Ruby trunk - Feature #14904

Make it possible to run instance_eval with zero-arguments lambda

07/09/2018 11:32 AM - prijutme4ty (Ilya Vorontsov)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

At a moment #instance_eval always yields an object to a given block. Though if we passed lambda as a block for instance_eval, its work depend on number of arguments. But I see no reason for such strictness when lambda gets zero arguments. In that case instance_eval can just skip yielding receiver to its block.
It will reduce surprise for a newcomers who don't know the difference between proc and lambda.

I got to this problem with code similar to this:

```
module ConfigParams
  def add_config_parameter(param_name, **kwargs, &block)
    # ...
    define_method(param_name){ kwargs[:default].call(self) }
  end
end
class KeyBindings
  extend ConfigParams
  add_config_parameter :undo
  add_config_parameter :redo, default: ->{ "shift+#{undo}" }
end
kb = KeyBindings.new
kb.undo = 'ctrl+z'
puts kb.redo # => shift+ctrl+z
kb.redo = 'ctrl+y'
```

I want to allow user to express defaults with lambda which will be later evaluated in the context of an instance. It's a bit more readable than:

```
add_config_parameter :redo, default: proc{ "shift+#{undo}" }
```

And anyway, it'd be good if a user preferred to change proc into lambda didn't get strange exceptions.

I've already found different solution - to use instance_exec instead of instance_eval. But I'm still sure that life could be easier it instance_eval handled zero-argument lambdas properly.

Related: #10513

History

#1 - 07/09/2018 11:42 AM - prijutme4ty (Ilya Vorontsov)

Sorry, define_method in example should be that:

```
define_method(param_name){ instance_eval(&kwargs[:default]) }
```

#2 - 07/09/2018 06:55 PM - shevegen (Robert A. Heiler)

I have no particular pro or con opinion on the suggestion itself.

I want to comment on one part though:

> I've already found different solution - to use instance_exec instead of instance_eval.
> But I'm still sure that life could be easier it instance_eval handled zero-argument
> lambdas properly.

I can't say whether instance_exec or instance_eval have similar or dissimilar use cases;
I don't think I personally have ever used instance_exec though. But in the event that

this is indeed the case, if there is already a "work-around", then perhaps this may
yield credibility to your wanted use case for instance_eval to also allow for the
same no argument passing as instance_exec would.

But again, I am neutral on this and it may very well be likely that I do not fully
understand the use case (I have to look up what instance_exec does after this :D).

For me, mentally, the different evals are not very easy to remember. I use
instance_eval a lot for class-method aliases, like via:

```
self.instance_eval { alias foo bar }
```

Not sure if there are alternatives to the above but if there are, it is not always
easy to discover them (I mention this just loosely in regards to instance_exec
but I think I am digressing too much from your issue, so I'll stop here.)

#3 - 07/10/2018 01:12 AM - shyouhei (Shyouhei Urabe)

-1. Seems you are already aware of the fact that lambdas are different from procs in handling of the arguments. Lambdas are strict, procs are loose. You are requesting to confuse this distinction. This is not a wise idea. At the beginning there was no lambdas in ruby. The strictness was found useful later, to introduce lambdas. Do not push things back.

If you want this feature only because lambdas are "more readable" in your sense, you would better propose a better syntax of procs.

#4 - 07/11/2018 02:35 AM - prijutme4ty (Ilya Vorontsov)

My use case is just a simple example showing why current behavior isn't as good as it can be.

That's not about lambdas at all. That's about instance_eval being more clever, because there's no reason to yield receiver to a block when block can't handle it. It doesn't make lambda functions weaker, just about correct use of blocks of different arity by instance_eval.

#5 - 07/11/2018 05:03 AM - jeremyevans0 (Jeremy Evans)

prijutme4ty (Ilya Vorontsov) wrote:

> My use case is just a simple example showing why current behavior isn't as good as it can be.
>
> That's not about lambdas at all. That's about instance_eval being more clever, because there's no reason to yield receiver to a block when block can't handle it. It doesn't make lambda functions weaker, just about correct use of blocks of different arity by instance_eval.

One problem here is the slippery slope. There are probably many core methods that take blocks where arguments are yielded, where potentially you could use them with zero argument lambdas. Integer#times comes to mind as the most obvious example. Having all these methods do block arity checking and changing what they yield based on the arity leads to added complexity, and I don't think adding such complexity is a good tradeoff. Especially in this case where you can just use instance_exec.