return from a proc in a module/class body returns out of script. Should be LJE.

09/25/2019 06:05 PM - enebo (Thomas Enebo)

Status: Closed
Priority: Normal
Assignee:
Target version:
ruby -v:
Backport: 2.5: UNKNOWN, 2.6: UNKNOWN

Description
return is not allowed from class/module body. But if we insert a return into a block then we can invoke the block then it returns all the way out of the script. This has to be accidental behavior doesn't it? I believe the case below should end up as a LocalJumpError:

```ruby
class Foo
  proc { return }.call
end
puts "NEVER SEEN"
```

This behavior started some time in 2.5 (it used to be an LJE).

History

#1 - 09/27/2019 11:00 PM - jeremyevans0 (Jeremy Evans)
It looked into this behavior, which started in Ruby 2.4 (when top level return started being allowed).

```ruby
class Foo
  alias proc lambda
  proc { return }.call
end
puts "NEVER SEEN"
```

Similarly, though this operates as a top level return, the warning for top level return with argument does not take effect, because the compiler doesn't know it will operate as a top level return:

```ruby
class Foo
  proc { return 1 }.call # no top level return with argument warning
end
puts "NEVER SEEN"
```

I don't think the compiler has enough information to handle this correctly, because it doesn't know at compilation time whether the block will be executed as a proc or as a lambda.

I do agree that a LocalJumpError is the most appropriate way to handle this.

#2 - 09/30/2019 06:21 PM - enebo (Thomas Enebo)
I agree this is impossible to be done by parser or at iseq generation time. It must be done when return is invoked.

JRuby will return LocalJumpError for this for 9.2.9.0. It also had from 9.2.6.0 and earlier.

As a runtime LJE we know lexically it is defined within a module/class and we can also look up stack to see if it has migrated or not (I assume MRI has even more flexibility in this than JRuby does). Similarly we know it is a lambda or not at that point so that is not really a problem at runtime.

Jeremy, since you are only person who has looked would you say semantically this should be some error vs silently only executing part of a file?

#3 - 09/30/2019 06:32 PM - jeremyevans0 (Jeremy Evans)
enebo (Thomas Enebo) wrote:

Jeremy, since you are only person who has looked would you say semantically this should be some error vs silently only executing part of a file?

I believe if return directly inside class/module is an error, return inside proc inside class/module should also be an error. Since it cannot be a compile time error, it should be a runtime error (LocalJumpError).
Alternatively, we start to allow return inside class/module, and then the current behavior for return inside proc inside class/module makes sense.

#4 - 09/30/2019 08:11 PM - enebo (Thomas Enebo)
Ok cool. Let's hope we get some more consensus on this. I don't like deviating from MRI in behavior so I hope to see more come to the same conclusion.

#5 - 09/30/2019 11:13 PM - jeremyevans0 (Jeremy Evans)
I've added a pull request that makes return in proc in class/module a LocalJumpError: https://github.com/ruby/ruby/pull/2511

#6 - 10/01/2019 03:03 PM - enebo (Thomas Enebo)
I added a comment about removing the pre 2.7 spec as it is unintended behavior.

#7 - 10/02/2019 03:14 PM - jeremyevans0 (Jeremy Evans)
- Status changed from Open to Closed

Fixed by ef697389bacedf36966a2edcdcf88baca342b9e2.