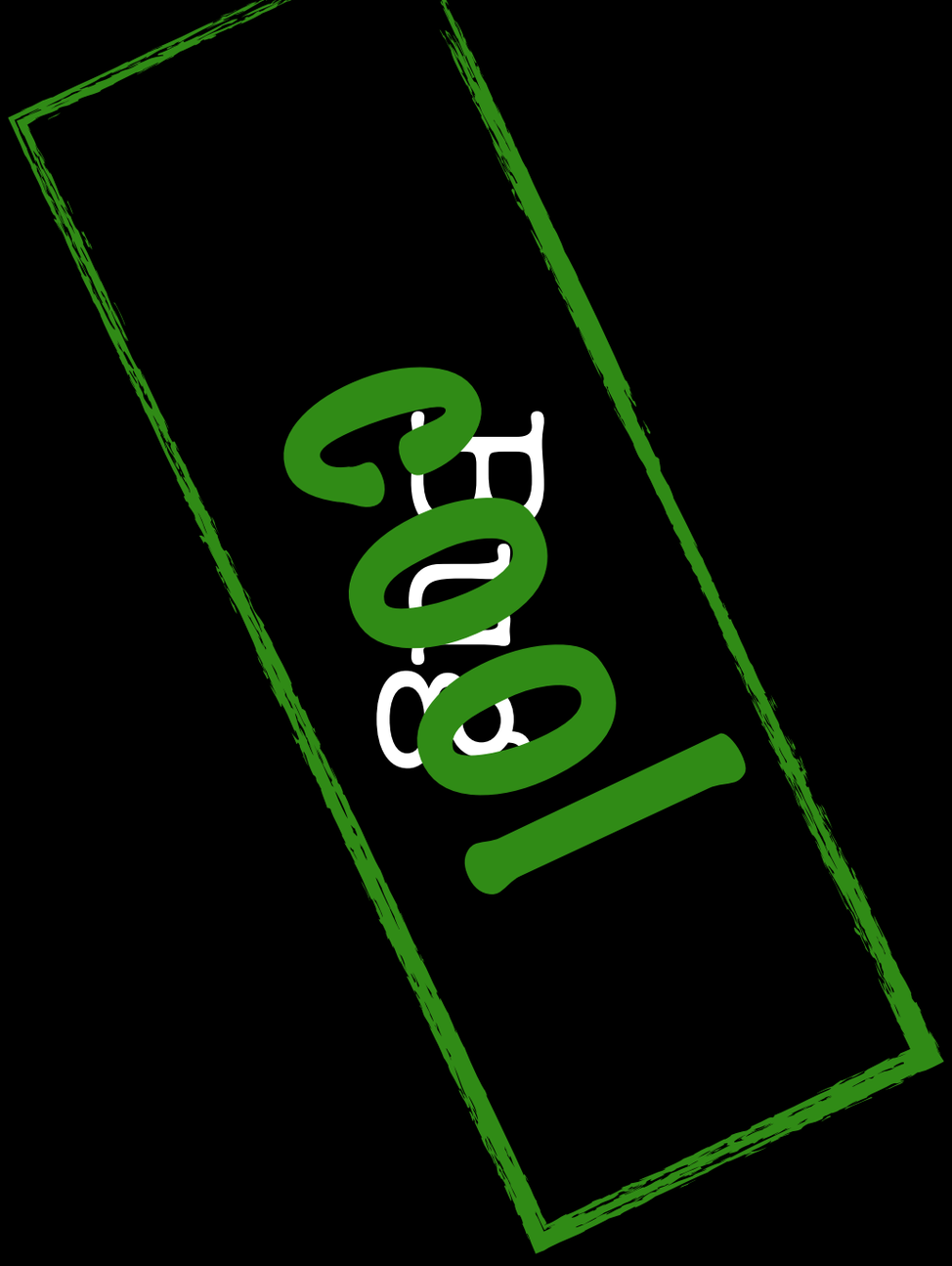
Божидар

ΔΙΑΤΗΡΕΙΤΕ ΤΗΝ
ΠΕΡΙΦΕΡΕΙΑ ΜΑΣ
ΚΑΘΑΡΗ
ΜΗ ΡΙΧΝΕΤΕ ΣΚΟΥΠΙΔΙΑ
ΣΤΟΥΣ ΔΡΟΜΟΥΣ
ΔΕΣΕ-ΠΕΡΙΦΕΡΕΙΑ ΔΥΤΙΚΗΣ ΕΛΛΑΔΟΣ

Bozhidar

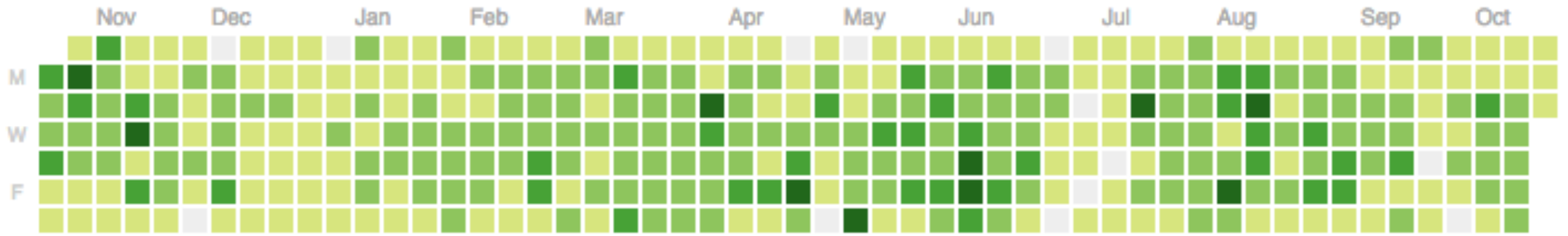

Good
cool


C2OH
100% Юол


Bio
Bio

bbatsov

Contributions



Summary of Pull Requests, issues opened, and commits. Learn more.

Less More

Year of contributions	Longest streak	Current streak
3,629 total	111 days	17 days
Oct 21, 2013 – Oct 21, 2014	December 30 – April 19	October 5 – October 21

Ruby & Rails style guides


RuboCop


toptal
Design. Build. Grow. Live.


toptal
VilniusRB


FRESH
CHIOS
BEER
ΖΥΘΟΠΟΙΙΑ ΧΙΟΥ
ΑΠΑΣΤΕΡΙΩΤΗ - ΑΦΙΑΤΡΑΡΙΣΤΗ


FRESH
CHIOS
BEER

15.

where is

(森) my mind?

RUBY 4.0:

TO INFINITY

AND BEYOUND

by Bozhidar Batsov

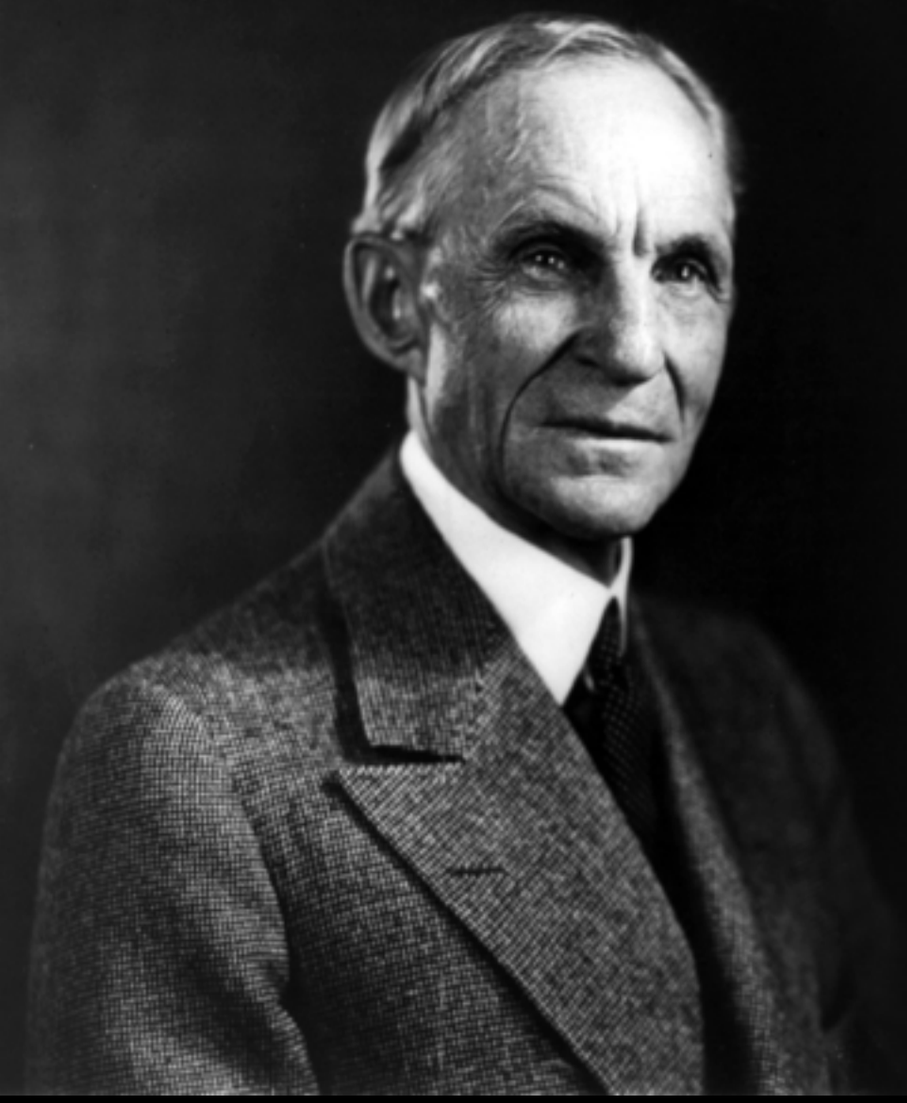
Not on Ruby's Core Team

"We'll aim to release Ruby 3 for the Olympic Games in Tokyo in 2020."

–Matz

Ruby is now mature

Build the things your users need, instead of the things they want.



"IF I HAD ASKED PEOPLE WHAT THEY WANTED, THEY WOULD HAVE SAID: **FASTER HORSES...**"

Henry Ford

The track record of recent Ruby innovation

Ruby != MRI

```ruby
3.times do
  puts "Ruby Rocks!"
end
```

Ruby 2.0

- keyword arguments
- %i
- UTF-8 is now the default source file encoding
- Refinements (experimental feature)

Ruby 2.1

- Rational/Complex Literal
- defs return value
- Refinements are no longer experimental feature

Ruby 2.2

Nada

Ruby 2.3

- frozen string literals pragma
- safe navigation operator (&.)

Ruby 2.4

- Unify Fixnum and Bignum into Integer
- Support Unicode case mappings

Java innovates more!

Ruby 2.2 includes many new features and improvements for the increasingly diverse and expanding demands for Ruby.

For example, Ruby's Garbage Collector is now able to collect Symbol type objects. This reduces memory usage of Symbols; because GC was previously unable to collect them before 2.2. Since Rails 5.0 will require Symbol GC, it will support only Ruby 2.2 or later. (See Rails 4.2 release post for details.)

Also, a reduced pause time thanks to the new Incremental Garbage Collector will be helpful for running Rails applications. Recent developments mentioned on the Rails blog suggest that Rails 5.0 will take advantage of Incremental GC as well as Symbol GC.

Ruby 2.2 includes many new features and improvements for the increasingly diverse and expanding demands for Ruby.

For example, Ruby's Garbage Collector is now able to collect Symbol type objects. This reduces memory usage of Symbols; because GC was previously unable to collect them before 2.2. Since Rails 5.0 will require Symbol GC, it will support only Ruby 2.2 or later. (See Rails 4.2 release post for details.)

Also, a reduced pause time thanks to the new Incremental Garbage Collector will be helpful for running Rails applications. Recent developments mentioned on the Rails blog suggest that Rails 5.0 will take advantage of Incremental GC as well as Symbol GC.

Ruby 2.2 includes many new features and improvements for the increasingly diverse and expanding demands for Ruby.

For example, Ruby's Garbage Collector is now able to collect Symbol type objects. This reduces memory usage of Symbols; because GC was previously unable to collect them before 2.2. Since Rails 5.0 will require Symbol GC, it will support only Ruby 2.2 or later. (See Rails 4.2 release post for details.)

Also, a reduced pause time thanks to the new Incremental Garbage Collector will be helpful for running Rails applications. Recent developments mentioned on the Rails blog suggest that Rails 5.0 will take advantage of Incremental GC as well as Symbol GC.

Ruby 2.2 includes many new features and improvements for the increasingly diverse and expanding demands for Ruby.

For example, Ruby's Garbage Collector is now able to collect Symbol type objects. This reduces memory usage of Symbols; because GC was previously unable to collect them before 2.2. Since Rails 5.0 will require Symbol GC, it will support only Ruby 2.2 or later. (See Rails 4.2 release post for details.)

Also, a reduced pause time thanks to the new Incremental Garbage Collector will be helpful for running Rails applications. Recent developments mentioned on the Rails blog suggest that Rails 5.0 will take advantage of Incremental GC as well as Symbol GC.

Ruby 2.2 includes many new features and improvements for the increasingly diverse and expanding demands for Ruby.

For example, Ruby's Garbage Collector is now able to collect Symbol type objects. This reduces memory usage of Symbols; because GC was previously unable to collect them before 2.2. Since Rails 5.0 will require Symbol GC, it will support only Ruby 2.2 or later. (See Rails 4.2 release post for details.)

Also, a reduced pause time thanks to the new Incremental Garbage Collector will be helpful for running Rails applications. Recent developments mentioned on the Rails blog suggest that Rails 5.0 will take advantage of Incremental GC as well as Symbol GC.

Ruby 2.2 includes many new features and improvements for the increasingly diverse and expanding demands for Ruby.

For example, Ruby's Garbage Collector is now able to collect Symbol type objects. This reduces memory usage of Symbols; because GC was previously unable to collect them before 2.2. Since Rails 5.0 will require Symbol GC, it will support only Ruby 2.2 or later. (See Rails 4.2 release post for details.)

Also, a reduced pause time thanks to the new Incremental Garbage Collector will be helpful for running Rails applications. Recent developments mentioned on the Rails blog suggest that Rails 5.0 will take advantage of Incremental GC as well as Symbol GC.


SAD HIPSTER
EVERYONE IS USING RUBY FOR WEB DEVELOPMENT.
quickmeme.com

What about Ruby 3.0?

Little is known about it...

Optional static typing?

Duck inference?

Better support for concurrent & parallel programming

3 times faster performance?

Getting rid of some quirky features?

We're not going to repeat the Python 3 mistakes!

— Matz

And what about the Perl 6 mistakes?

— Bozhidar

KEEP
CALM
AND
MAKE RUBY
GREAT AGAIN

Ruby 4.0

Codename Buzz

Ruby 4x4

Ruby 4 is going to be 4 times faster than Ruby 3

Ruby 4 is going to be
12 times faster than
Ruby 2

Ruby 4 is finally going to be fast enough!

Ruby 4.0,
the language

Ruby 4.0,
the language
(and maybe
the Standard Library)

Design principle #1

Continue to optimize for happiness

Add some useful new features

Immutable data structures

vector

```
v = @[1, 2, 3]
```

immutable hash

```
m = @{one: 1, two: 2}
```

immutable set

```
s = @${1, 2, 3}
```

```
s = ${1, 2, 3}
```

Static typing and runtime contracts

Inspired by RDL

```
type '(Fixnum, Fixnum) -> String'
def m(x, y) ... end
```

```
pre { |x| x > 0 }
post { |r,x| r > 0 }
def sqrt(x)
  # return the square root of x
end
```

```
type '(Float x {{ x>=0 }}) -> Float y
{{ y>=0 }}'
def sqrt(x)
    # return the square root of x
end
```

Better concurrency APIs

Inspired by concurrent-ruby

CSP

```ruby
messages = Concurrent::Channel.new

Concurrent::Channel.go do
  messages.put 'ping'
end

msg = messages.take
puts msg
```

```ruby
def sum(a, c)
  sum = a.reduce(0, &:+)
  c << sum # `<<` is an alias for `put` or `send`
end

a = [7, 2, 8, -9, 4, 0]
l = a.length / 2
c = Concurrent::Channel.new

Concurrent::Channel.go { sum(a[-l, l], c) }
Concurrent::Channel.go { sum(a[0, l], c) }
x, y = ~c, ~c # `~` is an alias for `take` or `receive`

puts [x, y, x+y].join(' ')
```

Design principle #2

Simplicity

Simplicity is the ultimate sophistication.

Less is more

Simplicity leads to happiness.

Let’s drop some stuff

Let’s drop some useless stuff

for loops

```
for name in names
  puts name
end
```

```
names.each do |name|
  puts name
end
```

BEGIN & END

```ruby
END {
  puts 'Bye!'
}
puts 'Processing...'

BEGIN {
  puts 'Starting...'
}
```

```
puts 'Bye!'
puts 'Starting...'

puts 'Processing...'
```

Kernel#at_exit,

anyone?

flip-flops

```ruby
DATA.each_line do |line|
  print(line) if (line =~ /begin/)..(line =~ /end/)
end
```

block comments

```
=begin
comment line
another comment line
=end
```

Must be placed at the very beginning of a line

```
class SomeClass
  =begin
  This is a top comment.
  Or is it?
  =end
  def some_method
  end
end
```

```ruby
class SomeClass
=begin
This is a top comment.
Or is it?
=end
  def some_method
  end
end
```

Character literals

```
pry(main)> ?a
=> "a"
```

Let’s drop some

redundant

stuff

There's more than one way to do it

(There are way too many ways to do it)

core library method aliases

```
collect => map
inject  => reduce
detect  => find
select  => find_all
sprintf => format
length  => size
raise   => fail
```

Where is filter?

map
reduce
find
filter
format
length
raise

procs

No arity check

Non-local return

Do we really need them?

So many languages are getting by just fine with only lambdas...


THERE CAN BE ONLY ONE

Single-quoted string literals

A ton of obscure %-something literals

%s, %x, %w, %W,

%r, %q, %Q, %, %i

So excited to be here!

```
puts "Hello, Athens!"
puts "Hello, Athens!"
puts "Hello, Athens!"
```

```
for i in 1..3
  puts "Hello, Athens!"
end
```

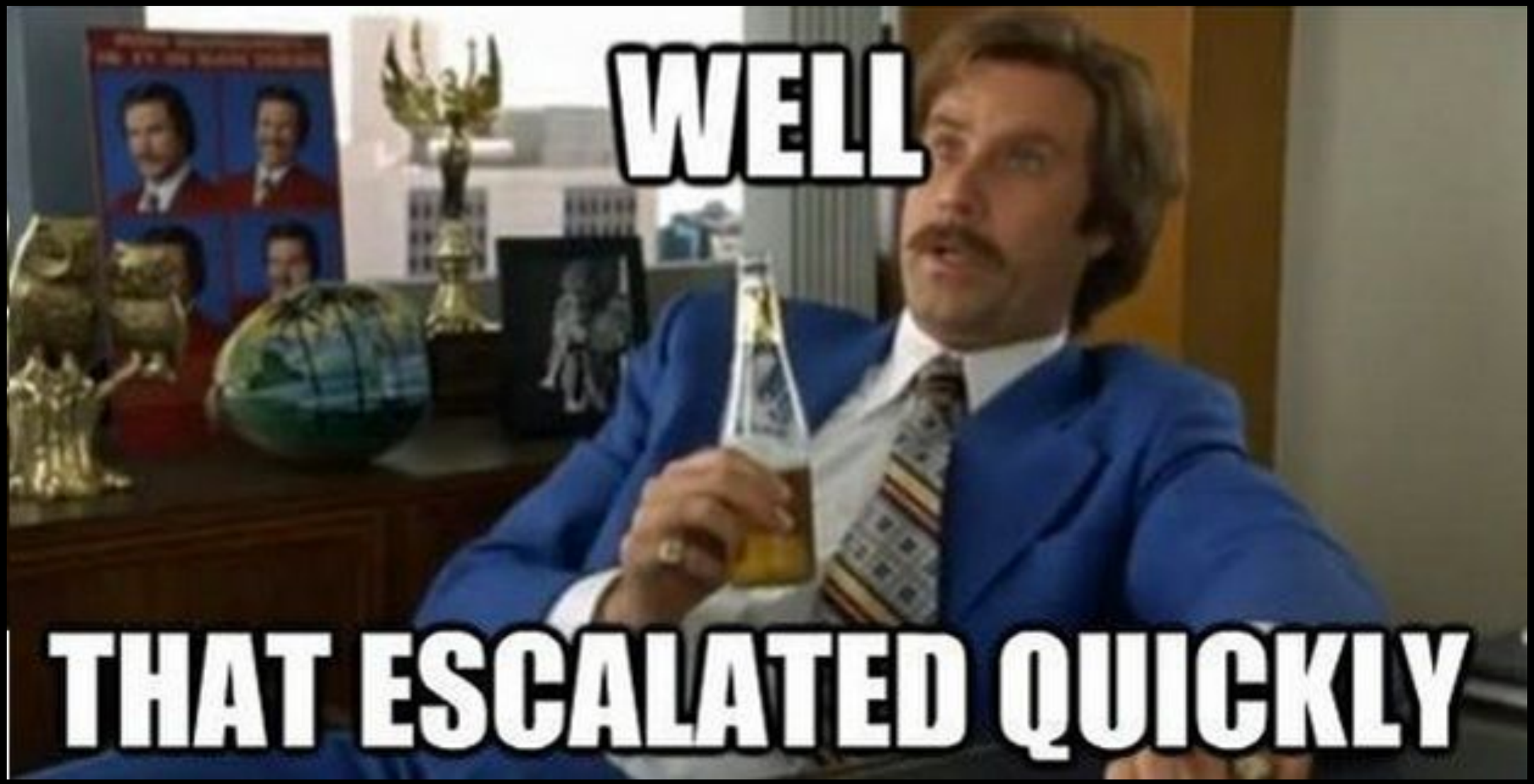
```ruby
3.times do
  puts "Hello, Athens!"
end
```

```ruby
3.times do
  puts %(Hello, Athens!)
end
```

```ruby
3.times do
  puts %Q(Hello, Athens!)
end
```

```ruby
3.times do
  puts 'Hello, Athens!'
end
```

```ruby
3.times do
  puts %q(Hello, Athens!)
end
```


WELL
THAT ESCALATED QUICKLY


DECISION MAKING
alternatives
uncertainty
high-risk consequences
interpersonal issues
complexity

Are all those options worth our while?

STOP BIKESHEDDING AND FOCUS ON WHAT MATTERS

Let’s fix some stuff!

and & or have the same precedence

So many nils floating around

```
pry(main)> "TOP".upcase
=> "TOP"
pry(main)> "TOP".upcase!
=> nil
```

Mutable strings

Even JavaScript got this right...

Reassignable constants

```
pry(main)> A = 5
=> 5
pry(main)> A = 6
(pry):39: warning: already initialized constant A
(pry):38: warning: previous definition of A was here
=> 6
pry(main)> Class = 3
(pry):40: warning: already initialized constant Class
=> 3
pry(main)> Class
=> 3
```

Class variables

```ruby
class Parent
  @@class_var = 'parent'

  def self.print_class_var
    puts @@class_var
  end
end

class Child < Parent
  @@class_var = 'child'
end

Parent.print_class_var # => will print "child"
```

Poorly named methods

Kernel#puts

Kernel#println,
anyone?

Kernel#print

defined?

```
[1] pry(main)> defined? 10
=> "expression"
[2] pry(main)> defined? Test
=> nil
[3] pry(main)> defined? TrueClass
=> "constant"
```

Enumerable#include?

Enumerable#includes?

Kernel#%

```
'%d %d' % [20, 10]
```

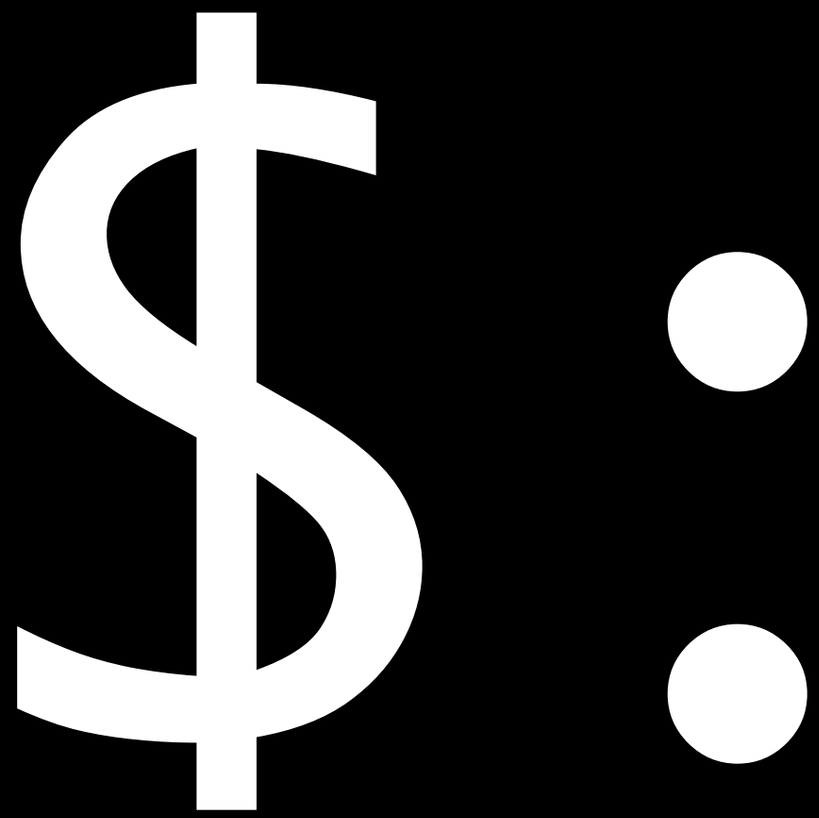
```
sprintf('%d %d', 20, 10)
```

```
sprintf(
  '%{first} %{second}',
  first: 20, second: 10
)
```

```
format('%{first} %{second}',
       first: 20, second: 10)
```

In what universe would you prefer `Kernel#%` over `Kernel#format`???

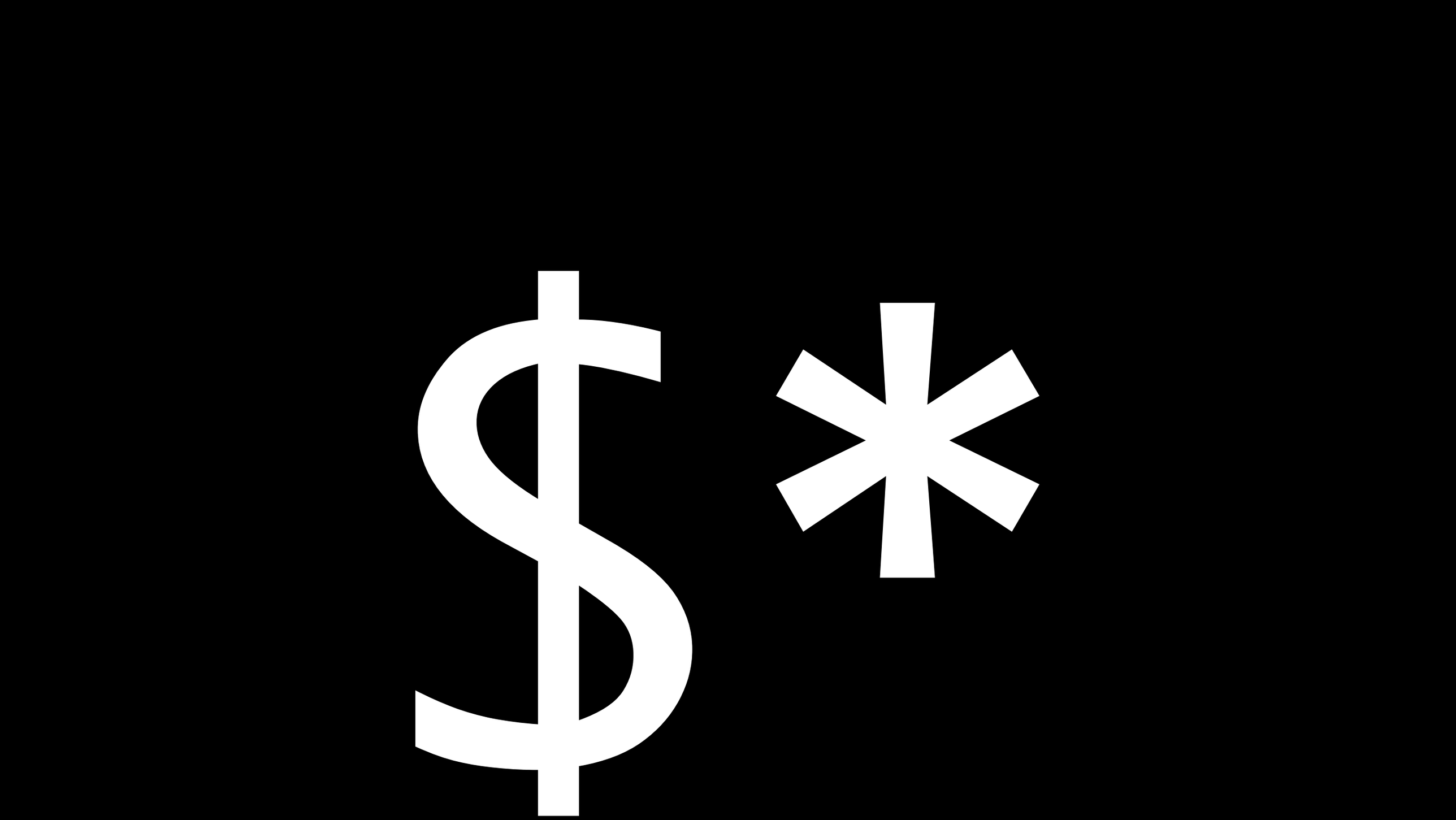
Perl-style global variables



$LOAD_PATH

¢.Ψ,

$FIELD_SEPARATOR



$ARGV

JRuby defines the English aliases by default

Ruby 4.0 will do this as well!

WTF? Global variables?

Even Java doesn't have globals...

THE FUTURE OF THE STANDARD LIBRARY

The Ruby Stdlib is a Ghetto

http://www.mikeperham.com/2010/11/22/the-ruby-stdlib-is-a-ghetto/

A ton of legacy code (often last updated 2000-2003)

Horrible APIs

net/http anyone?

The Kill List

- Net::*
- DRb
- REXML
- RSS
- Rinda
- WEBrick
- XML

What are the parts of the standard library you dislike the most? Why so?

Xavier Noria @fxn · May 8

@bbatsov net/http, basically you need a PhD and @peterc's cheat sheet to do anything

Aaron Cruz @mraaroncruz · May 8

@bbatsov net/http's API is confusing and I have to look it up every time I need more than a basic GET request.

benlovell @benlovell · May 8

@bbatsov timeout.rb because it's broken ass 💩💩💩

Erik Michaels-Ober @sferik · May 8

@bbatsov I dislike that Ruby's standard (and core) libs are rarely updated to integrate new features (e.g. keyword arguments).

Erik Michaels-Ober @sferik · May 8

@bbatsov For example, I'd prefer if Object#methods was changed to accept a keyword argument instead of a boolean when Ruby 2.0 was released.

Erik Michaels-Ober @sferik · May 8

@bbatsov Likewise, I think the initialize method for structs should accept keyword arguments.

Erik Michaels-Ober @sferik · May 8

@bbatsov There are lots of old libraries like REXML that are not actively maintained and not widely used (everyone uses Nokogiri instead).

Erik Michaels-Ober @sferik · May 8

@bbatsov Ditto for URI (Addressable is better).

Erik Michaels-Ober @sferik · May 8

@bbatsov Documentation for many standard libraries is bad or nonexistent.

Erik Michaels-Ober @sferik · May 8

@bbatsov The JSON implementation is not the fastest (that would be Oj) and pollutes all objects with Object#to_json. 😆

Erik Michaels-Ober @sferik · May 8

@bbatsov And don't even get me started on mathn. 😢

1. Move the important bits to the Core Library

2. Remove everything outdated/obscure

3. Leverage modern Ruby feature in the Standard Library

EPILOGUE

When will Ruby 4 be released?

Ruby 4.0 will likely never happen

Ruby 4.0 is already here!

Crystal

Clojure

Elixir

Scala

“The future is already here it's just not evenly distributed.”

–William Gibson


FELINA

One more thing...

Stewardship:
The Sobering Parts

https://www.youtube.com/watch?v=2y5Pv4yN0b0

File tickets

Send patches

Blog about the issues

Speak about the issues

Let's make Ruby better together!


Ruby
The Definitive Guide
David Flanagan
O'REILLY
Unearthing the Excellence in JavaScript
Ruby
The Good Parts
O'REILLY
YAHOO! PRESS
Douglas Crockford

Epilogue

twitter: @bbatsov
github: @bbatsov
http//batsov.com
http://emacsredux.com

Athens Ruby Meetup
Athens,
Greece
18.11.2016