Ruby master - Bug #18292

3.1.0-dev `include` cause Module to be marked as initialized

11/08/2021 01:27 PM - byroot (Jean Boussier)

Status: Closed
Priority: Normal
Assignee: 
Target version:
ruby -v: 
ruby 3.1.0dev (2021-11-08T09:35:22Z master 7cc4e147fc)
Backport: 2.6: DONTNEED, 2.7: DONTNEED, 3.0: DONTNEED, 3.1: DONE

Description
Some subtle change I found while testing our app compatibility with Ruby head:

class Mod1 < Module
  def initialize(...)
    super
  end
end
p Mod1.new

class Mod2 < Module
  def initialize(...)
    include Enumerable
    super
  end
end
p Mod2.new

On 3.0:
#<Mod1:0x00007fbf9b825b38>
#<Mod2:0x00007fbf9b825818>

On ruby-head:

#<Mod1:0x00000001108cb1d8>
/tmp/module.rb:11:in `initialize': already initialized module (TypeError)
  from /tmp/module.rb:11:in `initialize'

I suspect this might be a result of https://bugs.ruby-lang.org/issues/17048, but I have to admit I'm not certain so I'd rather report it.

Associated revisions
Revision a79c5947 - 01/06/2022 04:03 PM - jeremyevans (Jeremy Evans)

Allow include before calling Module#initialize

This is to allow Module subclasses that include modules before calling super in the subclass's initialize.

Remove rb_module_check_initializable from Module#initialize.
Module#initialize only calls module_exec if a block is passed, it doesn't have other issues that would cause problems if called multiple times or with an already initialized module.

Move initialization of super to Module#allocate, though I'm not sure it is required there. However, it's needed to be removed from Module#initialize for this to work.

Fixes [Bug #18292]

Revision 28ec24db - 01/30/2022 10:08 AM - naruse (Yui NARUSE)

merge revision(s) a79c59472df38297c246b27713c277f2edaefa7a: [Backport #18292]
Allow include before calling Module#initialize

This is to allow Module subclasses that include modules before calling super in the subclass's initialize.

Remove rb_module_check_initializable from Module#initialize. Module#initialize only calls module_exec if a block is passed, it doesn't have other issues that would cause problems if called multiple times or with an already initialized module.

Move initialization of super to Module#allocate, though I'm not sure it is required there. However, it's needed to be removed from Module#initialize for this to work.

Fixes [Bug #18292]

---

class.c | 1 +
object.c | 1 -
test/ruby/test_module.rb | 10 ++++++++++
3 files changed, 11 insertions(+), 1 deletion(-)

History

#1 - 11/08/2021 01:28 PM - byroot (Jean Boussier)
- ruby -v set to ruby 3.1.0dev (2021-11-08T09:35:22Z master 7cc4e147fc)

#2 - 01/06/2022 12:59 AM - jeremyevans0 (Jeremy Evans)
A simple work around for this is include after calling super.
initialize checks whether the module is uninitialized, and fails otherwise. When calling include, the module is marked as initialized. If you call include before super, then the module is initialized before initialize is called, resulting in an error.

I'm not sure why this check in initialize is needed. initialize only does the equivalent of module_exec if a block is passed. I think it's possible to rearrange the code to fix this. I submitted a pull request for this change: https://github.com/ruby/ruby/pull/5398

#3 - 01/06/2022 04:03 PM - jeremyevans (Jeremy Evans)
- Status changed from Open to Closed

Applied in changeset git|a79c59472df38297c246b27713c277f2edaefa7a.

Allow include before calling Module#initialize

This is to allow Module subclasses that include modules before calling super in the subclass's initialize.

Remove rb_module_check_initializable from Module#initialize. Module#initialize only calls module_exec if a block is passed, it doesn't have other issues that would cause problems if called multiple times or with an already initialized module.

Move initialization of super to Module#allocate, though I'm not sure it is required there. However, it's needed to be removed from Module#initialize for this to work.

Fixes [Bug #18292]

#4 - 01/10/2022 10:25 AM - byroot (Jean Boussier)
@jeremyevans thanks for the fix. Do you think we should mark this as a 3.1 backport?

#5 - 01/10/2022 03:12 PM - jeremyevans0 (Jeremy Evans)
- Backport changed from 2.6: UNKNOWN, 2.7: UNKNOWN, 3.0: UNKNOWN to 2.6: DONTNEED, 2.7: DONTNEED, 3.0: DONTNEED, 3.1: REQUIRED

byroot (Jean Boussier) wrote in #note-4:

@jeremyevans thanks for the fix. Do you think we should mark this as a 3.1 backport?

Sure, that makes sense to me.
Backport changed from 2.6: DONTNEED, 2.7: DONTNEED, 3.0: DONTNEED, 3.1: REQUIRED to 2.6: DONTNEED, 2.7: DONTNEED, 3.0: DONTNEED, 3.1: DONE

ruby_3_1 28ec24db4a8201fa624692cdc1cbe9b00489412 merged revision(s) a79c59472df38297c246b27713c277f2edaefa7a.