Ruby - Bug #6151

ArgumentError of lambda shows wrong location

03/16/2012 06:08 AM - trans (Thomas Sawyer)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:	2.0.0		
ruby -v:	ruby 1.9.3p0 (2011-10-30 revision 33570) [x86_64-linux]	**Backport:**	

Description

=begin

Perhaps there is a reason, but it sure seems like a bug to me. Given this simple class:

```
#
class BlockParser < BasicObject
  def initialize(data={}, &block)
    @data = data
    instance_eval(&block)
  end

  def method_missing(name, *args, &block)
    if block
      @data[name] = {}
      BlockParser.new(@data[name], &block)
    else
      case args.size
      when 0
        @data[name]
      when 1
        @data[name] = args.first
      else
        @data[name] = args
      end
    end
  end
end
```

Then we can do:

```
data = {}
BlockParser.new(data) do
  name 'Tommy'
end
data #=> {:name=>'Tommy'}
```

But,

```
data = {}
blk  = lambda do
  name 'Tommy'
end
BlockParser.new(data, &blk)
ArgumentError: wrong number of arguments (1 for 0)
 from (irb):44:in `block in irb_binding'
 from (irb):16:in `instance_eval'
 from (irb):16:in `initialize'
 from (irb):46:in `new'
 from (irb):46
 from /home/trans/.rbfu/rubies/1.9.3-p0/bin/irb:12:in `<main>'
```

If I use ((({Proc.new}))) instead of ((({lambda}))) it works. But since I am using ((({#instance_eval}))) to evaluate the Proc, that shouldn't matter.

Note the reported line number of the error corresponds to ((({name 'Tommy'}))).
=end

Associated revisions

Revision 817eb7d17d3f1327dfb4c706ef6604dd1483ece7 - 03/16/2012 03:00 AM - nobu (Nobuyoshi Nakada)

- vm_insnhelper.c (argument_error): use line number at the beginning
  of lambda, not the first code ob its body.
  [ruby-core:43314][Bug #6151]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@35052 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 817eb7d1 - 03/16/2012 03:00 AM - nobu (Nobuyoshi Nakada)

- vm_insnhelper.c (argument_error): use line number at the beginning
  of lambda, not the first code ob its body.
  [ruby-core:43314][Bug #6151]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@35052 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 03/16/2012 06:13 AM - trans (Thomas Sawyer)

That's fun, go to report a bug in Ruby and hit a bug in Ruby's Bug tracker.

Try again, this time with no RD crap.

Given:

```
#
class BlockParser < BasicObject
  def initialize(data={}, &block)
    @data = data
    instance_eval(&block)
  end

  def method_missing(name, *args, &block)
    if block
      @data[name] = {}
      BlockParser.new(@data[name], &block)
    else
      case args.size
      when 0
        @data[name]
      when 1
        @data[name] = args.first
      else
        @data[name] = args
      end
    end
  end
end

data = {}
blk = lambda do
  name 'tommy'
end

BlockParser.new(data, &blk)
ArgumentError: wrong number of arguments (1 for 0)
```

But it works if we use:

```
blk = Proc.new do
  name 'tommy'
end
```

Since BlockParser is using instance_eval, I don't see why this should be an error. Indeed, how can I depend on it if my API simply accepts a block --I don't know if it's a Proc or a lambda.

#2 - 03/16/2012 06:56 AM - nobu (Nobuyoshi Nakada)

- Description updated

#3 - 03/16/2012 11:59 AM - nobu (Nobuyoshi Nakada)

- Subject changed from BasicObject instance_eval of lambda causes errors to ArgumentError of lambda shows wrong location

- Target version set to 2.0.0

You need to indent with spaces, before tabs.

#4 - 03/16/2012 12:30 PM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

- % Done changed from 0 to 100

fixed in r35051-r35053.

#5 - 03/16/2012 12:42 PM - john_firebaugh (John Firebaugh)

The reason it raises an error is that instance_eval yields one argument (the receiver), and lambdas are strict about arity -- see #2476.

#6 - 03/17/2012 03:16 AM - trans (Thomas Sawyer)

Just to be clear, where both issues fixed? i.e. the fact that an error was raised in this case at all, as well as the location of lambda errors?

#7 - 03/17/2012 03:18 AM - john_firebaugh (John Firebaugh)

Just the location was fixed. The error is "by design" -- as you discovered, you will need to use Proc.new (or lambda {|*| ... }).

#8 - 03/17/2012 04:24 AM - trans (Thomas Sawyer)

How can that be by design? What kind of design criteria says that "instance_eval should blow up if a lambda is passed to it"?

So you are saying that instance_eval passes the receiver into the Proc as an argument? Why would it do this when the Block has zero arity? Why not just check the arity of the block and pass it if it can take an argument otherwise not?

This snafu is really ashame for me too, b/c I finally had a good reason to use Ruby 1.9's -> operator!

#9 - 03/06/2013 02:49 AM - TylerRick (Tyler Rick)

I agree, @trans, this is a very surprising behavior. I was expecting instance_eval to call the block I passed to it without any args. Since self is already implicitly available from the block, I just don't understand why instance_eval would yield self as an argument.

Fortunately, I think instance_exec http://ruby-doc.org/core-2.0/BasicObject.html#method-i-instance_exec does what we are wanting so I'm using it instead.

We should update the documentation. http://ruby-doc.org/core-2.0/BasicObject.html#method-i-instance_eval does not mention that it yields self as the first argument.