Ruby - Feature #6210

load should provide a way to specify the top-level module

03/28/2012 02:57 AM - now (Nikolai Weibull)

Status:	Closed	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:		

Description

load currently takes an optional second argument that allows you to load into an anonymous and new top-level Module. It would be nice if the second argument could also be a Module that would then be used as the top-level Module. That way one could provide a set of methods that should be available to the content being loaded without having to put them in Kernel.

Associated revisions

Revision b35b7a1ef25347735a6bb7c28ab7e77afea1d856 - 11/18/2021 06:43 AM - jeremyevans (Jeremy Evans)

Allow Kernel#load to load code into a specified module

Instead of always using a new anonymous module for Kernel#load if
the wrap argument is not false/nil, use the given module if a module
is provided.

Implements [Feature #6210]

Revision b35b7a1ef25347735a6bb7c28ab7e77afea1d856 - 11/18/2021 06:43 AM - jeremyevans (Jeremy Evans)

Allow Kernel#load to load code into a specified module

Instead of always using a new anonymous module for Kernel#load if
the wrap argument is not false/nil, use the given module if a module
is provided.

Implements [Feature #6210]

Revision b35b7a1e - 11/18/2021 06:43 AM - jeremyevans (Jeremy Evans)

Allow Kernel#load to load code into a specified module

Instead of always using a new anonymous module for Kernel#load if
the wrap argument is not false/nil, use the given module if a module
is provided.

Implements [Feature #6210]

History

#1 - 03/28/2012 03:36 AM - luislavena (Luis Lavena)

- Assignee set to matz (Yukihiro Matsumoto)

- Target version set to 2.0.0

Sounds partially like #5643?

#2 - 03/31/2012 11:10 AM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

#3 - 10/27/2012 06:50 AM - ko1 (Koichi Sasada)

- Target version changed from 2.0.0 to 2.6

I changed target to next minor because no discussion on it.

#4 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)

#5 - 10/18/2021 06:32 PM - jeremyevans0 (Jeremy Evans)

I think this is a useful feature, and it can be implemented without any API changes/backwards compatibility issues. I submitted a pull request for it: https://github.com/ruby/ruby/pull/4986

#6 - 11/10/2021 01:00 PM - byroot (Jean Boussier)

Agreed, it could be used to experiment with new ways to load code in a non global manner.

That being said it would entirely sidestep iseq caching. Not that it's a deal breaker, but if this were to be used for loading lots of code, we might also need RubyVM::InstructionSequence#eval(wrapping_module)

#7 - 11/18/2021 06:23 AM - matz (Yukihiro Matsumoto)

OK, I (finally) accepted the proposal. Go ahead.

Matz.

#8 - 11/18/2021 08:41 AM - jeremyevans (Jeremy Evans)

- Status changed from Assigned to Closed

Applied in changeset git|b35b7a1ef25347735a6bb7c28ab7e77afea1d856.

Allow Kernel#load to load code into a specified module

Instead of always using a new anonymous module for Kernel#load if
the wrap argument is not false/nil, use the given module if a module
is provided.

Implements [Feature #6210]

#9 - 11/18/2021 08:46 AM - byroot (Jean Boussier)

What about the RubyVM::InstructionSequence was there any discussion to allow caching Kernel.load(path, module_instance)?

#10 - 11/18/2021 05:59 PM - jeremyevans0 (Jeremy Evans)

byroot (Jean Boussier) wrote in #note-9:

> What about the RubyVM::InstructionSequence was there any discussion to allow caching Kernel.load(path, module_instance)?

The dev meeting log does not indicate this was discussed. If you think it's important, can you add it as a new feature request?