Ruby trunk - Feature #8948

Frozen regex

09/25/2013 04:02 AM - sawa (Tsuyoshi Sawada)

Status:	Assigned
Priority:	Normal
Assignee:	matz (Yukihiro Matsumoto)
Target version:	

Description

=begin
I see that frozen string was accepted for Ruby 2.1, and frozen array and hash are proposed in https://bugs.ruby-lang.org/issues/8909 . I feel there is even more use case for a frozen regex, i.e., a regex literal that generates a regex only once. It is frequent to have a regex within a frequently repeated portion of code, and generating the same regex each time is a waste of resource. At the moment, we can have a code like:

```
class Foo
  RE1 = /pattern1/
  RE2 = /pattern1/
  RE3 = /pattern1/
  def classify
    case self
    when RE1 then 1
    when RE2 then 2
    when RE3 then 3
    else 4
    end
  end
end
```

but suppose we have a frozen Regexp literal //f. Then we can write like:

```
class Foo
  def classify
    case self
    when /pattern1/f then 1
    when /pattern1/f then 2
    when /pattern1/f then 3
    else 4
    end
  end
end
```

=end

History

#1 - 09/25/2013 04:08 AM - sawa (Tsuyoshi Sawada)

Sorry, there was a mistake in the above. The three regexes with the same content /pattern1/ (or /pattern1/f) in the respective examples are supposed to represent different patterns.

#2 - 09/25/2013 06:00 AM - Eregon (Benoit Daloze)

We already have immutable (created only once) regexps: it is always the case for literal regexps and for dynamic regexps you need the 'o' flag: /a#{2}b/o.

So there are in practice immutable, but currently not #frozen?. Do you want to request it? I think it makes sense.

You can check with #object_id to know if 2 references reference the same object.

```
def r; /ab/; end
r.object_id
=> 2160323760
r.object_id
=> 2160323760
```

```
def s; /a#{2}b/; end
s.object_id
=> 2153197860
s.object_id
=> 2160163740

def t; /a#{2}b/o; end
t.object_id
=> 2160181200
t.object_id
=> 2160181200
```

#3 - 09/25/2013 08:42 AM - sawa (Tsuyoshi Sawada)

Eregon, thank you for the information.

#4 - 09/25/2013 09:46 PM - Eregon (Benoit Daloze)

- Status changed from Open to Feedback

sawa: do you want to request Regexp to always be #frozen? or should the issue be closed?

#5 - 09/25/2013 11:46 PM - jwille (Jens Wille)

besides regexps being frozen, there might still be a use case for regexp literals that would only be allocated once:

```
def r1; /ab/; end; r1.object_id  #=> 70043421664620
def r2; /ab/; end; r2.object_id  #=> 70043421398060

def r3; /ab/f; end; r3.object_id  #=> 70043421033140
def r4; /ab/f; end; r4.object_id  #=> 70043421033140
```

i think it's in the same vein as #8579 and #8909.

#6 - 09/26/2013 02:47 PM - sawa (Tsuyoshi Sawada)

jwille, I agree with the use case, but it would be difficult to tell which regexes are intended to be the same, so I would not request that feature.

Probably, it makes sense to have all static regexes frozen, and have the f flag freeze dynamic regexes as well. I can't think of a use case for a regex that is immutable but not frozen. I am actually not clear about the difference.

#7 - 09/27/2013 08:15 PM - jwille (Jens Wille)

> but it would be difficult to tell which regexes are intended to be the same

i'm not sure i understand. how is

```
def r1; /ab/f; end
def r2; /ab/f; end
```

different from

```
def s1; 'ab'f; end
def s2; 'ab'f; end
```

?

#8 - 09/29/2013 06:35 PM - sawa (Tsuyoshi Sawada)

jwille,

My understanding with the case of string in your example is that the two strings would count as different strings, but for respective method calls would not create new strings. It would mean one of the string can be "ab" and the other a different string such as "cd".

If that is what you intended for your regex examples, then there is no difference.

#9 - 09/30/2013 12:11 PM - ko1 (Koichi Sasada)

- Category set to syntax

- Assignee set to matz (Yukihiro Matsumoto)

- Target version set to 2.6

I like to freeze normal regexp literal that Eregon said.

2.2 matter?

Anyone set instance variable for each regexp? :)

#10 - 09/30/2013 05:25 PM - Eregon (Benoit Daloze)

ko1 (Koichi Sasada) wrote:

> 2.2 matter?

2.1 would make sense to me, so it goes along with other frozen literals.

> Anyone set instance variable for each regexp? :)

I highly doubt it.

#11 - 08/21/2015 05:31 AM - ko1 (Koichi Sasada)

- Status changed from Feedback to Assigned

There are two options:

1. Freeze only literal regexps
2. Freeze all of regexps

I like (2) because I have no idea to change regexp objects.

History of "Frozen":

- Ruby 2.0
  - Integer (Bignum, Fixnum) https://bugs.ruby-lang.org/issues/3222
  - Float https://bugs.ruby-lang.org/issues/6936
- Ruby 2.1
  - Symbols https://bugs.ruby-lang.org/issues/8906
- Ruby 2.2
  - nil/true/false https://bugs.ruby-lang.org/issues/8923

#12 - 08/21/2015 05:56 AM - ko1 (Koichi Sasada)

A patch is small.

```
Index: re.c
===================================================================
--- re.c    (revision 51650)
+++ re.c    (working copy)
@@ -2548,6 +2548,8 @@
     if (!re->ptr) return -1;
     RB_OBJ_WRITE(obj, &re->src, rb_fstring(rb_enc_str_new(s, len, enc)));
     RB_GC_GUARD(unescaped);
+
+    rb_obj_freeze(obj);
     return 0;
 }
```

But I got many failures on rubyspec.
https://gist.github.com/ko1/da52575de115c928ce4a

#13 - 09/06/2017 09:31 AM - ko1 (Koichi Sasada)

Now we can't run make test-all because test/lib/test/unit.rb defines singleton method for a regexp object.

```
      def non_options(files, options)
        filter = options[:filter]
        if filter
          pos_pat = /\A\/(.*)\/\z/
          neg_pat = /\A!\/(.*)\/\z/
```

```
          negative, positive = filter.partition {|s| neg_pat =~ s}
          if positive.empty?
            filter = nil
          elsif negative.empty? and positive.size == 1 and pos_pat !~ positive[0]
            filter = positive[0]
          else
            filter = Regexp.union(*positive.map! {|s| Regexp.new(s[pos_pat, 1] || "\\A#{Regexp.quote(s)}\\z"
)})
          end
          unless negative.empty?
            negative = Regexp.union(*negative.map! {|s| Regexp.new(s[neg_pat, 1])})
            filter = /\A(?=.*#{filter})(?!.*#{negative})/
          end
          if Regexp === filter
            # bypass conversion in minitest
            def filter.=~(other)    # :nodoc:   <-- singleton method!!
              super unless Regexp === other
            end
          end
          options[:filter] = filter
        end
        true
      end
```

Is it acceptable (should we allow such code)?

#14 - 10/19/2017 08:26 AM - matz (Yukihiro Matsumoto)

I agree with frozen regexp literals. If you want to freeze all regexp objects, you need more info to persuade me.

Matz.

#15 - 10/19/2017 08:58 AM - naruse (Yui NARUSE)

This change will break compatibility.
If the change has clear benefit, such incompatibility is acceptable.
But this benefit seems not so much.

If so this should be postponed on introducing parallelization.

#16 - 10/19/2017 11:10 AM - Eregon (Benoit Daloze)

matz (Yukihiro Matsumoto) wrote:

> I agree with frozen regexp literals. If you want to freeze all regexp objects, you need more info to persuade me.
>
> Matz.

I think the inconsistency between Regexp being freezed by default or not (based on being literals) would be confusing.
I believe it would be better all Regexp frozen or none frozen, but not something in between.

#17 - 10/20/2017 01:40 AM - shyouhei (Shyouhei Urabe)

We had a discussion around this issue at yesterday's developer meeting.

Seems everyone there agreed that in long term future, regex shall be frozen. However the status quo is that there are wild applications that do override regex methods, particularly for testing purposes. So there needs to be some transition paths.

Freezing regex literals sounded a good starting point. We already optimize the return value of them:

```
3.times {
  p(/foo/.object_id)
}
```

The merit of freezing regex literals are:

- Seems very safe to do. If it hurts someone, that means that person has already been bothered by the literal being shared among evaluations.

The disadvantage of freezing regex literals are:

- It benefits no one while it does confuse people. Regexp class has no built-in destructive methods so freezing them does almost nothing; just prohibits method redefinitions. Is it worth doing by deafult?

#18 - 10/20/2017 11:02 PM - Eregon (Benoit Daloze)

> Regexp class has no built-in destructive methods so freezing them does almost nothing; just prohibits method redefinitions.

I think it is useful in communicating this is a immutable object.
So singleton methods definitions are prevented (it has to behave like every other Regexp), but methods can be added to the Regexp class.
Adding instances variables to the Regexp are also prevented, which seems a good idea as attaching state to immutable object makes no sense.

I agree with the rationale that it seems safer to first only freeze Regexp literals.
Does /#{2*3}/ counts as literal though? That expression produces different instances, unlike the /#{2*3}/o variant.
I think it should be considered as literal and be frozen.
Then the distinction is between literals and Regexp.new, which is much clearer than "literals except those with interpolation but without /o".