Ruby master - Bug #14148

Longstanding behavior regarding correspondence of toplevel with Object is surprising

12/02/2017 12:14 AM - RickHull (Rick Hull)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.5.0preview1 (2017-10-10 trunk 60153) [x86_64-linux]	**Backport:**	2.3: UNKNOWN, 2.4: UNKNOWN

Description

```
module Kernel
  X = 1
end

puts String::X

X = 2

puts String::X

$ ruby test.rb
1
Traceback (most recent call last):
test.rb:9:in `<main>': uninitialized constant String::X (NameError)
Did you mean?  X

$ ruby --version
ruby 2.5.0preview1 (2017-10-10 trunk 60153) [x86_64-linux]
```

Related issues:

Related to Ruby master - Feature #11547: remove top-level constant lookup	**Closed**

History

#1 - 12/02/2017 12:15 AM - RickHull (Rick Hull)

- Subject changed from Ruby 2.5.0-preview1 - broken constant lookup to Ruby 2.5.0-preview1 - NameError on scoped constant after toplevel constant is defined

#2 - 12/02/2017 12:28 AM - RickHull (Rick Hull)

Expected behavior: String::X should continue to evaluate to 1 after X = 2 at the toplevel

Actual behavior: String::X results in NameError after X = 2 at the toplevel

more strangeness:

```
class Object
  X = 1
end

String.superclass #=> Object

String::X #=> NameError

class C
  X = 1
end

class D < C; end

D.superclass #=> C

D::X #=> 1
```

Expected behavior: Since String is a subclass of Object, where Object::X is defined and initialized, so should String::X. This behavior holds with user-created classes C and D.

Actual behavior: Despite Object::X being defined and initialized, String::X results in NameError

#3 - 12/02/2017 12:36 AM - RickHull (Rick Hull)

Behavior on 2.4.0p0:

```
module Kernel
  X = 1
end

String::X #=> 1

X = 2

String::X #=> 2
# warning: toplevel constant X referenced by String::X

class Object
  X = 1
end

String.superclass #=> Object

String::X #=> 1
# warning: toplevel constant X referenced by String::X
```

Comments: This looks like equivalent behavior to 2.5. The "strangeness" is present in 2.4. The only difference is the expected change from warning to NameError.

#4 - 12/02/2017 01:36 AM - RickHull (Rick Hull)

Further down the rabbit hole, ruby 2.4.0p0:

```
class BasicObject
  X = 1
end

String::X #=> 1

class Object
  X = 2
end

String::X #=> 2
# warning: toplevel constant X referenced by String::X
```

On ruby 2.5.0-preview1:

```
class BasicObject
  X = 1
end

String::X #=> 1

class Object
  X = 2
end

String::X #=> NameError

String.const_get(:X) #=> 2
```

#5 - 12/02/2017 08:07 AM - RickHull (Rick Hull)

After some clarifying discussions with matthewd (Matthew Draper), I don't think is a problem for 2.5. The overall behavior seems consistent with 2.4, if surprising. My surprise mostly revolves around the strange correspondence of toplevel with Object. This bug report is not about the escalation of warning to NameError.

What about constants defined on Kernel and BasicObject?

```
module Kernel
  K = 1
end
```

```
class BasicObject
  B = 2
end

class Object
  O = 3
end

K #=> 1
B #=> 2
O #=> 3

String::K #=> 1
String::B #=> 2
String::O #=> NameError
String.const_get(:O) #=> 3
```

#6 - 12/02/2017 01:34 PM - nobu (Nobuyoshi Nakada)

Only Kernel and BasicObject, but not modules included at the top level?
It feels weird a little to me.

#7 - 12/02/2017 06:18 PM - RickHull (Rick Hull)

- Subject changed from Ruby 2.5.0-preview1 - NameError on scoped constant after toplevel constant is defined to Longstanding behavior regarding correspondence of toplevel with Object is surprising

I edited the title of this issue. I reported this initially as 2.5.0 issue, as I discovered it while playing around with the 2.5.0 changes regarding toplevel constant lookup. Now, this ticket is more about surprising behavior in general, having to do with the "strange" correspondence between toplevel and Object -- e.g. https://banisterfiend.wordpress.com/2010/11/23/what-is-the-ruby-top-level -- that I believe could or should be fixed at the design level.

As this is longstanding behavior, I understand if this is a WONTFIX or similar. Furthermore, lacking detailed knowledge of the history and internals that led to this state of affairs, I will refrain from commenting further unless prompted.

In terms of design, I think that the toplevel should be its own "special" scope, not derived from Object. I'm not sure how "special" it needs to be. I think toplevel constants should be something like toplevel::X and not correspond to Object::X. Naked methods like def foo could likewise correspond to a hidden toplevel#foo or main#foo. But I don't think I have the level of understanding to make a concrete proposal that would preserve some level of compatibility while meeting the expectations of both new and longtime rubyists.

#8 - 12/02/2017 09:42 PM - shevegen (Robert A. Heiler)

> should be something like toplevel::X

I think that one problem here is that a new method? Or keyword, aka
"toplevel", would be added in the case that you showed. So that would
be a change to the status quo and I am not sure it is (or was) entirely
necessary.

I guess it would be nice if ruby could clearly document what toplevel
is and make this available somewhere, be it part of the official docs
or the ruby wiki here.

I remember having had read on the old pickaxe that the methods on
"toplevel" become private methods of class Object, so perhaps this
may explain some of the behaviour. But either way, I think it is best
to describe the whole toplevel situation rathern than have to rely
on blog entries (not that blog entries are bad, mind you; I just think
that the documentation of ruby should ideally be kept within official
ruby).

Search for toplevel yields some weird results by the way. :D

First, some blogs are found, and then RDoc:

https://ruby-doc.org/stdlib-2.2.1/libdoc/rdoc/rdoc/RDoc/TopLevel.html

Pretty hilarious to me. :D

> But I don't think I have the level of understanding to make a
> concrete proposal that would preserve some level of compatibility
> while meeting the expectations of both new and longtime rubyists.

I would not either, but I think either way, it should be documented
somewhere. I mean, to me it is not a huge result in the sense that
after reading the old pickaxe on "def foo" on toplevel becoming
a part of class Object, so it's really not something that confuses
me personally. But perhaps newcomers may be confused, and here it
would be helpful if they could read up on some of the situation of
toplevel.

#9 - 12/03/2017 09:57 PM - shevegen (Robert A. Heiler)

Actually, I think there may also be a bug ... I can not yet pinpoint
what causes it and I have to soon go to get some sleep, but switching
between ruby version triggers a NameError for me:

ruby 2.5.0dev (2017-11-30 trunk 60945) [x86_64-linux]

```
atomic_composition.rb:48:in `atomic_composition': uninitialized constant Bioroebe::Constants::FILE_AMINOACIDS_
MOLECULAR_FORMULA (NameError)
```

The line in question is:

```
dataset_molecular_formula_for_the_aminoacids = YAML.load_file(
  ::Bioroebe::Constants::FILE_AMINOACIDS_MOLECULAR_FORMULA
)
```

(I am just loading a yaml file... the above is a bit verbose but
it definitely works on ruby 2.4.x)

It works when I remove the '::Bioroebe::' part, but running it
with ruby 2.5.x, I run into another error lateron

```
Traceback (most recent call last):
        4: from atomic_composition.rb:111:in `<main>'
        3: from atomic_composition.rb:70:in `atomic_composition'
        2: from atomic_composition.rb:70:in `with_index'
        1: from atomic_composition.rb:70:in `map'
atomic_composition.rb:82:in `block in atomic_composition': undefined method `remove_this_molecule_from' for Ch
emistryParadise:Module (NoMethodError)
```

With the line being:

```
::ChemistryParadise.remove_this_molecule_from('OH', formula_for_this_amino_acid)
```

So I assume it is the leading '::'.

I am not 100% positive, perhaps it is a feature rather than a bug, but I am absolutely
sure that something between 2.4.2 and 2.5.0 changed there, leading to code that does
not yet work (for me) on 2.5.0.

#10 - 12/04/2017 12:37 AM - mame (Yusuke Endoh)

This change is intended. See https://bugs.ruby-lang.org/issues/11547. Matz said in that ticket:

> I am for this proposal, but also concern about code breakage. Let's try removing top-level constant look-up in 2.4dev and see how much code it breaks.

I think this change will not be cancelled just because you are surprised. ("Surprise" is NG word for persuasion in Ruby!) This change might be cancelled if it breaks an existing code. Did your actual code break?

#11 - 12/04/2017 04:09 AM - RickHull (Rick Hull)

Hi mame (Yusuke Endoh),

I agree that the change from warning to NameError is intended, and this behavior change is not the concern:

> I don't think is a problem for 2.5. The overall behavior seems consistent with 2.4, if surprising. My surprise mostly revolves around the strange correspondence of toplevel with Object. This bug report is not about the escalation of warning to NameError.

When I first created this ticket, I was not aware of the "strange correspondence" between toplevel and Object, and the strange behavior resulting from this correspondence is now what this ticket is about. Again, this is longstanding behavior, so I do understand if there is no desire to change this behavior, and I certainly wouldn't suggest such a change for the 2.5 series. Maybe for Ruby 3. Again, I think this is a design bug or else a "wart". I apologize if the ticket is confusing and take all the credit for that :)

The only connection to 2.5 and https://bugs.ruby-lang.org/issues/11547 at this point, is that I was studying ticket 11547 when I stumbled across the behavior I detailed above. Finally, as a side note, the proposal for 11547 seems ignorant of the correspondence between toplevel and Object, mirroring my own ignorance. I am sure matz (Yukihiro Matsumoto)nobu (Nobuyoshi Nakada) et al understand this correspondence implicitly, and this is why they expressed so much caution regarding the change, which, if the scope were truly limited to just toplevel and did not include Object constants, would be a "no-brainer".

In all honesty, this ticket should probably be closed and maybe shifted to a mailing list discussion. But I will leave that in other, more capable hands.

Cheers,
Rick

#12 - 12/04/2017 07:25 AM - mrkn (Kenta Murata)

- Related to Feature #11547: remove top-level constant lookup added

#13 - 07/26/2019 10:41 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Open to Closed