Ruby - Bug #21322

Namespaces and builtin classes as arguments and return values

05/10/2025 09:12 AM - fxn (Xavier Noria)

Status:	Open		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:		**Backport:**	3.2: UNKNOWN, 3.3: UNKNOWN, 3.4: UNKNOWN

Description

@tagomoris (Satoshi Tagomori) thanks for the docs under doc/namespace.md.

Unless I missed it, I believe there is an edge case related to builtin classes (using the vocabulary there). Consider:

```
# test.rb
ns = Namespace.new
ns.require_relative 'foo'

X = 1
ns::C.x(Object)

# foo.rb
class C
  def self.x(obj)
    obj::X
  end
end
```

obj::X raises. I believe this is consistent with the feature, but maybe would deserve docs, because from the point of view of the Ruby programmer I am passing an object, no constant name resolution is happening in foo.rb. See what I mean?

I believe, from my tests, that something analogous happens if ns::C.x returns (the namespaced) Object. In the main namespace, you don't get the object passed up as-is.

I am also curious about how is this implemented (maybe to comment here, not necessarily in the docs).

Related issues:

Related to Ruby - Bug #21317: Namespaces leak with object IDs	**Closed**

History

#1 - 05/10/2025 09:40 AM - fxn (Xavier Noria)

> I believe this is consistent with the feature

I say so in the sense that objects with the same ID are equal in the namespace.

However, it could be argued that it is not consistent with Ruby.

Object is a constant that stores a class object, just like after X = 1 the constant X stores an integer.

So, I am doing a method call passing an object reference, and the method is not receiving the same thing.

So, there is a tension between the feature and this basic aspect of the language, in my view.

#2 - 05/10/2025 09:50 AM - fxn (Xavier Noria)

For example, if I instead do

```
ns = Namespace.new
ns.require_relative 'foo'
```

```
c = Class.new
c::X = 1
ns::C.x(c)
```

then obj::X resolves as expected in the method.

Method calls somehow seem to be swapping VALUEs that correspond to builtin classes on the fly.

Please excuse I am not reading the source code to answer this myself, I do not know enough about CRuby internals to understand the patch and its implications.

#3 - 05/10/2025 12:20 PM - Eregon (Benoit Daloze)

Right, IOW it's the duality between builtin classes and other classes.
Builtin classes' constants, methods, ivar and cvars are all a copy per Namespace (and so potentially differ), even though for all builtin classes they are equal? with the corresponding builtin class in all namespaces.

I think that's part of what makes the semantics of Namespace complicated, but as I understand it's also part of the design of Namespace.

#4 - 05/10/2025 12:22 PM - Eregon (Benoit Daloze)

- Related to Bug #21317: Namespaces leak with object IDs added

#5 - 05/11/2025 01:01 PM - fxn (Xavier Noria)

@Eregon (Benoit Daloze) maybe, but the fact is that you pass an object, and the callee receives a different object. This is not consistent with the current language.

Ideas:

- Introduce the concept of "namespace-level root objects", documented to be swapped on the fly or something. Sounds too ad-hoc to me.
- Restricting method calls to primitive types (not a good one, because their classes can still be reopened in the caller).
- Disallowing cross-namespace constant access or method calls.

The last one would also fix that the design allows mixing objects from different versions of the same gem.

Implementation details aside, some conceptual solution has to be devised to make this sound.

BTW, my examples are synthetic. But for more practical minds, that Object could be an attribute three levels down the object tree from your namespace-level class instance.

#6 - 05/11/2025 01:24 PM - fxn (Xavier Noria)

The third option is more aligned with my intuitive idea of a new entity with new rules. Instead of a container of methods and constants that :: can work with, you may need a new kind of language-level entity an "isolate", a "cell", a "shield", something.

#7 - 05/11/2025 03:44 PM - fxn (Xavier Noria)

We have followed the discussion in https://bugs.ruby-lang.org/issues/21311.

I may contribute a doc patch to explain the scope to which root object references behave differently in different namespaces.

A consequence of this design is that you will need to mark classes as "cross-namespace-safe".

If your class depends on a core extension, instances of this class cannot be passed safely across namespaces.

That dependency may not be obvious, could come from a transitive dependency deep in your object tree you are not even aware of.

#8 - 05/13/2025 10:20 PM - fxn (Xavier Noria)

This ticket was kind of oriented to docs.

I believe the emphasis has to be different and have created https://bugs.ruby-lang.org/issues/21334.

This one can be close if you will.

#9 - 06/03/2025 05:47 AM - hsbt (Hiroshi SHIBATA)

- Tags set to namespace