

Ruby - Feature #19755

Module#class_eval and Binding#eval use caller location by default

07/03/2023 11:19 AM - byroot (Jean Boussier)

Status:Closed Priority:Normal Assignee: Target version:	
Description Background In Ruby we're very reliant on Method#source_location as well as caller_locations to locate source code. However, code generated via Binding#eval, Module#class_eval etc defeat this if called without a location: <pre>Foo.class_eval <<~RUBY def bar end RUBY</pre> <pre>p Foo.instance_method(:bar).source_location # => ["(eval)", 1]</pre> The overwhelming majority of open source code properly pass a filename and lineno, however a small minority doesn't and make locating the code much harder than it needs to be. Here's some example of anonymous eval uses I fixed in the past: https://github.com/ruby/mutex_m/pull/11 https://github.com/rails/execjs/pull/120 https://github.com/jnunemaker/httparty/pull/776 https://github.com/SiftScience/sift-ruby/pull/76 https://github.com/activemerchant/active_merchant/pull/4675 https://github.com/rails/thor/pull/807 https://github.com/dry-rb/dry-initializer/pull/104 https://github.com/rmosolgo/graphql-ruby/pull/4288 Proposal I suggest that instead of defaulting to "(eval)", the optional filename argument of the eval family of methods should instead default to: "(eval in #{caller_locations(1, 1).first.path})" and lineno to caller_locations(1, 1).first.lineno. Which can pretty much be monkey patched as this: <pre>module ModuleEval def class_eval(code, location = "(eval in #{caller_locations(1, 1).first.path})", lineno = caller_locations(1, 1).first.lineno) super end end Module.prepend(ModuleEval)</pre> <pre>module Foo class_eval <<~RUBY def foo end RUBY end</pre> <pre>p Foo.instance_method(:foo)</pre> before:	

```
#<UnboundMethod: Foo#foo() (eval):1>
```

after:

```
#<UnboundMethod: Foo#foo() (eval in /tmp/foo.rb):10>
```

Of course the lineno part is likely to not be fully correct, but the reasoning is that it's better than defaulting to 0 or 1.

Another possibility would be to include the caller lineno inside the filename part, and leave the actual lineno default to 1:

```
#<UnboundMethod: Foo#foo() (eval at /tmp/foo.rb:10):1>
```

Associated revisions

Revision 43a5c191358699fe8b19314763998cb8ca77ed90 - 07/24/2023 12:51 PM - byroot (Jean Boussier)

Use the caller location as default filename for eval family of methods

[Feature #19755]

Before (in /tmp/test.rb):

```
Object.class_eval("p __FILE__") # => "(eval) "
```

After:

```
Object.class_eval("p __FILE__") # => "(eval at /tmp/test.rb:1) "
```

This makes it much easier to track down generated code in case the author forgot to provide a filename argument.

Revision 43a5c191358699fe8b19314763998cb8ca77ed90 - 07/24/2023 12:51 PM - byroot (Jean Boussier)

Use the caller location as default filename for eval family of methods

[Feature #19755]

Before (in /tmp/test.rb):

```
Object.class_eval("p __FILE__") # => "(eval) "
```

After:

```
Object.class_eval("p __FILE__") # => "(eval at /tmp/test.rb:1) "
```

This makes it much easier to track down generated code in case the author forgot to provide a filename argument.

Revision 43a5c191 - 07/24/2023 12:51 PM - byroot (Jean Boussier)

Use the caller location as default filename for eval family of methods

[Feature #19755]

Before (in /tmp/test.rb):

```
Object.class_eval("p __FILE__") # => "(eval) "
```

After:

```
Object.class_eval("p __FILE__") # => "(eval at /tmp/test.rb:1) "
```

This makes it much easier to track down generated code in case the author forgot to provide a filename argument.

History

#1 - 07/03/2023 12:01 PM - Eregon (Benoit Daloze)

#<UnboundMethod: Foo#foo() (eval in /tmp/foo.rb):10> sounds great to me, +1.

The (eval makes it clear it's an eval in that file and so the line of an exception inside might not be exact.

#2 - 07/03/2023 01:15 PM - Dan0042 (Daniel DeLorme)

+1

Just be careful about the implementation, because that monkey patch doesn't work if another module is in the call chain

```
module DebugEval
  def class_eval(code, ...)
    p debug: code
    super # <= source_location will report the eval comes from here
  end
  prepend_features Module
end
```

Actually this is a problem I tend to have whenever I want to use caller_locations. Thread.each_caller_location has made it easier but it would be nice to have something like #first_caller_location_which_is_not_self_or_super

#3 - 07/03/2023 01:43 PM - byroot (Jean Boussier)

doesn't work if another module is in the call chain

I'm not sure we can / should handle this. Decorating class_eval / eval should be quite rare anyways.

#4 - 07/03/2023 03:59 PM - Eregon (Benoit Daloze)

byroot (Jean Boussier) wrote in [#note-3](#):

Decorating class_eval / eval should be quite rare anyways.

Indeed, and class_eval/eval is broken if decorated e.g. for a = 3; class_eval "a".
The direct caller of Module#class_eval should always be the code around it.
Full example:

```
# works as-is, breaks with DebugEval
module M
  a = 3
  class_eval "p a"
end
```

#5 - 07/03/2023 04:30 PM - Dan0042 (Daniel DeLorme)

Indeed, and class_eval/eval is broken if decorated e.g. for a = 3; class_eval "a".

Oh yeah, good point, that one slipped my mind.

#6 - 07/11/2023 12:30 PM - nobu (Nobuyoshi Nakada)

```
Foo.class_eval <<~RUBY
  def bar
  end
RUBY
```

This code equals Foo.class_eval "def bar\n\"\"end\n\"\"".
Which do you expect 1 or 2 as __LINE__ at the def line?

#7 - 07/11/2023 12:33 PM - byroot (Jean Boussier)

Which do you expect 1 or 2 as __LINE__ at the def line?

I don't feel strongly about either, as it is assumed that the line number can't always be correct anyway.

I think 1 seem more logical to me, as it would be weird to assume a heredoc was used.

#8 - 07/11/2023 01:19 PM - Dan0042 (Daniel DeLorme)

I think 1 seem more logical to me, as it would be weird to assume a heredoc was used.

Ah, but with [#19458](#) it would be *possible* to know that a heredoc was used, and use 2 for the def line. :-D

#9 - 07/13/2023 08:44 AM - matz (Yukihiro Matsumoto)

Accepted. I prefer the last format `#<UnboundMethod: Foo#foo() (eval at /tmp/foo.rb:10):1>`.

Matz.

#10 - 07/13/2023 01:20 PM - nobu (Nobuyoshi Nakada)

matz (Yukihiro Matsumoto) wrote in [#note-9](#):

Accepted. I prefer the last format `#<UnboundMethod: Foo#foo() (eval at /tmp/foo.rb:10):1>`.

In that case, what should `__FILE__` and `__dir__` be?

`"(eval at /tmp/foo.rb:10)"` as `__FILE__` may be ok, but `"(eval at /tmp" as __dir__?`

#11 - 07/13/2023 01:43 PM - byroot (Jean Boussier)

Oh, `__dir__` is a good point, I haven't thought of it.

Currently, it's nil:

```
>> eval("__dir__")
=> nil
```

I suppose we should just keep that?

#12 - 07/13/2023 01:47 PM - byroot (Jean Boussier)

I just finished a PR for it, but I agree we need to handle `__dir__`: <https://github.com/ruby/ruby/pull/8070>

#13 - 07/18/2023 10:23 AM - byroot (Jean Boussier)

we need to handle `__dir__`

So the way it was handled until now was simply:

```
if (path == eval_default_path) {
  return Qnil;
}
```

So how I'm handling it right now is basically `path.start_with?("eval at ") && path.end_with?("(")`. It doesn't feel great, but I can't think of another solution, and I think it's good enough.

The last blocker is that this change breaks `syntax_suggest` test suite, so I need https://github.com/ruby/syntax_suggest/pull/200 merged first. Other than that I think the PR is good to go.

#14 - 07/20/2023 06:01 PM - schneems (Richard Schneeman)

I've merged the PR. Thanks for your work here!

#15 - 07/24/2023 12:51 PM - byroot (Jean Boussier)

- Status changed from Open to Closed

Applied in changeset [git|43a5c191358699fe8b19314763998cb8ca77ed90](https://github.com/ruby/ruby/commit/43a5c191358699fe8b19314763998cb8ca77ed90).

Use the caller location as default filename for eval family of methods

[Feature [#19755](#)]

Before (in `/tmp/test.rb`):

```
Object.class_eval("p __FILE__") # => "(eval)"
```

After:

```
Object.class_eval("p __FILE__") # => "(eval at /tmp/test.rb:1)"
```

This makes it much easier to track down generated code in case the author forgot to provide a filename argument.