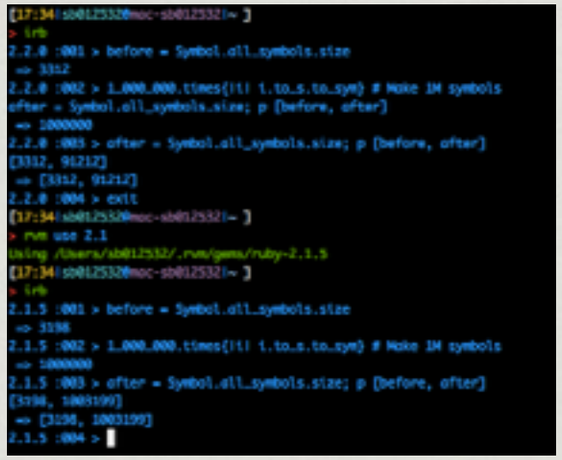
RUBY 2.2

New Features

- Symbol GC (needed for Rails 5.0)
- Incremental GC
- Use frozen string literals for Hash
- Fast keyword arguments passing
- http://www.atdot.net/~ko1/activities/2014_rubyconf_pub.pdf
- https://github.com/ruby/ruby/blob/v2_2_0/NEWS

Symbol GC

- https://bugs.ruby-lang.org/issues/9634
- http://www.slideshare.net/authorNari/symbol-gc



mortal symbols - Mortal symbols, e.g. "foo".to_sym, do not have an integer id and are thus garbage collectable.

immortal symbols - Immortal symbols continue to use integer id's and so are never GC'ed. Examples of immortal symbols include method names, variable names, constants and other language elements.

Incremental GC

- https://bugs.ruby-lang.org/issues/10137
- https://www.omniref.com/blog/blog/2014/11/18/ko1-at-rubyconf-2014-massive-garbage-collection-speedup-in-ruby-2-dot-2/

http://patshaughnessy.net/2013/10/30/generational-gc-in-python-and-ruby

Generational GC - classifies objects into generations

Frozen Hash Strings

- Almost as fast as symbols
- http://www.sitepoint.com/unraveling-string-key-performance-ruby-2-2/

Fast Keyword Arguments

- https://bugs.ruby-lang.org/issues/10440
- 15 times faster
- “In order to address this problem, we are reimplementing how Ruby 2.2 handles keyword arguments to avoid creating Hash objects as much as possible. Meanwhile, we are implementing a new design that will collect the names of keyword arguments at compile time, so that caller need only pass the values at runtime. The callee will then recombine the values with the names collected by the compiler. “ - Koichi Sasada
- https://www.omniref.com/blog/blog/2014/11/18/ko1-at-rubyconf-2014-massive-garbage-collection-speedup-in-ruby-2-dot-2/

```
[17:42|sb012532@moc-sb012532|~ ]
> irb
2.1.5 :001 > def foo(k1: nil, k2: nil, k3: nil, k4: nil, k5: nil, k6: nil)
2.1.5 :002?>   end
 => :foo
2.1.5 :003 >
2.1.5 :004 >   require "benchmark"
 => true
2.1.5 :005 >
2.1.5 :006 >   Benchmark.bm do |x|
2.1.5 :007 >       x.report { 50_000_000.times{foo(k1: 1, k2: 2, k3: 3, k4: 4, k5: 5, k6: 6)} }
2.1.5 :008?>   end
       user     system      total        real

 50.740000   0.510000  51.250000 ( 51.589016)
 => [#<Benchmark::Tms:0x007f9f4c86d448 @label="", @real=51.589016, @cstime=0.0, @cutime=0.0, @stime=0.51, @utime=50.74000000000002, @total=51.25000000000004>]
2.1.5 :009 >
2.1.5 :010 >   exit
[17:44|sb012532@moc-sb012532|~ ]
> rvm use 2.2.0
Using /Users/sb012532/.rvm/gems/ruby-2.2.0
[17:44|sb012532@moc-sb012532|~ ]
> irb
2.2.0 :001 > def foo(k1: nil, k2: nil, k3: nil, k4: nil, k5: nil, k6: nil)
2.2.0 :002?>   end
 => :foo
2.2.0 :003 >
2.2.0 :004 >   require "benchmark"
 => true
2.2.0 :005 >
2.2.0 :006 >   Benchmark.bm do |x|
2.2.0 :007 >       x.report { 10.000.000.times{foo(k1: 1, k2: 2, k3: 3, k4: 4, k5: 5, k6: 6)} }
2.2.0 :008?>   end
       user     system      total        real
   1.750000   0.000000   1.750000 (  1.751923)
 => [#<Benchmark::Tms:0x007fb7ec1268e8 @label="", @real=1.7519230809994042, @cstime=0.0, @cutime=0.0, @stime=0.0, @utime=1.75, @total=1.75>]
2.2.0 :009 >
```