Ruby master - Feature #1082
add Object#singleton_class method

02/01/2009 05:58 AM - sunaku (Suraj Kurapati)

Status: Closed
Priority: Normal
Assignee: matz (Yukihiro Matsumoto)
Target version: 1.9.2

Description
=begin
Please add Object#singleton_class method to the Ruby API.
This method would allow us to write:

hello.singleton_class

Instead of always having to write:

(class << hello; self; end)

Furthermore, the name "singleton_class" should be used to minimize disruption to the existing Ruby API, which already uses the name "singleton":

Object.instance_methods.grep(/singleton/)  
=> [:singleton_methods, :define_singleton_method]

RUBY_DESCRIPTION  
=> "ruby 1.9.1p0 (2009-01-20 revision 21700) [i686-linux]"

Thanks for your consideration.
=end

History
#1 - 02/01/2009 07:00 AM - radarek (Radosław Bułat)

=begin
+1
"Singleton" is most popular name for describe... singleton classes (sometimes named eigenclass, virtual class, metaclass). To be consistent with singleton_methods, define_singleton_method it would be very nice to name it singleton_class.
=end

#2 - 02/01/2009 10:02 AM - murphy (Kornelius Kalnbach)

=begin
+1
I thought this feature was already added to 1.9? class << self; end is not very helpful for beginners.
What about singleton_class_eval? (Or singleton_class.eval, for that matter...)
=end

#3 - 02/01/2009 08:31 PM - dblack (David Black)

=begin
Hi --

On Sun, 1 Feb 2009, Kornelius Kalnbach wrote:

Issue #1082 has been updated by Kornelius Kalnbach.

+1
I thought this feature was already added to 1.9? class << self; end is not very helpful for beginners.
What about singleton_class_eval? (Or singleton_class.eval, for that matter...)

singleton_class.class_eval. The less it's treated like a special case, the better.

David

--
David A. Black / Ruby Power and Light, LLC
Ruby/Rails consulting & training: http://www.rubypal.com
Coming in 2009: The Well-Grounded Rubyist (http://manning.com/black2)
http://www.wishsight.com => Independent, social wishlist management!

#4 - 02/02/2009 12:01 PM - ko1 (Koichi Sasada)
- Assignee set to matz (Yukihiro Matsumoto)

#5 - 01/03/2010 04:53 AM - sunaku (Suraj Kurapati)

#6 - 01/04/2010 03:25 PM - matz (Yukihiro Matsumoto)

#7 - 01/04/2010 03:46 PM - murphy (Kornelius Kalnbach)

03/13/2022
I'd like to see it included in future Ruby versions.

[murphy]
=end

#8 - 01/05/2010 01:14 AM - judofyr (Magnus Holm)
=begin
+1 for whatever. (Okay then, maybe singleton_class is better)

//Magnus Holm

On Mon, Jan 4, 2010 at 07:25, Yukihiro Matsumoto matz@ruby-lang.org wrote:

Hi,

In message "Re: [ruby-core:27372] [Feature #1082] add Object#singleton_class method" on Sun, 3 Jan 2010 04:53:36 +0900, Suraj Kurapati redmine@ruby-lang.org writes:

|Any thoughts on this feature request?
| hello.singleton_class == (class << hello; self; end)
|Thanks for your consideration.

We haven't met any consensus of a name for the method. singleton_class, singletonclass, eigenclass, or whatever.

matz.

=end

#9 - 01/05/2010 02:06 AM - mame (Yusuke Endoh)
=begin
Hi,

2010/1/4 Yukihiro Matsumoto matz@ruby-lang.org:

We haven't met any consensus of a name for the method. singleton_class, singletonclass, eigenclass, or whatever.

The draft of ruby-std uses eigenclass. (13.4 Eigenclass)

--
Yusuke ENDOH mame@tsg.ne.jp

=end

#10 - 01/05/2010 03:52 AM - murphy (Kornelius Kalnbach)
=begin
On 04.01.10 17:28, Haase, Konstantin wrote:

+1 for metaclass, as is it compatible with ActiveSupport, Rubinius and _why.

The current methods are all named singleton_*, so anything else than singleton_class would be inconsistent.

ActiveSupport can (and will) still implement metaclass. And I didn't know Rubinius was having its own Ruby language. Compatibility with _why's metaid can be installed as a gem.

Also, it confuses people with a Python background. Is that a plus? A metaclass are just the class of a class. I think _why didn't pick the best name.

"Eigenclass" still sounds like a mathematician's joke to me ;)

03/13/2022
On Mon, Jan 4, 2010 at 10:52 AM, Kornelius Kalnbach murphy@rubychan.de wrote:

On 04.01.10 17:28, Haase, Konstantin wrote:

+1 for metaclass, as is it compatible with ActiveSupport, Rubinius and _why.

The current methods are all named singleton_*, so anything else than singleton_class would be inconsistent.

Agreed.

It's an appropriate name that makes its role in Ruby easier to understand.

"Meta" and "eigen" are less concrete and make Ruby seem more like arcane spellbook.

ActiveSupport can (and will) still implement metaclass. And I didn't know Rubinius was having its own Ruby language. Compatibility with _why's metaid can be installed as a gem.

Active Support always prefers Ruby's choice and will gladly change.

Best,
jeremy

---

Hi,

In message "Re: [ruby-core:27394] Re: [Feature #1082] add Object#singleton_class method" on Tue, 5 Jan 2010 02:05:19 +0900, Yusuke ENDOH mame@tsq.ne.jp writes:

| > We haven't met any consensus of a name for the method.
| > singleton_class, singletonclass, eigenclass, or whatever.
| |
| > The draft of ruby-std uses eigenclass. (13.4 Eigenclass)

Of course I know. I was right there when we made the decision.

matz

---

Hi,

In message "Re: [ruby-core:27385] Re: [Feature #1082] add Object#singleton_class method" on Mon, 4 Jan 2010 15:46:32 +0900, Kornelius Kalnbach murphy@rubychan.de writes:

_why suggested "metaclass" back in 2005. But since we already have #define_singleton_method and #singleton_methods, "singleton_class" should be the name of it.

"metaclass" means "a class of a class" but "that something" is not really a metaclass, since it's not only for classes, but for any object. "singleton_class" is too easily confused with a class appears in the Singleton Pattern.
I'd like to see it included in future Ruby versions.

Me too. But I will wait for the right word.

matz.

#14 - 01/05/2010 11:26 AM - shugo (Shugo Maeda)

Hi,

2010/1/5 Yusuke ENDOH mame@tsg.ne.jp:

We haven't met any consensus of a name for the method. singleton_class, singletonclass, eigenclass, or whatever.

The draft of ruby-std uses eigenclass. (13.4 Eigenclass)

Yes, it does. However, we can choose another word.

I prefer singleton_class than eigenclass. I suppose that "singleton class" is not so confusing with a class which implements the Singleton pattern because the word singleton remind Ruby users of singleton methods rather than the Singleton pattern.

--
Shugo Maeda

#15 - 01/05/2010 11:41 AM - matz (Yukihiro Matsumoto)

Hi,

In message "Re: [ruby-core:27407] Re: [Feature #1082] add Object#singleton_class method" on Tue, 5 Jan 2010 11:26:41 +0900, Shugo Maeda shugo@ruby-lang.org writes:

I prefer singleton_class than eigenclass. I suppose that "singleton_class" is not so confusing with a class which implements the Singleton pattern because the word singleton remind Ruby users of singleton methods rather than the Singleton pattern.

Basically, I agree. But I cannot deny the fact that we already have singleton.rb in the distribution, which provide Singleton class. I confess I did. Probably I shouldn't have.

matz.

#16 - 01/05/2010 11:56 AM - austin (Austin Ziegler)

On Mon, Jan 4, 2010 at 9:41 PM, Yukihiro Matsumoto matz@ruby-lang.org wrote:

In message "Re: [ruby-core:27407] Re: [Feature #1082] add Object#singleton_class method" on Tue, 5 Jan 2010 11:26:41 +0900, Shugo Maeda shugo@ruby-lang.org writes:

I prefer singleton_class than eigenclass. I suppose that "singleton_class" is not so confusing with a class which implements the Singleton pattern because the word singleton remind Ruby users of singleton methods rather than the Singleton pattern.

Basically, I agree. But I cannot deny the fact that we already have singleton.rb in the distribution, which provide Singleton class. I confess I did. Probably I shouldn't have.

I just did a quick check of 1.8's singleton.rb, and Singleton is a module, not a class (which I'm sure you really knew, but it's important to my point).
I think that it's easy enough to say that every object can have a singleton_class, which sits between the object and any of its ancestors; not every object, however, implements the Singleton Pattern through the Singleton module.

That said, taking my interpretation above (that the {singleton,eigen}class sits between the object and its ancestors, if it's correct) a quick thesaurus check suggests either:

- betwixt_class
- inter_class (as in "international", "existing or occurring between classes")

betwixt is more fun, but the idea of calling it an interclass makes a lot of sense to me.

-austin

Austin Ziegler • halostatue@gmail.com • austin@halostatue.ca
http://www.halostatue.ca/ • http://twitter.com/halostatue

#17 - 01/05/2010 12:24 PM - bitsweat (Jeremy Daer)

On Mon, Jan 4, 2010 at 6:41 PM, Yukihiro Matsumoto matz@ruby-lang.org wrote:

Hi,

In message "Re: [ruby-core:27407] Re: [Feature #1082] add Object#singleton_class method" on Tue, 5 Jan 2010 11:26:41 +0900, Shugo Maeda shugo@ruby-lang.org writes:

|I prefer singleton_class than eigenclass. I suppose that "singleton class" is not so confusing with a class which implements the Singleton pattern because the word singleton remind Ruby users of singleton methods rather than the Singleton pattern. Basically, I agree. But I cannot deny the fact that we already have singleton.rb in the distribution, which provide Singleton class. I confess I did. Probably I shouldn't have.

That can't be denied, but I think this confusion is mostly hypothetical.

I sympathize with your aesthetic concerns with collision, but the term "singleton class" has been in common usage for years and has proven understandable to new Ruby users crossing the "class << self" chasm for the first time.

It's a good name that has stood the test of time. Let's officially adopt what we've already unofficially chosen.

jeremy

#18 - 01/05/2010 08:13 PM - murphy (Kornelius Kalnbach)

why not add both? eigenclass could be an alias for singleton_class. [murphy]

#19 - 01/05/2010 09:31 PM - Eregon (Benoit Daloze)

Hi,

Surely, seeing:

obj.bad_name_class
is preferable to:

class << obj; self end

and we can do even better than that with singleton_class. :)

James Edward Gray II

+1 for eigenclass:

I think something is confusing for me between

(1) class << obj
  self
end

and

(2) class << MyClass
  def method # this is a class method, a method that MyClass-Object owns, alone (without needing to make an instance to use)
  end
end

Here, I feel far less confusing the eigenclass.
Well, the syntax we are trying to name, is describing a class who allow us to write methods that concern the object globally(1), as a single object(2).

Eigenclass means then the "proper" (~ global, single) class to that Object, a class who concerns only one Object, but that Object doesn't look like a singleton especially, it's just a single object. We didn't build it with a class describing it (or that doesn't matter), we are just speaking to it directly, ignoring what's around.

For me, singleton is confusing, and wrong, as it has a name of "created and it's the only one of a 'class' ", while that 'class' means nothing here.

"eigenclass" seem, according to me, the only alternative in the 3 proposed to name both the 2 situations above.
(While I think we'll still use a lot of class << MyClass; def m(); end; end, that is kind of very useful)

"singleton_class" is already used in Ruby, yes, but not very wide, and it can still change if needed:
$> ri singleton
1 Numeric#singleton_method_added
2 Object#define_singleton_method
3 Object#singleton_methods
4 RDoc::Generator::Method#singleton
(I must admit a singleton method SEEMS more clear here than an eigen method..., but it would not be obfuscating neither I think)

And the last reason: eigenclass is original, sounds cool, is not so confusing and unique (like Ruby) !

Here was my piece of advice to name this.

Benoit Daloze

Hi,2010/1/5 James Edward Gray II <james@graysoftinc.com>
Surely, seeing:

obj.bad_name_class

is preferable to:

class << obj; self end

and we can do even better than that with singleton_class. :)

James Edward Gray II

+1 for eigenclass:
I think something is confusing for me between

(1) class << obj
  self
end

and

(2) class << MyClass
  def method # this is a class method, a method that MyClass-Object owns, alone (without needing to make an instance to use)
end

Here, I feel far less confusing the eigenclass. Well, the syntax we are trying to name is describing a class who allow us to write methods that concern the object globally(1), as a single object(2).

Eigenclass means then the "proper"(~ global, single) class to that Object, a class who concerns only one Object, but that Object doesn't look like a singleton especially, it's just a single object. We didn't build it with a class describing it (or that doesn't matter), we are just speaking to it directly, ignoring what's around.

For me, singleton is confusing, and wrong, as it has a name of "created and it's the only one of a 'class'", while that 'class' means nothing here."eigenclass" seem, according to me, the only alternative in the 3 proposed to name both the 2 situations above.

(While I think we'll still use a lot of class << MyClass; def m(); end; end, that is kind of very useful)"singleton_class" is already used in Ruby, yes, but not very wide, and it can still change if needed:

$> ri singleton
1 Numeric#singleton_method_added 2 Object#define_singleton_method 3 Object#singleton_methods 4
RDoc::Generator::Method#singleton

I must admit a singleton method SEEMS more clear here than a eigen method..., but it would not be obfuscating neither I think)

And the last reason: eigenclass is original, sounds cool, is not so confusing and unique (like Ruby) !Here was my piece of advice to name this.

Benoit Daloze

#20 - 01/06/2010 07:25 AM - ammar (Ammar Ali)

begin
Suggesting new terms might be completely unwelcome, but...

The distinction between a "regular" class and a singleton/eigen/whatever class appears similar to the distinction between phonetic and phonemic in phonology. Some shorten these terms and broaden their meaning; "emic" (loosley meaning below the surface) and "etic" (loosely meaning on the surface). Would emic-class make sense to others?

Also from linguistics, the prefix "allo-" generally means "differing from the normal"... so I wonder if "alloclass" is a good possible name.

Just a thought,

ammar

end

#21 - 01/06/2010 08:01 AM - RickDeNatale (Rick DeNatale)

begin
On Tue, Jan 5, 2010 at 5:25 PM, Ammar Ali redmine@ruby-lang.org wrote:

Issue #1082 has been updated by Ammar Ali.

Suggesting new terms might be completely unwelcome, but...

The distinction between a "regular" class and a singleton/eigen/whatever class appears similar to the distinction between phonetic and phonemic in phonology. Some shorten these terms and broaden their meaning; "emic" (loosley meaning below the surface) and "etic" (loosely meaning on the surface). Would emic-class make sense to others?

Also from linguistics, the prefix "allo-" generally means "differing from the normal"... so I wonder if "alloclass" is a good possible name.

In the edition of the Pickaxe corresponding to Ruby 1.8. Dave Thomas used the term Virtual class, which never seems to get mentioned in these discussions.

Not that I think that that's the right term either, in fact, it's really the opposite of the characteristic which I think the 'classes' which provide singleton methods, and the 'classes' which provide class methods (which IMHO really aren't singleton methods because they apply to subclasses as well) share.

There's an old CS joke.

If it's there and you can see it, it's real,
If it's not there and you can see it, it's virtual
If it's there and you can't see it, it's transparent,
If it's not there and you can't see it it's gone.

I would suggest transparent_class for singleton classes whether they are singleton classes of regular objects (which really do provide methods ONLY for one object) or 'singleton' classes of Class objects. The key aspect is that the Ruby implementation does its best to hide these from the Ruby programmer.

Or perhaps, since in a kind of heisenberg effect, providing an official method to make them visible means that they wouldn't really be totally transparent, perhaps hidden_class, and for transparent/hidden classes which are the (meta?)class of a class, a
hidden superclass method, so that

String.hidden_class.hidden_supersclass would return Object.hidden_class

--
Rick DeNatale

Blog: http://talklikeaduck.denhaven2.com/
Twitter: http://twitter.com/RickDeNatale
WWR: http://www.workingwithrails.com/person/9021-rick-denatale
LinkedIn: http://www.linkedin.com/in/rickdenatale

#22 - 01/06/2010 10:57 AM - dblack (David Black)

=begin
Hi --
On Tue, 5 Jan 2010, Yukihiro Matsumoto wrote:

   Hi,

   In message "Re: [ruby-core:27407] Re: [Feature #1082] add Object#singleton_class method" on Tue, 5 Jan 2010 11:26:41 +0900, Shugo Maeda shugo@ruby-lang.org writes:

   |I prefer singleton_class than eigenclass. I suppose that "singleton class" is not so confusing with a class which implements the Singleton pattern because the word singleton remind Ruby users of singleton methods rather than the Singleton pattern.

   Basically, I agree. But I cannot deny the fact that we already have singleton.rb in the distribution, which provide Singleton class. I confess I did. Probably I shouldn't have.

I've heard lots of people say that the similarity of names presents a problem, but I've seen no evidence that there's actually a problem. I've only very, very rarely heard anyone express confusion, and the confusion only lasts for about two seconds until the difference is explained:

   Student: Is that related to the Singleton pattern?
   Me: No, "singleton" here has to do with object-specific behavior in Ruby's object model. It's a different concept that happens to have a similar-sounding name.
   Student: Oh, OK.

And that's all there is to it (if even that much).

David

--
David A. Black
Senior Developer, Cyrus Innovation Inc.
THE COMPLEAT RUBYIST, Ruby training with Black/Brown/McAnally!
January 22-23, Tampa, Florida
Info and registration at http://www.thecompleatrubyist.com

=end

#23 - 02/23/2010 04:14 PM - shugo (Shugo Maeda)

=begin
Hi,

2010/1/5 Yukihiro Matsumoto matz@ruby-lang.org:

   Basically, I agree. But I cannot deny the fact that we already have singleton.rb in the distribution, which provide Singleton class. I confess I did. Probably I shouldn't have.

Today, I talked with Matz, and he has finally agreed with the use of the term "singleton class." The reason why he has chosen the term

03/13/2022
"singleton class" is that we have already had Kernel#singleton_methods.

Then, the next issue is whether Kernel#singleton_class should create an eigenclass when the receiver has no eigenclass. It would be better to have an optional argument (singleton_class(true/false) or singleton_class(:create => true)?) to switch the behavior.

Any thoughts?
--
Shugo Maeda

#24 - 02/23/2010 04:46 PM - murphy (Kornelius Kalnbach)

On 23.02.10 08:14, Shugo Maeda wrote:

Today, I talked with Matz, and he has finally agreed with the use of the term "singleton class." The reason why he has chosen the term "singleton class" is that we have already had Kernel#singleton_methods. Yay! \o/

Then, the next issue is whether Kernel#singleton_class should create an eigenclass when the receiver has no eigenclass. It would be better to have an optional argument (singleton_class(true/false) or singleton_class(:create => true)?) to switch the behavior. I always perceived singleton classes to be transparently created in the background. Would it be wise to give the programmer access to its existence status? (I assume singleton_class(false) == nil would behave like a singleton_class? method.) Today, Ruby has no way of telling whether an object already has a singleton class, or am I wrong?

Also, why would I ask for a singleton_class without using it?

[murphy]

#25 - 02/23/2010 11:26 PM - shugo (Shugo Maeda)

Hi,

2010/2/23 Kornelius Kalnbach murphy@rubychan.de:

Then, the next issue is whether Kernel#singleton_class should create an eigenclass when the receiver has no eigenclass. It would be better to have an optional argument (singleton_class(true/false) or singleton_class(:create => true)?) to switch the behavior. I always perceived singleton classes to be transparently created in the background. Would it be wise to give the programmer access to its existence status? (I assume singleton_class(false) == nil would behave like a singleton_class? method.) Today, Ruby has no way of telling whether an object already has a singleton class, or am I wrong?

You are right, and it is why Matz wondered whether singleton_class should create singleton classes. If it always creates singleton classes, obj.singleton_class is just a short form of class << obj; self, end. He said that it may be worth considering whether singleton_class should provide an (optional) way to tell the existence of a singleton class.

I think transparent creation of singleton classes is reasonable, but it sometimes annoys me when I'm investigating the behavior of Ruby. However, I don't come up with such a situation in a real-world application.

--
Hi,

2010/2/23 Shugo Maeda shugo@ruby-lang.org:

Then, how about to introduce singleton_class as just a short form of class <<obj; self; end at first? We can add another method or an optional argument of singleton_class later.

singleton_class?

x.singleton_class? sounds like "Is x a singleton class?", doesn't it?

--
Shugo Maeda

I have attached a patch with a test.

--
Shugo Maeda

Attachment: singleton_class.diff

Hi, 

2010/2/23 Shugo Maeda shugo@ruby-lang.org:

Then, how about to introduce singleton_class as just a short form of class <<obj; self; end at first?

I have attached a patch with a test.

--
Shugo Maeda

I think the idea that some objects don't have a singleton class is an internal detail that is, as yet, not part of Ruby's semantics. I don't think we should add it to Ruby's semantics.

Any update here? Why don't you commit it? Does Matz still hate "just a short form"?

--

#27 - 02/26/2010 12:08 AM - shugo (Shugo Maeda)

#28 - 02/26/2010 03:54 AM - wycats (Yehuda Katz)

#29 - 03/17/2010 11:05 PM - mame (Yusuke Endoh)

03/13/2022
Hi,

2010/3/17 Yusuke Endoh redmine@ruby-lang.org:

Then, how about to introduce singleton_class as just a short form of class << obj; self; end at first?

I have attached a patch with a test.

Any update here? Why don't you commit it?

I'm not sure whether it should be included in Ruby 1.9.2 or not. What do you think of it, Yugui?

Does Matz still hate "just a short form"?

I don't think so.
Do you accept the patch, Matz?

--
Shugo Maeda

#31 - 03/19/2010 07:01 PM - yugui (Yuki Sonoda)

On Thu, Mar 18, 2010 at 3:22 PM, Shugo Maeda shugo@ruby-lang.org wrote:

I'm not sure whether it should be included in Ruby 1.9.2 or not. What do you think of it, Yugui?

You do not seem to have reached an agreement. We need more discussion.

-- Yuki Sonoda (Yugui)

#32 - 03/19/2010 07:20 PM - mame (Yusuke Endoh)

Hi,

2010/3/19 Yugui yugui@yugui.jp:

On Thu, Mar 18, 2010 at 3:22 PM, Shugo Maeda shugo@ruby-lang.org wrote:

I'm not sure whether it should be included in Ruby 1.9.2 or not. What do you think of it, Yugui?

You do not seem to have reached an agreement. We need more discussion.

What should we discuss?

I think that there is no one who disagree with singleton_class as "a short form of class << obj; self; end", except matz :-) And, I thought matz was likely to accept "a short form" because Shugo finally sent a patch for it. [ruby-core:28338]

--
Yusuke ENDOH mame@tsg.ne.jp
2010/3/19 Yugui yugui@yugui.jp:

On Thu, Mar 18, 2010 at 3:22 PM, Shugo Maeda shugo@ruby-lang.org wrote:

I'm not sure whether it should be included in Ruby 1.9.2 or not. What do you think of it, Yugui?

You do not seem to have reached an agreement. We need more discussion.

My proposal is to introduce Object#singleton_class as just a short form of class =<<obj; self; end. Any objections?

It's OK for me to postpone it until Ruby 1.9.3 because users can easily implement it on their own. However, I think it's important to agree with the method name.

--
Shugo Maeda

2010/3/19 Yuki Sonoda yugui@yugui.jp:

On Fri, Mar 19, 2010 at 7:20 PM, Yusuke ENDOH mame@tsg.ne.jp wrote:

I think that there is no one who disagree with singleton_class as "a short form of class =<<obj; self; end", except matz :-)

And, I thought matz was likely to accept "a short form" because Shugo finally sent a patch for it. [ruby-core:28338]

I see. There is no problem to add it to Ruby 1.9.2 if there is an agreement as the result of the discussion.

-- Yuki Sonoda (Yugui)

2010/3/22 Yukihiro Matsumoto matz@ruby-lang.org:

In message "Re: [ruby-core:28731] Re: [Feature #1082] add Object#singleton_class method" on Thu, 18 Mar 2010 15:22:23 +0900, Shugo Maeda shugo@ruby-lang.org writes:

|> Does Matz still hate "just a short form"?
| | I don't think so.
|> Do you accept the patch, Matz?

I accept.

matz.

2010/3/22 Yukihiro Matsumoto matz@ruby-lang.org:
> Does Matz still hate "just a short form"?
| I don't think so.
| Do you accept the patch, Matz?
| I accept.

I have committed it to the SVN trunk. Thank you.

--
Shugo Maeda
=end

#37 - 03/23/2010 06:47 PM - shugo (Shugo Maeda)
- Status changed from Open to Closed
- % Done changed from 0 to 100

=begin
This issue was solved with changeset r27022.
Suraj, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.
=end