Ruby - Feature #11547

remove top-level constant lookup

09/23/2015 01:43 PM - gucki1 (Corin Langosch)

Status:	Closed	
Priority:	Normal	
Assignee:		
Target version:		

Description

If ruby cannot find a class in the specified scope it uses the top-level constant of the same name if it exists and emits a warning:

```
irb(main):006:0> class Auth; end
=> nil
irb(main):007:0> class Twitter; end
=> nil
irb(main):008:0> Twitter::Auth
(irb):8: warning: toplevel constant Auth referenced by Twitter::Auth
=> Auth
```

In some cases this is not playing nicely with rails autoloading as can be seen here: https://github.com/rails/rails/issues/6931. Many more issues like this exist.

Imo I don't see any reason why this fallback makes any sense. So I'd like to suggest to remove it completely or at least add an option to disable it.

Related issues:

Related to Ruby - Bug #14148: Longstanding behavior regarding correspondence ...	**Closed**
Related to Ruby - Bug #14407: defined? still returning true for top-level con...	**Closed**
Related to Ruby - Bug #18622: const_get still looks in Object, while lexical ...	**Closed**

Associated revisions

Revision 44a2576f798b07139adde2d279e48fdbe71a0148 - 01/01/2017 09:07 AM - nobu (Nobuyoshi Nakada)

variable.c: top-level constant look-up

- variable.c (rb_const_search): [EXPERIMENTAL] remove top-level constant look-up. [Feature #11547]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@57244 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 44a2576f - 01/01/2017 09:07 AM - nobu (Nobuyoshi Nakada)

variable.c: top-level constant look-up

- variable.c (rb_const_search): [EXPERIMENTAL] remove top-level constant look-up. [Feature #11547]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@57244 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 9df88e9cae57aa421230f14500e88f33f127414f - 01/01/2017 09:17 AM - nobu (Nobuyoshi Nakada)

test for [Feature #11547]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@57245 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 9df88e9c - 01/01/2017 09:17 AM - nobu (Nobuyoshi Nakada)

test for [Feature #11547]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@57245 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 10/28/2015 06:32 AM - shugo (Shugo Maeda)

- Status changed from Open to Feedback

Corin Langosch wrote:

> If ruby cannot find a class in the specified scope it uses the top-level constant of the same name if it exists and emits a warning:
> (snip)
> Imo I don't see any reason why this fallback makes any sense. So I'd like to suggest to remove it completely or at least add an option to disable it.

How should the following code behave?

```
class Foo
  X = 1
end

module Bar
  Y = 2
end

class Baz < Foo
  include Bar
end

p Baz::X
p Baz::Y
```

#2 - 12/10/2015 10:00 PM - gucki1 (Corin Langosch)

Hi Shugo. Just as it does now, it doesn't perform any magic (fallback to some other scope) and doesn't emit any warning. How is it related to my bug report/ feature request? Cheers, Corin.

#3 - 12/15/2015 04:28 AM - shugo (Shugo Maeda)

Corin Langosch wrote:

> Hi Shugo. Just as it does now, it doesn't perform any magic (fallback to some other scope) and doesn't emit any warning. How is it related to my bug report/ feature request? Cheers, Corin.

In my example, Baz::X and Baz::Y refer to constants defined in ancestors of Baz.
In your example, Twitter::Auth also refers to a constant defined one of its ancestors, Object.

So, I'd like to clarify what behavior do you want in these cases.

Instead of changing the behavior of constant lookup, we may be able to introduce a variant of const_missing which is invoked when a constant is not directly defined in the target class.

#4 - 02/11/2016 03:33 AM - bronson (Scott Bronson)

Corin, I completely agree. Recently, Rails's nondeterministic autoload made it very hard for me to discover this problem. Without the insightful warning, I would have been sunk.

Shugo Maeda wrote:

> Instead of changing the behavior of constant lookup, we may be able to introduce a variant of const_missing which is invoked when a constant is not directly defined in the target class.

Couldn't Ruby 3.0 just raise an error instead? Shugo, your example seems to demonstrate that Ruby is smart enough to realize when the behavior is intentional vs. when it's probably an accident.

I'd be happy to bang together a patch if the concept seems sound.

#5 - 03/04/2016 08:48 AM - shugo (Shugo Maeda)

Scott Bronson wrote:

> Corin, I completely agree. Recently, Rails's nondeterministic autoload made it very hard for me to discover this problem. Without the insightful warning, I would have been sunk.
>
> Shugo Maeda wrote:
>
> > Instead of changing the behavior of constant lookup, we may be able to introduce a variant of const_missing which is invoked when a constant is not directly defined in the target class.

> Couldn't Ruby 3.0 just raise an error instead? Shugo, your example seems to demonstrate that Ruby is smart enough to realize when the behavior is intentional vs. when it's probably an accident.
>
> I'd be happy to bang together a patch if the concept seems sound.

The behavior change might be acceptable in Ruby 3.0 if Matz wants.
Experiments with real world applications such as Rails might help his decision.

#6 - 04/13/2016 05:24 AM - matz (Yukihiro Matsumoto)

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.

Matz.

#7 - 12/30/2016 11:45 AM - tisba (Sebastian Cohnen)

With 2.4 released and I wasn't able to find anything related to this issue in https://github.com/ruby/ruby/blob/v2_4_0/NEWS I guess it did not make it into 2.4.

I was wondering if there are any plans regarding working on this. In almost all projects I encounter this issue and it would be great if this could improve.

#8 - 01/01/2017 09:06 AM - nobu (Nobuyoshi Nakada)

Sorry, missed this feature.

#9 - 01/01/2017 09:08 AM - nobu (Nobuyoshi Nakada)

- Status changed from Feedback to Closed

Applied in changeset r57244.

---

variable.c: top-level constant look-up

- variable.c (rb_const_search): [EXPERIMENTAL] remove top-level
  constant look-up. [Feature #11547]

#10 - 01/01/2017 02:58 PM - bronson (Scott Bronson)

The fix is even simpler than what I was picturing: https://github.com/ruby/ruby/commit/44a2576f798b07139adde2d279e48fdbe71a0148

If it sticks, this will make a bunch of Rails autoload issues go away. Right on nobu, thanks!

#11 - 01/01/2017 03:07 PM - fxn (Xavier Noria)

In particular this one http://guides.rubyonrails.org/autoloading_and_reloading_constants.html#when-constants-aren-t-missed.

We'd love to base Rails autoloading on Kernel#autoload and have the exact same semantics, but that is not possible as of this writing (http://guides.rubyonrails.org/autoloading_and_reloading_constants.html#module-autoload-isn-t-involved). So, Rails does not attempt to emulate constant resolution at all (it does not have nesting information, it does not follow ancestor chains by design, etc.), you have to think autoloading as a feature with its own contract. You have to program against that contract.

A change like this one is welcome though. Thanks.

#12 - 01/02/2017 11:31 PM - zenspider (Ryan Davis)

I would honestly rather see it raise an exception instead of silently return nil. I'd rather KNOW my bug than hunt it.

#13 - 01/03/2017 07:07 PM - fxn (Xavier Noria)

Oh, didn't look at the patch. This means String::Hash returns nil?

#14 - 01/03/2017 07:12 PM - fxn (Xavier Noria)

Ah, no, https://bugs.ruby-lang.org/projects/ruby-trunk/repository/revisions/57244/entry/variable.c#L419 seems to say that Qundef is what the search function returns to indicate failure, raising in the caller.

#15 - 11/02/2017 09:51 AM - fxn (Xavier Noria)

I realized that Object does not halt the lookup, but it is rather just skipped. For example, constants in ancestors of Object are still accessible as qualified constants:

```
module Kernel
  X = 1
end

X         # top-level accessible constant
String::X # still accessible this way
```

Is that intentional?

#16 - 12/01/2017 11:47 PM - RickHull (Rick Hull)

Hm, this is surprising:

```
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]
```

#17 - 12/04/2017 07:25 AM - mrkn (Kenta Murata)

- Related to Bug #14148: Longstanding behavior regarding correspondence of toplevel with Object is surprising added

#18 - 03/07/2022 05:14 PM - Eregon (Benoit Daloze)

- Related to Bug #14407: defined? still returning true for top-level constant when referenced with scope added

#19 - 03/10/2022 04:08 PM - Eregon (Benoit Daloze)

There is some inconsistency here between literal constant lookup and the meta API (const_get).
const_get still looks in Object, even though that's confusing, inconsistent and IMHO shouldn't really happen.

```
module ConstantSpecsTwo
  Foo = :cs_two_foo
end

module ConstantSpecs
end

p ConstantSpecs.const_get("ConstantSpecsTwo::Foo") # => :cs_two_foo
p ConstantSpecs::ConstantSpecsTwo::Foo
# => const_get.rb:9:in `<main>': uninitialized constant ConstantSpecs::ConstantSpecsTwo (NameError)
```

#20 - 03/10/2022 04:13 PM - Eregon (Benoit Daloze)

- Related to Bug #18622: const_get still looks in Object, while lexical constant lookup no longer does added

#21 - 03/11/2022 06:55 PM - fxn (Xavier Noria)

(Pervious comment moved to #18622)