Ruby - Bug #21334

Namespaces and object reference sharing

05/13/2025 10:19 PM - fxn (Xavier Noria)

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

Description

Implications related to builtin classes and modules

As we know, when a namespace is created, builtin classes in the namespace are as in the root one. This applies to all references to them, explicit or implicit.

This is at the core of the current implementation and semantics of namespaces.

However, the feature also allow object references to cross namespaces. For example, via method calls:

```
ns::Foo.m(ref)
```

And this has implications that do not seem good to me. Examples:

1. If ref has an HTTP client in an attribute, that depends on a parser gem, which depends on a string utils gem, that adds blank? to String, that call is broken because the transitive code at some point may perhaps hit a blank? call, and the user has no way to tell.

2. If ref is a reference created by a gem managed by Zeitwerk (there are hundreds), the call is broken too. When m uses ref, it might need to lazy load something, and that won't work because the Kernel#require and Module#const_added decorations are gone within the namespace.

Please, note whether a gem loads with Zeitwerk or not is transparent to the user of the gem, by design. How do gems load their code is a private matter that is not, and should not be, documented. So the user does not know.

Implications related to different gem versions

In the example above, ref could be an instance of an object that corresponds to v2.0 of a gem, and perhaps the code under the namespace has a transitive dependency on v1.0. Mixing such references seems like a receipt for a good amount of Paracetamol.

Proposal

This is all so brittle, that I would like to ask for a fundamental reconsideration of cross-namespace object reference passing.

The rules are consistent within a namespace, problem is opening that up to cross-namespace communication.

Maybe redefine communication across namespaces in a very restrictive way that make these issues by design. Maybe remove cross-namespace communication in this manner altogether. Or some other option people can think of.

History

#1 - 05/13/2025 10:25 PM - fxn (Xavier Noria)

> Maybe redefine communication across namespaces in a very restrictive way that make these issues by design

Maybe redefine communication across namespaces in a very restrictive way that **prevents** these issues by design

#2 - 05/13/2025 11:01 PM - mame (Yusuke Endoh)

- Status changed from Open to Rejected

I've lost count of how many times I've said this, but here it is again:

You must provide reproducible code, your expectation, and actual results.

The current implementation of namespaces is incomplete, and code using real-world gems might not work. But that does not mean you can't construct a minimal example to demonstrate the issue.

Your abstract explanations without concrete code are no longer acceptable.

I'm rejecting this ticket. If you want to pursue the issue, create a new ticket with a minimal, self-contained demonstration code.

> If ref has an HTTP client in an attribute, that depends on a parser gem, which depends on a string utils gem, that adds blank? to String, that call is broken...

Since there's no code, I can't say for sure, but based on the description, I guess you may have misunderstood how namespaces work.

If you call something like ref.http_client.fetch, the method fetch is executed in the context of the namespace where http_client lives. That's the namespace where blank? is defined. So in that case, the call should work.

Just to add a personal view: passing complex, non-trivial objects between namespaces is generally not recommended. However, mechanically forbidding it would be un-Ruby-like.

Ruby has always prioritized developer freedom. Monkey patching, for instance, is full of dangerous implications, but we allow it. We let users shoot themselves in the foot if they so choose. I think the same principle should apply here.

#3 - 05/13/2025 11:09 PM - fxn (Xavier Noria)

> If you call something like ref.http_client.fetch, the method fetch is executed in the context of the namespace where http_client lives. That's the namespace where blank? is defined. So in that case, the call should work.

blank? was defined in the main namespace, m is running in the "foo" namespace, therefore *within* that namespace String#blank? does not exist.

I do not provide code because these are thought experiments. I would guess people working on namespaces are going to read, nod, and think about it.

Tickets for namespaces like this one are not normal tickets due to the special and preliminary nature of the feature.

If there is a time to reconsider, it is now that has been exposed to the public.

I am not reporting a regular bug in regular code.

#4 - 05/13/2025 11:41 PM - fxn (Xavier Noria)

I have spent countless hours of my free time on namespaces in the last days. I experimented with the feature again tonight, and wrote this ticket with care, looking for edge cases, and presenting them, past midnight.

While rejecting on that basis is your call, I am truly disappointed by this reaction.

The thought experiments are clear, please consider them at your own discretion, I won't open any other ticket.

#5 - 05/14/2025 12:03 AM - mame (Yusuke Endoh)

This is not about whether the ticket is "normal" or not. The fact is, your explanation is vague and incomplete. You may feel it’s obvious, but it isn’t.

Do not assume others are smart enough to understand your explanation based on natural language alone without code. They are not, and they shouldn't have to be.

If you're just doing a thought experiment, please write it somewhere else. If you want us to understand you, you need to write reproducible code, expected behavior, actual behavior. They are not optional. No exceptions.

Here's an example of the code I was hoping you'd write. Just FYI.

```
# main.rb
class String
```

```
  def blank? = :test
end

class HTTPClient
  def do_something
    p "string".blank? # This call works
    "fetched_data"
  end
end

Test = Struct.new(:http_client)

ref = Test.new(HTTPClient.new)

ns = Namespace.new
ns.require "./sub"
ns::Foo.m(ref)

# sub.rb
class Foo
  def self.m(ref)
    res = ref.http_client.do_something # This call works

    res.blank? # As I was writing this code I was thinking,
               # are you saying the problem is that this call doesn't work?
  end
end

$ RUBY_NAMESPACE=1 ./miniruby main.rb
./miniruby: warning: Namespace is experimental, and the behavior may change in the future!
See doc/namespace.md for know issues, etc.
:test
/home/mame/work/ruby/sub.rb:4:in 'm': undefined method 'blank?' for an instance of String (NoMethodError)
        from main.rb:18:in '<main>'
```

#6 - 05/14/2025 12:17 AM - fxn (Xavier Noria)

I would have appreciated something like, the two last examples are clear, could you provide code for the first one?

By rejecting this way, you have lost me.