Unhelpful error message when naming a module with the same name as an existing class

The error message when naming a module with the same name as an existing class causes more trouble than it helps.

Steps to Reproduce

class X;end
module X;end
=> TypeError: X is not a module

Expected Results

Ruby has this very helpful and explicit error when reassigning a value to an existing constant.

A = 1
A = 2
warning: already initialized constant A
warning: previous definition of A was here

which makes me expect Ruby to keep track of names of global constants and prevent me from colliding names with helpful warnings. This warning however raises a "Type error" which is not very intuitive. When users think of type errors they think of things like "foo" * "foo" => TypeError: no implicit conversion of String into Integer.

Actual Results

Confusion about what the actual error is, especially for people learning Ruby.

The reason for the difference in behavior here is due to Ruby’s open classes. When using class X or module X, if the constant X is already defined, Ruby will reopen the class or module definition, so you can operate in the context of the class or module. However, attempting to reopen a class when the object is not actually a class is an error, whereas reassigning a constant is just a warning.

The warning message you get during constant reassignment is used for classes and modules if you assign them:

X = Class.new
X = Module.new
# warning: already initialized constant X
# warning: previous definition of X was here
Likewise, the error raised when attempting to reopening an existing constant is pretty much the same for all objects that don't match the expected type:

```ruby
X = 1
class X; end
# TypeError (X is not a class)
module X; end
# TypeError (X is not a module)
```

So I don't think there is a bug here. It is possible to change the format the error message, though. Attached is a patch that changes the error message to include the current class of the object:

```ruby
class X; end
module X; end
# TypeError (X is not a module, it is a Class)
module Y; end
# TypeError (Y is not a class, it is a Module)
```

I would appreciate feedback on whether this more detailed error message is an improvement.

---

#2 - 07/04/2019 01:15 AM - duerst (Martin Dürst)
jeremyevans0 (Jeremy Evans) wrote:

So I don't think there is a bug here. It is possible to change the format the error message, though. Attached is a patch that changes the error message to include the current class of the object:

```ruby
class X; end
module X; end
# TypeError (X is not a module, it is a Class)
module Y; end
# TypeError (Y is not a class, it is a Module)
```

I would appreciate feedback on whether this more detailed error message is an improvement.

I think it is.

#3 - 07/04/2019 07:16 AM - nobu (Nobuyoshi Nakada)

Is "a Integer" OK? :)

#4 - 07/04/2019 07:35 AM - sawa (Tsuyoshi Sawada)
nobu (Nobuyoshi Nakada) wrote:

Is "a Integer" OK? :)

"a(n) Integer"

#5 - 07/04/2019 08:22 AM - nobu (Nobuyoshi Nakada)

- File 0001-Show-the-previous-definition-location.patch added
- Tracker changed from Bug to Feature
- ruby -v deleted (ruby 2.2.0p0)
- Backport deleted (2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN)

#6 - 07/04/2019 01:06 PM - shevegen (Robert A. Heiler)

I would appreciate feedback on whether this more detailed error message is an improvement.

I think this depends on what the ruby user at hand is doing; and how much expertise that ruby user may have. Some behaviour in ruby can be confusing to newcomers, such as when variables may be assigned to nil, without explicit "x = nil".

I personally do not really need a change either way in the context of this suggestion (I am fine with the current behaviour), but I think it is not completely unreasonable to assume that there could be use cases for letting ruby be more verbose in this (or
similar) situation, if it leads to an improvement such as less confusion for newcomers.

In the long run it may be good to sort of have ruby in different "modes": a bit like the did-you-mean gem and the -w flag for running ruby code, but with more control over the messages (as long as they are just warnings, but perhaps also some that relate to errors, in particular when ruby interacts with the user). But as written before, I am really mostly neutral either way - not really decidedly adopting any pro or con opinion, just commenting to the question by jeremy. :-)

#7 - 08/29/2019 04:24 AM - matz (Yukihiro Matsumoto)
Accepted.
Matz.

#8 - 08/29/2019 04:59 AM - nobu (Nobuyoshi Nakada)
- Status changed from Feedback to Closed

Applied in changeset git|761346a9604ca2c797777d1d671b5d5fc3c3f69.

Show the previous definition location,
when reopened class/module redefinition mismatched the previous definition. [Feature #11460]

Files
mod-reopen-error-message.patch 2.82 KB 07/03/2019 jeremyevans0 (Jeremy Evans)
0001-Show-the-previous-definition-location.patch 4.89 KB 07/04/2019 nobu (Nobuyoshi Nakada)