Ruby - Bug #21316

Namespaces leak with permanent names

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

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

Description

Namespaces are not transparent for this program

```
C = Class.new
C.name == 'C'
```

because under a non-main user namespace, the name of C has the namespace as a prefix.

Related issues:

Related to Ruby - Feature #21335: Namespaces should be present in the backtrace	**Open**

History

#1 - 05/10/2025 06:22 AM - mame (Yusuke Endoh)

This is definitely not ideal. Class#name could end up referring to a different constant.

```
# main.rb
NS = Namespace.new
NS.require "./sub"

# sub.rb
class C; end
p C #=> expected: C
    #=> actual: NS::C ← This could refer to a different constant, which is problematic
```

@tagomoris (Satoshi Tagomori) suggested that Class#name should behave as follows:

- If the current namespace is at the front, omit it and just return "C"
- Otherwise, return something like "#<Namespace: ...>::C"

```
# main.rb
NS = Namespace.new
NS.require "./sub"
p NS::C #=> #<Namespace: ...>::C

# sub.rb
class C; end
p C #=> C
```

#2 - 05/10/2025 08:18 AM - fxn (Xavier Noria)

Main problem here is that there are many programs that depend on that name.

They may store it somewhere for later const_get. For example, polymorphic Active Record associations store names in the database. Users may also set class and module names in configuration, to be later lazily loaded. Associations do this to specify the name of the target model. Active Job queue adapters are configured this way too, etc.

Also, the other way around. You may pass the name to a callback, and when the program detects a class or module object with that name is created (by looking at Module#name), the callback is invoked. This is the case in on_load callbacks in Zeitwerk, for instance.

That the name matches the constant path of the class or module being defined is a strong property of the language people rely on.

#3 - 05/10/2025 12:13 PM - Eregon (Benoit Daloze)

Right I think in the namespace defining the class/module the Module#name needs to not have a prefix, or it will break many gems.
OTOH when used outside, it could be very confusing without a prefix.

suggested that Class#name should behave as follows:

Yeah, that's probably the best trade-off, although of course it means the Module#name for a given Module changes based on which Namespace is the current one.

BTW, I think the main namespace constants should also be prefixed when seen in another namespace, currently it's not the case:

```
$ RUBY_NAMESPACE=1 ruby -ve 'main = Namespace.current; ns = Namespace.new; class C; end; ns::MAIN_C = C; File.
write "ns.rb", "p MAIN_C; p eval(MAIN_C.name)"; ns.require "./ns"'
ruby 3.5.0dev (2025-05-10T07:50:29Z namespace-on-read-.. bd4f57f96b) +PRISM [x86_64-linux]
ruby: warning: Namespace is experimental, and the behavior may change in the future!
See doc/namespace.md for know issues, etc.
C
(eval at /home/eregon/ns.rb:1):1:in '<top (required)>': uninitialized constant #<Namespace:24,user,optional>::
C (NameError)
 from /home/eregon/ns.rb:1:in 'Kernel#eval'
 from /home/eregon/ns.rb:1:in '<top (required)>'
 from -e:1:in 'Namespace#require'
 from -e:1:in '<main>'
```

#4 - 05/13/2025 11:20 AM - make_now_just (Hiroya Fujinami)

I found an issue on Marshal and Namespace maybe related to this ticket.

When dumping an object defined in a namespace using Marshal, the result will vary depending on whether the namespace is held as a variable or constant, and whether Marshal.dump is performed inside or outside the namespace.

In other words, we have the following files:

ns.rb:

```
class Foo
  def dump_in_ns = Marshal.dump(self)
end
```

main.rb:

```
ns = Namespace.new
ns.require("./ns.rb")

NS = Namespace.new
NS.require("./ns.rb")

puts "var / in ns"
begin
  ns::Foo.new.dump_in_ns
  p :ok
rescue => ex
  p :error
  p ex
end
puts

puts "const / in ns"
begin
  NS::Foo.new.dump_in_ns
  p :ok
rescue => ex
  p :error
  p ex
end
puts

puts "var / out ns"
begin
  Marshal.dump(ns::Foo.new)
  p :ok
rescue => ex
  p :error
  p ex
end
puts
```

```
puts "const / out ns"
begin
  Marshal.dump(NS::Foo.new)
  p :ok
rescue => ex
  p :error
  p ex
end
```

Then, the result is as follows:

```
$ RUBY_NAMESPACE=1 ruby main.rb
var / in ns
:error
#<TypeError: can't dump anonymous class #<Namespace:0x000000011bf3ed98>::Foo>

const / in ns
:error
#<ArgumentError: undefined class/module NS::>

var / out ns
:error
#<TypeError: can't dump anonymous class #<Namespace:0x000000011bf3ed98>::Foo>

const / out ns
:ok
```

I'm not sure this issue is completely related to this ticket, but Marshal.dump should work in any case.

#5 - 05/13/2025 11:49 AM - Eregon (Benoit Daloze)

@make_now_just (Hiroya Fujinami) In the first two cases, it should be .dump_in_ns not .dump

#6 - 05/13/2025 11:52 AM - make_now_just (Hiroya Fujinami)

@Eregon (Benoit Daloze) It has already been fixed. Thank you.

#7 - 05/14/2025 12:50 AM - mame (Yusuke Endoh)

- Related to Feature #21335: Namespaces should be present in the backtrace added

#8 - 05/23/2025 07:15 AM - ko1 (Koichi Sasada)

FYI: Java's case
"OBJ09-J. Compare classes and not class names"
https://wiki.sei.cmu.edu/confluence/display/java/OBJ09-J.+Compare+classes+and+not+class+names

#9 - 05/23/2025 07:31 AM - fxn (Xavier Noria)

@ko1 (Koichi Sasada) yeah, in Ruby you can have two classes with the same permanent name today. You know that, but let me show an example for the archives:

```
c = Class.new
C = c

Object.send(:remove_const, :C)

d = Class.new
C = d

p c.name == d.name # true
```

So, where possible, in arbitrary settings, you better work with class objects.

But there are common scenarios (configuration, etc., I described a few above) in which you do not have the object and the natural way to refer to the class is by its name.

Today, that is a common practice, and if you want to be able to load arbitrary code within a namespace, I think this has to be preserved. Otherwise, such code won't work correctly under a namespace.

#10 - 06/03/2025 05:48 AM - hsbt (Hiroshi SHIBATA)

- Tags set to namespace

#11 - 06/05/2025 04:32 AM - matz (Yukihiro Matsumoto)

As #21335, we should not provide information that depends on namespace, until we fix high level API.

Matz.