Ruby master - Feature #7377

#indetical? as an alias for #equal?

11/17/2012 09:47 AM - aef (Alexander E. Fischer)

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

Description

As my feature request #7359 got rejected, here a more backward-compatible approach:

In my opinion the difference between #eql? and #equal? is really unintuitive. How about making their difference more obvious by giving one of them a more accurate name?

My proposal is to alias #equal? to #identical?.

I'll write a patch, if this is acceptable.

History

#1 - 11/20/2012 07:03 AM - Anonymous

+1, due to similar sounding name, these methods confuse beginners (me). #identical? feels much better.

#2 - 11/20/2012 02:22 PM - marcandre (Marc-Andre Lafortune)

identical? can also mean "exactly alike", e.g. identical twins.

For the price of two additional letters, I'd recommend same_object? instead.

#3 - 11/21/2012 05:09 AM - aef (Alexander E. Fischer)

#same_object? would also be ok by me.

#4 - 11/21/2012 05:40 AM - trans (Thomas Sawyer)

=begin
The reason for the word "identical" is b/c of the root "id" which corresponds to the fact that the id's are the same. They would not be identical if the id's were not the same. So yes, "exactly alike" is precise. Only the identical object is exactly like itself. Remember too we are comparing two references, not the objects themselves. We are really asking if reference x is identical to reference y.

Also, the #same_object? seems redundant b/c everything is an object. So if an alternate is to be suggested, I would think it should just be (({#same?})). It would suffice, but does just "same?" really have such a precise meaning?
=end

#5 - 11/21/2012 01:20 PM - marcandre (Marc-Andre Lafortune)

trans (Thomas Sawyer) wrote:

> =begin
> The reason for the word "identical" is b/c of the root "id" which corresponds to the fact that the id's are the same. They would not be identical if the id's were not the same.

Identical twins are not the same person (i.e. not the same object), and calling SomeActiveRecordModel.first twice will not give you the same object even if they share the same id and values.

> Also, the #same_object? seems redundant b/c everything is an object. So if an alternate is to be suggested, I would think it should just be (({#same?})). It would suffice, but does just "same?" really have such a precise meaning?

You clearly have never been to Thaïland ;-)

'object' is not redundant, since one can compare sameness in different ways, by value being the most popular.

See also https://bugs.ruby-lang.org/issues/6367 for another proposed use of same?

#6 - 11/21/2012 05:42 PM - Anonymous

Marc, sorry to break it to you, but "identical twins" is misuse of the word identical, something like "dictatorial democracy" :)

#7 - 11/21/2012 10:12 PM - trans (Thomas Sawyer)

"and calling SomeActiveRecordModel.first twice will not give you the same object even if they share the same id"

That's not the id we mean. In Ruby every object has a unique id.

```
x = "foo"
x.object_id  #=> 17086140
```

So to ask x.identical? y is the to ask x.object_id == y.object_id. And that's how it works under the hood. Which is why I think a term the indicates the "id" aspect of this is nicer and really fortunate that we even have available to us. I'm not saying #same_object? is an awful choice. It would still be better than the current confusion. But #identical? gives us a little bit stronger semantics.

#8 - 11/24/2012 06:22 AM - drbrain (Eric Hodel)

Boris, sorry to break it to you, but "identical" means both "similar in every detail" (so not a misuse as in "identical twins") and also "expressing a [mathematical] identity" according to the Oxford English Dictionary.

If we already have a convention and need to debate the meanings of a proposed new convention it is probably best if we just drop the discussion.

By carrying on with the existing conventions we reduce cognitive load on rubyists. New rubyists learning by themselves don't have to wonder "why can I do this two ways?" and "which way is the recommended way?". People who teach ruby won't need to tell new rubyists "we have two ways to do this because of a long discussion that would distract from the current lesson". Long-time rubyists won't need to wonder "Why was a second way of doing this added? How does it make my life better?"

The pursuit of a perfect language is noble, but there are always going to be rough edges. Invalidating years of working code and years of blog posts and instruction to rearrange a few methods to be more "perfect" is probably not the best use of our time.

#9 - 11/24/2012 10:51 AM - Anonymous

Eric, your post could be shortened to 10% without loss of meaning: You give +1 to stability. I give +1 to having a synonym #identical? for #equal?

#10 - 11/24/2012 12:44 PM - mame (Yusuke Endoh)

- Target version set to 2.6

#11 - 11/24/2012 01:26 PM - trans (Thomas Sawyer)

This is nothing unusual or difficult about an alias. Ruby has a number of them and it hasn't turned any nuby minds to mush. On the contrary, it really provides a good way for the language to mature naturally. Case in point, early on I used Enumerable#collect, others did as well. But now I only use #map. I've notice almost everyone uses #map now too. #collect is still around, and it probably will never leave us, and despite having two methods for the same thing I don't hear anyone complaining that it should. I for one am thankful we've had the choice (if it were up for debate back in the day I would have voted for #collect and been poorer for it).

The pursuit of a *better* language is noble, and thus should be pursued. After all, that's why we are all here. Isn't it? There is no invalidation of code and years of blog posts by adding an alias in any case. Besides that, other features of the language have changed much more drastically over the years in pursuit of the better language (the transition for #id to #object_id and the coming deprecation of #autoload come to mind). Asking for a simple alias because it conceptually makes more sense and helps to differentiate the more confusing question of how #equal? differs from #eql? is a feather by comparison --and it's how it makes our life better.

#12 - 11/24/2012 01:35 PM - trans (Thomas Sawyer)

boris (Boris Ruf)

> You give +1 to stability. I give +1 to having a synonym #identical? for #equal?

Nothing to do with stability. A bit more like +1 to defeatism.

#13 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)