Ruby master - Feature #16241

Shorter syntax for anonymous refinements

10/05/2019 07:04 PM - palkan (Vladimir Dementyev)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

The original discussion is here https://bugs.ruby-lang.org/issues/14344.

tl;dr

- Refinements are often used in-place with anonymous modules
- Having shorter syntax could lower the barrier of entry for Rubyist wanting to explore this feature
- Previous syntax suggestions were not accepted.

I suggest adding a shorter syntax (technically, API):

```
# before
using(Module.new do
  refine Array do
    def foo;"bar";end
  end
end)

# after
refining Array do
  def foo; "bar"; end
end
```

The original idea was to use using_refined instead of refining but after discussing with Matz we decided that it's too verbose, and refining seems better. But is it good enough? Any thoughts?

Related issues:

Related to Ruby master - Feature #14344: refine at class level	**Rejected**
Has duplicate Ruby master - Feature #17336: using refined: do ... end	**Closed**

History

#1 - 10/05/2019 10:05 PM - zverok (Victor Shepelev)

I believe that is really super-good thing. As I've written elsewhere in this tracker, the most useful case for refinements is "in current module, I want some shortcuts for my code to look cleaner". For example, here and here I experimented with representing some graphics algorithm in a most "readable" way, adding really small methods to core (Numeric) and RMagick objects. The point is "here, in this module, I crave for some things to exists in some objects".

So, "inplace refinements" is probably the main kind of refinements the developer could care; while "refinements in a module" could be used, probably, for some "more hygienic" ActiveSupport-alike gems, or some specific blends of Ruby (like using Geometry with ton of specific geometry-related services added to numbers and matrices and whatnot).

#2 - 10/05/2019 10:40 PM - shevegen (Robert A. Heiler)

Hmm. I think there was another suggestion some months ago that also suggested simplifications or "beautification" in regards to refinements.

In general I like ideas that try to make the refinement feature more accessible.

Syntax has been the major reason why I have not been using refinements in my own code; oddly enough, I 100% agree that it is a good idea to have refinements, but the syntax always felt ... clunky to me. That is mostly my reasoning why I think simplifications would be nice.

Matz provided a good syntactic "base". For example, to subclass, we do:

class Foo < Bar

That's very expressive and short.

Refinements on the other hand always felt a bit too verbose to me. That may be a personal choice/preference, granted; on the other hand, looking at suggestions to simplify or beautify it, it does not seem to be a unique opinion as such. :)

(I also have to admit that I once assumed for refinements to work differently, until I read up on the original proposal. I think a few other ruby users may have also mis-interpreted how refinements work now and then. I mention this in particular because I believe matz and the core team would like to have clear definitions for any specifications related to refinements.)

I am still not necessarily in big favour of using the word "refining"; it also seems not as good as the "Foo < Bar" notation :) - but on the other hand, this is, I think, still an improvement over the even more (!) verbose variant that you gave as an example.

By the way, I also completely agree with you here:

> Having shorter syntax could lower the barrier of entry for Rubyist wanting to explore this feature

I think simple/expressive syntax can help a LOT.

I think the only complaint I have about "refining" is that the word seems a bit strange to me ... I do not have a better name as suggestion, though.

By the way, would we then have two words? E. g. "refine" and "refining"? Because this may lead to confusion; may be better to settle just on one either way. (Or perhaps, if that requires modifications and deprecations, to aim for ruby 4.0 with that, but with simplification as the primary goal.)

zverok wrote:

> As I've written elsewhere in this tracker, the most useful case for refinements is "in current module, I want some shortcuts for my code to look cleaner".

Yup, I agree.

> So, "inplace refinements" is probably the main kind of refinements the developer could care; while "refinements in a module"

I think we have to be careful here to not make refinements more complicated or complex, or add more than one refinements in place. This is, I think, what is good about the proposal here in principle, since the primary (or sole) aim appears to be to lead to a lighter syntax, which I think would be good, even if the semantics remain the same in regards to refinements.

If possible, anyone able to come up with other names or syntax could add a comment or two in regards to the word "refining". I personally have not fully made up my mind yet; I am not in favour or disfavour of the word per se. I think the only statement I can make is that it seems to be better than the older/current variant. But perhaps I am overlooking something; may need someone who has used refinements in the past (I can not remember that other issue where someone has suggested a simplification syntax-wise some months ago ...)

#3 - 10/06/2019 07:15 PM - Eregon (Benoit Daloze)

- Related to Feature #14344: refine at class level added

#4 - 11/20/2020 05:00 PM - k0kubun (Takashi Kokubun)

- Has duplicate Feature #17336: using refined: do ... end added

#5 - 11/20/2020 07:22 PM - Eregon (Benoit Daloze)

This sounds fine to me.

I think using do; refine Array do; ...; end; end would be a good way too (https://bugs.ruby-lang.org/issues/17336#note-6), but matz thinks it's confusing

whether the block is for defining refinements or for using refinements, and whether refinements apply after (https://bugs.ruby-lang.org/issues/14344#note-15).