Ruby master - Bug #15539

Proc.new with no block shouldn't always warn

01/15/2019 11:04 PM - tenderlovemaking (Aaron Patterson)

Status:	Rejected		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:		**Backport:**	2.4: UNKNOWN, 2.5: UNKNOWN, 2.6: UNKNOWN

Description

Hi,

r66772 introduced a warning for the following code:

```
def foo bar = Proc.new
  bar.call
end

foo       { p "block" } # warn
foo ->() { p "block" } # no warn
```

I don't think this case of Proc.new should warn. To eliminate warnings, I have to rewrite as:

```
def foo bar = nil
  if bar
    bar.call
  else
    yield
  end
end
```

Rails uses this "Proc.new" trick here:

- https://github.com/rails/rails/blob/a08827a90b5a9be79379019cf5b242bd7236d2e3/actionpack/lib/action_controller/metal.rb#L29
- https://github.com/rails/rails/blob/a08827a90b5a9be79379019cf5b242bd7236d2e3/actionpack/lib/action_dispatch/middleware/stack.rb#L100
- https://github.com/rails/rails/blob/a08827a90b5a9be79379019cf5b242bd7236d2e3/activerecord/lib/active_record/statement_cache.rb#L116
- https://github.com/rails/rails/blob/a08827a90b5a9be79379019cf5b242bd7236d2e3/activesupport/lib/active_support/notifications/fanout.rb#L21

I can change Rails, but I want to know why and I don't see any discussion of r66772 (the commit doesn't have a feature number).

Thanks!

Associated revisions

Revision 67519 - 04/13/2019 12:56 AM - marcandre (Marc-Andre Lafortune)

Proc.new: change deprecation warning for clarity (issue #15539)

History

#1 - 01/16/2019 12:25 AM - ko1 (Koichi Sasada)

I'll file the reason soon.

Anyway, I recommend you to rewrite it with a block parameter:

```
def foo bar = nil, &b
  (b || bar).call
end

# or
```

```
def foo bar = nil, &b
  bar ||= b
  bar.call
end
```

#2 - 01/16/2019 01:03 AM - tenderlovemaking (Aaron Patterson)

ko1 (Koichi Sasada) wrote:

> I'll file the reason soon.
>
> Anyway, I recommend you to rewrite it with a block parameter:
>
> ```
> def foo bar = nil, &b
>   (b || bar).call
> end
>
> # or
> def foo bar = nil, &b
>   bar ||= b
>   bar.call
> end
> ```

Ok, no problem! I'll update Rails and reference the feature when posted. □□

#3 - 01/16/2019 01:33 AM - jeremyevans0 (Jeremy Evans)

ko1 (Koichi Sasada) wrote:

> I'll file the reason soon.
>
> Anyway, I recommend you to rewrite it with a block parameter:
>
> ```
> def foo bar = nil, &b
>   (b || bar).call
> end
> ```

I also ran into this issue as Proc.new was used in a few places in Sequel. Removing implicit block support for Proc.new makes it no longer possible to have a default value for a positional argument that uses the block, since the block argument variable is not set until after default values for positional arguments (e.g. https://github.com/jeremyevans/sequel/commit/34faf32c6eddc02d1499748a431f3fdf46bfc5a4#diff-29540b6b617d51d3f5356b1cdfc534b0L61). I feel the loss of functionality is acceptable, since Proc.new's implicit block support was fairly magical and not widely known/used.

#4 - 01/16/2019 10:12 AM - ko1 (Koichi Sasada)

- Status changed from Open to Rejected

> makes it no longer possible to have a default value for a positional argument that uses the block

Surprisingly, we can:

```
def foo o = binding.local_variable_get(:b), &b
  p o.class
end

foo{} #=> Proc
```

(of course it is joke code. it is too implementation dependent code so we shouldn't rely on this behavior)

Anyway, I close this ticket and discuss more on new ticket I'll file.

Thanks,
Koichi

#5 - 01/16/2019 10:20 AM - Eregon (Benoit Daloze)

I always thought Proc.new working without an explicit block was an unintended feature in MRI, and indeed fairly magical, so I support taking a more explicit way in 2.7.