Ruby master - Bug #10661

The "possible reference to past scope" warning is quite frustrating and is forcing me to change my variable names from what I want

12/27/2014 09:14 AM - myronmarston (Myron Marston)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.2.0p0 (2014-12-25 revision 49005) [x86_64-darwin12.0]	**Backport:**	2.0.0: DONTNEED, 2.1: DONTNEED, 2.2: DONE

Description

I find the change in r48986 to be quite frustrating. It's forcing me to change many of my variable and/or method names if I want to keep my ruby code warning free (which is a thing we enforce in the RSpec code base).

The problem I see is that, in my experience, it's quite common to use the same name for a local variable in one part of a file that you later use for an arg-less method name at a later part in the file.

Consider this ruby command:

```
ruby -w -e '[1, 2, 3].sample.tap { |rand| puts "Random value: #{rand}" }; puts "Another random val
ue: #{rand}"'
```

This produces:

```
-e:1: warning: possible reference to past scope - rand
Random value: 1
Another random value: 0.7483347748677992
```

Changing the rand call to self.rand is one solution I would consider to avoid the warning, but it doesn't work here because rand is private (as it comes from Kernel), so I'm forced to change the block local variable name to a name I do not want.

In RSpec it's an even bigger issue as it's quite common to have a common name for a certain collaborator role in your tests where in some cases there's a helper method (often defined using let) that exposes an object for that role and in other tests you might build it in-line and assign it to a local. In our rspec-mocks test suite, we had 280 warnings from this. I went through and changed many variable and method names to names I do not like as much (e.g. the_dbl instead of dbl or klazz instead of klass) simply to avoid this warning:

https://github.com/rspec/rspec-mocks/commit/3b909ed1a951bbca340ea98c27ab65da7f43881c

While I can understand the kinds of errors this warnings helps you avoid, I think that it is too strict and noisy in its current form.

Related issues:

Related to Ruby master - Feature #14153: [PATCH] resurrection of # -*- warn_p...	**Open**

Associated revisions

Revision ac47d6a8 - 12/31/2014 07:48 AM - nobu (Nobuyoshi Nakada)

parse.y: disable past scope warnings

- parse.y (gettable_gen): disable warnings of possible reference to a local variable defined in a past scope. [ruby-core:67162] [Bug #10661]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@49082 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 49082 - 12/31/2014 07:48 AM - nobu (Nobuyoshi Nakada)

parse.y: disable past scope warnings

- parse.y (gettable_gen): disable warnings of possible reference to a local variable defined in a past scope. [ruby-core:67162] [Bug #10661]

Revision 49082 - 12/31/2014 07:48 AM - nobu (Nobuyoshi Nakada)

parse.y: disable past scope warnings

- parse.y (gettable_gen): disable warnings of possible reference to a local variable defined in a past scope. [ruby-core:67162] [Bug #10661]

Revision 49082 - 12/31/2014 07:48 AM - nobu (Nobuyoshi Nakada)

parse.y: disable past scope warnings

- parse.y (gettable_gen): disable warnings of possible reference to a local variable defined in a past scope. [ruby-core:67162] [Bug #10661]

Revision b5623f98 - 02/05/2015 07:30 AM - naruse (Yui NARUSE)

merge revision(s) 49082: [Backport #10661]

```
    * parse.y (gettable_gen): disable warnings of possible reference
      to a local variable defined in a past scope.
      [ruby-core:67162] [Bug #10661]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_2@49509 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 49509 - 02/05/2015 07:30 AM - naruse (Yui NARUSE)

merge revision(s) 49082: [Backport #10661]

```
* parse.y (gettable_gen): disable warnings of possible reference
  to a local variable defined in a past scope.
  [ruby-core:67162] [Bug #10661]
```

History

#1 - 12/27/2014 11:03 AM - nobu (Nobuyoshi Nakada)

Myron Marston wrote:

> Changing the rand call to self.rand is one solution I would consider to avoid the warning, but it doesn't work here because rand is private (as it comes from Kernel), so I'm forced to change the block local variable name to a name I do not want.

You can use rand().

> While I can understand the kinds of errors this warnings helps you avoid, I think that it is too strict and noisy in its current form.

Any idea of less-strict form?

#2 - 12/27/2014 05:46 PM - myronmarston (Myron Marston)

> You can use rand().

Good point. I'm so used to calling arg-less methods without parens that I didn't even think of that here! I'll probably use that rather than renaming variables or let declarations in the 500+ warnings I'm fixing in the RSpec suites.

That said, I've always liked that Ruby supports the same syntax for local variables as for an arg-less message send to self. Avdi's screencast on barewords explains how this can be helpful to be able to refactor from a local to a helper method:

http://devblog.avdi.org/2012/10/01/barewords

This is basically the entire idea of let in RSpec -- when you've got the duplication of creating the same named object in multiple specs as a local variable, you can extract that object definition into a let with the same name and not have to update the referencing syntax. This new warning makes let much less usable for RSpec and Minitest::Spec users who run with warnings enabled, as they may get warnings when doing this refactoring if the same name is used by a local variable earlier in the file (which may be a different context that creates the collaborator object with different parameters or whatever).

> Any idea of less-strict form?

That's a hard one. Given that it is implemented in the parser, I don't see how it could know there is a method def and that it should therefore not warn. I think this issue shows a weakness in Ruby's warning system: it's all or nothing. This warning is helpful in some circumstances, but harmful in others (IMO) since it inhibits bareword refactorings, so it would be nice to be able to opt-out of this warning while still running with the rest of Ruby's warnings enabled. Could the magic comment system (used to set a file's encoding) be used to opt-out of specific warnings?

If that's not feasible, my preference would be to see this removed, as I think being able to refactor between local variables and methods (such as RSpec's and Minitest's let system) is very useful and any warning that inhibits that is, IMO, harmful. Plus, in almost 8 years of doing Ruby, I can't think of a single time I ever hit a situation where this warning would have helped me.

I can completely understand if you're not willing to remove this warning, though!

Thanks for being willing to engage on this so promptly.

#3 - 12/29/2014 11:48 PM - xshay (Xavier Shay)

This is going to warn for most RSpec suites. The following pattern is common (I have it in all of my projects) due to the common let extraction myron talks about:

(sample of a spec from one of my projects - you can also see this in the RSpec commit Myron linked)

```
describe Event do
  describe '#complete!' do
    it 'marks event as complete' do
      event = Fabricate(:event)
      event.complete!
      expect(event.reload.ran?).to eq(true)
    end
  end

  describe '#set_handicap!' do
    let!(:event) { Fabricate(:event, distance: Distance.km(3)) }
    let!(:runs) { [
      Fabricate(:run,
        event:      event
      )
    ]}
    it 'updates a handicap and start time for a run' do
      run = event.set_handicap!(runs[1].competitor_id, 300)

      # ...
    end
  end
end
```

> While I can understand the kinds of errors this warnings helps you avoid

I'm not so convinced on this. The test included with the change already breaks in a fairly obvious way. I haven't seen this class of problem affect any of my newer developers*, and I speculate that the warning might be too technical to be obvious to newer developers anyway - "scope" is a more advanced term.

Put another way: I'd like to see a stronger case made for this change (or any "less strict" version) that justifies breaking such a common pattern.

*yes, this is a small sample!

#4 - 12/30/2014 04:21 AM - myronmarston (Myron Marston)

I'm also seeing this warning from rubygems in the ruby-head builds on travis:

```
/home/travis/.rvm/rubies/ruby-head/lib/ruby/site_ruby/2.3.0/rubygems/compatibility.rb:25: warning: possible re
ference to past scope - path
/home/travis/.rvm/rubies/ruby-head/lib/ruby/site_ruby/2.3.0/rubygems/specification.rb:2480: warning: possible
reference to past scope - name
/home/travis/.rvm/rubies/ruby-head/lib/ruby/site_ruby/2.3.0/rubygems/specification.rb:2482: warning: possible
reference to past scope - name
```

(from https://travis-ci.org/rspec/rspec-support/jobs/45403196)

This shows that it's not just a problem in spec-style suites -- it shows up in normal code that ships with Ruby. In general, when parts of the stdlib emit warnings, it hinders *everyone's* ability to run Ruby with warnings enabled, as stdlib warnings get mixed in with your own, so you train yourself to ignore warning output since it is always present (or, more likely, you turn warnings off). IMO, Ruby should not ship with new types of warnings until the stdlib has been updated to avoid the new warning. Besides ensuring the code ruby ships with is warning-free, it also works as a good vetting tool to show that the situations that the new warning warns about are actually worth having warnings. In this case, it looks like the warning was committed on release day, with no chance for the community to provide feedback on the warning in a preview or release candidate.

#5 - 12/30/2014 05:38 AM - nobu (Nobuyoshi Nakada)

Myron Marston wrote:

> > You can use rand().
> > Any idea of less-strict form?
>
> That's a hard one. Given that it is implemented in the parser, I don't see how it could know there is a method def and that it should therefore not

> warn. I think this issue shows a weakness in Ruby's warning system: it's all or nothing.

Indeed, and far as I remember, some feature requests for it has been proposed.

> This warning is helpful in some circumstances, but harmful in others (IMO) since it inhibits bareword refactorings, so it would be nice to be able to opt-out of this warning while still running with the rest of Ruby's warnings enabled. Could the magic comment system (used to set a file's encoding) be used to opt-out of specific warnings?

Like warn-indent?

Myron Marston wrote:

> I'm also seeing this warning from rubygems in the ruby-head builds on travis:

```
/home/travis/.rvm/rubies/ruby-head/lib/ruby/site_ruby/2.3.0/rubygems/compatibility.rb:25: warning: possibl
e reference to past scope - path
/home/travis/.rvm/rubies/ruby-head/lib/ruby/site_ruby/2.3.0/rubygems/specification.rb:2480: warning: possi
ble reference to past scope - name
/home/travis/.rvm/rubies/ruby-head/lib/ruby/site_ruby/2.3.0/rubygems/specification.rb:2482: warning: possi
ble reference to past scope - name
```

The first one is the bug that found by this warning and fixed at r48985.

#6 - 12/31/2014 07:49 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

- % Done changed from 0 to 100

Applied in changeset r49082.

---

parse.y: disable past scope warnings

- parse.y (gettable_gen): disable warnings of possible reference to a local variable defined in a past scope. [ruby-core:67162] [Bug #10661]

#7 - 12/31/2014 08:04 AM - myronmarston (Myron Marston)

Thank you, nobu (Nobuyoshi Nakada)!

#8 - 01/20/2015 03:04 AM - usa (Usaku NAKAMURA)

- Backport changed from 2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN to 2.0.0: DONTNEED, 2.1: DONTNEED, 2.2: REQUIRED

#9 - 02/05/2015 08:14 AM - naruse (Yui NARUSE)

- Backport changed from 2.0.0: DONTNEED, 2.1: DONTNEED, 2.2: REQUIRED to 2.0.0: DONTNEED, 2.1: DONTNEED, 2.2: DONE

ruby_2_2 r49509 merged revision(s) 49082.

#10 - 12/04/2017 10:38 AM - shyouhei (Shyouhei Urabe)

- Related to Feature #14153: [PATCH] resurrection of # -- warn_past_scope: true -*- added*