

Ruby master - Feature #15752

A dedicated module for experimental features

04/07/2019 01:56 PM - Eregon (Benoit Daloze)

Status:	Closed	
Priority:	Normal	
Assignee:		
Target version:		
Description		
<p>I believe we should introduce a module for experimental features portable across Ruby implementations.</p> <p>An example of such a portable experimental feature is <code>RubyVM.resolve_feature_path</code>. This feature has nothing MRI specific in it, it is a part of basic require functionality. In the future, I would think more experimental features will be introduced, and I think <code>RubyVM</code> is not a good place for it.</p> <p>Currently, <code>RubyVM</code> is sometimes used for experimental features, but I believe <code>RubyVM</code> should be defined only on MRI and contain only MRI-specific features. This means it is <i>impossible</i> for other implementations such as <code>JRuby</code> and <code>TruffleRuby</code> to define <code>resolve_feature_path</code> (even though it's trivial and might be useful for some users), and keeping <code>RubyVM</code> not defined for clearly marking MRI specific features are not available.</p> <p>This is a problem that will only gets worse as portable experimental features are added to <code>RubyVM</code>. Here is one example of adding an experimental feature but unfortunately there is no common place between Ruby implementations to add it: https://github.com/jruby/jruby/issues/5206</p> <p>If other implementations defined <code>RubyVM</code>, then only parts of it would be portable and other parts would be MRI specific, which would be very confusing to both users and Ruby implementers.</p> <p>Also, <code>RubyVM</code> doesn't really indicate by its name that it contains experimental features.</p> <p>So I propose the obvious name <code>ExperimentalFeatures</code>.</p> <p>I think such a long name is unlikely to clash with existing Object constants, is very clear, and marks that any usage of it is by definition using not stable APIs that might be removed or changed.</p> <p>In combination with #15743, this would mean we can very clearly see what kind of feature it is due to explicit naming:</p> <ul style="list-style-type: none">• <code>ExperimentalFeatures.resolve_feature_path</code> is a portable experimental feature, which can be supported on other Ruby implementations too.• <code>CRuby::InstructionSequence</code> is a <code>CRuby</code>/MRI-specific feature, which will only be supported on MRI. <p>OTOH, the <code>RubyVM</code> name doesn't indicate this important difference, and doesn't even indicate the features under it might be experimental or not portable.</p> <p>My main motivation here, is allowing other Ruby implementations to support some of these portable experimental features. There is no reason for only MRI to be able to support code using portable experimental features.</p> <p>cc name (Yusuke Endoh)headius (Charles Nutter)</p>		
Related issues:		
Related to Ruby master - Feature #15743: <code>RubyVM</code> should be renamed to <code>CRuby</code>		Closed
Related to Ruby master - Feature #15903: Move <code>RubyVM.resolve_feature_path</code> to ...		Closed
Related to Ruby master - Feature #15966: Introducing experimental features be...		Rejected
Related to Ruby master - Feature #14844: Future of <code>RubyVM::AST?</code>		Open

History

#1 - 04/07/2019 01:56 PM - Eregon (Benoit Daloze)

- Related to Feature #15743: `RubyVM` should be renamed to `CRuby` added

#2 - 04/07/2019 01:58 PM - Eregon (Benoit Daloze)

- Description updated

#3 - 04/08/2019 01:26 AM - ioquatix (Samuel Williams)

I think it's a good idea. Might I suggest two potential ideas?

Firstly, maybe have a shared Ruby module for common but interpreter specific functionality.

Then, interpreter specific modules e.g. CRuby, MRuby, JRuby, TruffleRuby for interpreter specific functionality.

For experimental stuff, you could choose either CRuby::Experimental or Ruby::Experimental. The path for loading such features would be require 'ruby/experimental/thing'.

Maybe also worthwhile considering how Python's `__future__` works, e.g.

<https://stackoverflow.com/questions/7075082/what-is-future-in-python-used-for-and-how-when-to-use-it-and-how-it-works> is a quick overview if anyone is unfamiliar.

+1

#4 - 04/08/2019 09:31 AM - Eregon (Benoit Daloze)

ioquatix (Samuel Williams) wrote:

I think it's a good idea. Might I suggest two potential ideas?

Go ahead :)

Firstly, maybe have a shared Ruby module for common but interpreter specific functionality.

This is equivalent to my proposition of ExperimentalFeatures with a different name, right?

I don't understand how it can be "common" and also "interpreter specific". They are opposite to me. Did you mean "but not"?

Then, interpreter specific modules e.g. CRuby, MRuby, JRuby, TruffleRuby for interpreter specific functionality.

For experimental stuff, you could choose either CRuby::Experimental or Ruby::Experimental.

I was thinking both the common namespace and the interpreter-specific namespaces are experimental. But indeed, maybe we need to be more fine-grained.

For instance, TruffleRuby defines a few methods and classes under TruffleRuby and those are fairly stable and documented.

<https://github.com/oracle/truffleruby/blob/master/doc/user/truffleruby-additions.md#truffleruby-methods-and-classes>

BTW, a few of those would probably make sense under ExperimentalFeatures (most of these are needed by ConcurrentRuby).

I think anything under ExperimentalFeatures is experimental.

Common stable (non-experimental) features should just be under another namespace (e.g., Kernel or whatever feels appropriate, including possibly a new class/module for the feature).

For interpreter-specific namespaces, I think it's OK to delegate to the documentation of the implementation to say what's experimental and what's stable,

although the distinction should be simple such as TruffleRuby would be stable and TruffleRuby::Experimental would be experimental.

I think CRuby::Experimental would be good (as a replacement for RubyVM), because it clearly marks such API are MRI specific and have not matured to a stable API yet.

Just as an example, RubyVM::AbstractSyntaxTree will probably break usages of it whenever a node field is added, removed or reordered, so being clearly marked as experimental in the usages seems good ([#14844](#) is an example that this is not clear at all for users currently with RubyVM).

The path for loading such features would be require 'ruby/experimental/thing'.

Typically the require is not needed, such functionality is just declared from startup.

That's the case for JRuby, TruffleRuby and Rubinius.

I'm not against it, but I don't see what it solves.

For feature checking, defined?(ExperimentalFeatures.foo) (or respond_to?) seems good enough.

Maybe also worthwhile considering how Python's `__future__` works, e.g.

<https://stackoverflow.com/questions/7075082/what-is-future-in-python-used-for-and-how-when-to-use-it-and-how-it-works> is a quick overview if anyone is unfamiliar.

How would that work in Ruby?

It seems more targeted at trying to make code more compatible with more recent versions, which I think we simply do by deprecation in Ruby and not breaking syntax.

#5 - 04/08/2019 09:54 AM - naruse (Yui NARUSE)

Web browsers showed us that it cannot achieve at once both experimental and portable.

#6 - 04/08/2019 11:41 AM - shevegen (Robert A. Heiler)

One worry that I have here is that this change may add bureaucratic overhead to MRI in particular. I have nothing against alternative ruby implementations at all, quite the opposite - the easier it is to implement ruby/rubies the better. The core team also tried to help here, e. g. ISO spec of ruby; and alternative implementations also helped likewise the other way around, such as rubinius + ruby spec early on (and still maintained, also by ruby contributors; I think Benoit extended the spec too, and this may be a partial reason for the suggestion perhaps). But having a situation where changes to MRI could possibly be delayed due to difficulties of alternative implementations would be a bad thing too, in my opinion. I think most people use MRI and any change to MRI in this regard should also be kept in mind.

#7 - 04/08/2019 11:49 AM - Eregon (Benoit Daloze)

naruse (Yui NARUSE) wrote:

Web browsers showed us that it cannot achieve at once both experimental and portable.

I will dare to challenge that.

Why would `ExperimentalFeatures.resolve_feature_path` not be portable?

Portable here just means it *can* be implemented by other Ruby implementations, and it is designed to not be specific to a given implementation (i.e., it can be implemented on other Ruby implementations).

Re browsers, I think the main problem is every browser used their own prefixes.

If we use a common namespace, and discuss all additions in this tracker, I don't think we'll have that problem.

#8 - 04/08/2019 11:52 AM - Eregon (Benoit Daloze)

shevegen (Robert A. Heiler) wrote:

But having a situation where changes to MRI could possibly be delayed due to difficulties of alternative implementations would be a bad thing too, in my opinion.

I am not proposing anything like that.

Practically, whenever MRI decides to add an experimental feature,

the only change is if it could potentially be implemented on other Ruby implementations ("portable" as I just defined above), add it under `ExperimentalFeatures` instead of under `RubyVM`. That's all.

#9 - 04/08/2019 06:02 PM - Eregon (Benoit Daloze)

- Subject changed from A dedicated module for portable experimental features to A dedicated module for experimental features

#10 - 04/17/2019 08:19 AM - knu (Akinori MUSHASHI)

Using a plural constant name sounds like a good idea because it wouldn't likely conflict with existing model names. ☐☐

#11 - 04/17/2019 08:27 AM - knu (Akinori MUSHASHI)

I think it's a good idea to reserve a namespace globally shared among Ruby implementations, even if it's up to each implementation whether to follow individual features proposed by other implementations.

#12 - 06/05/2019 08:14 PM - Eregon (Benoit Daloze)

- Related to Feature #15903: Move `RubyVM.resolve_feature_path` to `Kernel.resolve_feature_path` added

#13 - 06/30/2019 12:26 PM - Eregon (Benoit Daloze)

- Related to Feature #15966: Introducing experimental features behind a flag, disabled by default added

#14 - 12/14/2019 11:06 AM - Eregon (Benoit Daloze)

FWIW, Java has such a concept too that they call "incubator modules" and basically just have the new experimental API under `jdk.incubator` (<https://www.azul.com/openjdk-more-speed-less-haste/> for details).

I think we should introduce `ExperimentalFeatures` in Ruby, it's so much clearer than `RubyVM` (which is also problematic, see above).

Since we can't really agree on what RubyVM is (currently a mix of stable and less stable APIs), I think RubyVM should just becoming a supported module like the rest of the core API, and only ExperimentalFeatures should have this special status for experimental APIs. Thoughts?

#15 - 08/01/2020 01:01 PM - Eregon (Benoit Daloze)

- Related to Feature #14844: Future of RubyVM::AST? added

#16 - 08/01/2020 01:25 PM - Eregon (Benoit Daloze)

The existence of RubyVM prevents other Ruby implementations to be fully 100% compatible, because more and more gems start to rely on it. So either:

- We move most of RubyVM to ExperimentalFeatures so that other Ruby implementations can implement it for compatibility (e.g., AbstractSyntaxTree, InstructionSequence). That also gives a nice way to experiment with not-yet-stable features/APIs.
- We are fine that other Ruby implementations have a RubyVM constant too. That will make it even less clear that RubyVM is experimental.

In practice, if CRuby exposes any API, then some gems might rely on it and with enough gems it becomes hard/impossible to remove it. So if CRuby wants extra API for e.g. internal VM debugging, it should be added only with some special ./configure flag. That I think is the only way to prevent gems to (probably unknowingly) depend on CRuby-private APIs that are not meant to be used except for CRuby internal debugging.

#17 - 08/01/2020 01:29 PM - Eregon (Benoit Daloze)

To clarify, by "move" I mean moving under ExperimentalFeatures but we could still e.g. have a deprecated constant under RubyVM to help transition.

#18 - 08/03/2020 07:04 PM - Dan0042 (Daniel DeLorme)

Can I ask what would be so bad about having the RubyVM constant in other implementations? I mean, in itself the name is very general. Every implementation can be said to be a "Ruby VM".

If support for an experimental feature is indicated by RubyVM.has_feature_xyz?, an implementation can choose to return false for the moment. When the feature moves past the experimental stage the implementation can add it, and RubyVM.has_feature_xyz? can return true, and there's no need to move it out of the ExperimentalFeatures namespace.

If the intention is to rename RubyVM::AbstractSyntaxTree to AbstractSyntaxTree once it's no longer experimental, then I agree it would make more sense as Experimental::AbstractSyntaxTree. But if RubyVM::AbstractSyntaxTree is intended as the final name then temporarily stuffing it into an Experimental namespace would be way more trouble than it's worth.

Honestly asking: what is the benefit of messing around with what seems to have become a de-facto standard?

(apologies in advance if I missed the point of all this)

#19 - 08/03/2020 07:34 PM - Eregon (Benoit Daloze)

Dan0042 (Daniel DeLorme) wrote in [#note-18](#):

Can I ask what would be so bad about having the RubyVM constant in other implementations? I mean, in itself the name is very general. Every implementation can be said to be a "Ruby VM".

It's a good question. My understanding is that RubyVM was really meant as the JRuby or TruffleRuby module of JRuby/TruffleRuby, i.e., a module specifically for CRuby-specific things.

I tried to rename it to CRuby for clarity in <https://bugs.ruby-lang.org/issues/15743> but failed: <https://bugs.ruby-lang.org/issues/15743#note-21>

See also <https://bugs.ruby-lang.org/issues/15743#note-7> which notes that about everything under RubyVM was meant to be CRuby-specific. But since CRuby is the standard implementation, about any public API, including RubyVM will become used and depended on. Even more so if the name doesn't imply "experimental/unstable".

I would like experimental APIs to be clearly marked as such, and RubyVM doesn't achieve that well at all.

In fact I would bet many users of RubyVM don't even know it's experimental, or don't even know it's CRuby-specific (I tried to document that, but people don't read the documentation all the time).

So if RubyVM becomes shared it will become clear it's no longer CRuby-specific experiments. I'm fine with that, because RubyVM is already used in production by now (e.g., bootsnap).

I think as a result CRuby will no longer have a module for experimental features then.

Not necessarily a bad thing, as truly experimental APIs should probably be behind a ./configure flag if developers are serious about not having gems depend on it.

And I think it's good that new APIs, even if experimental, consider about portability since we have more than 1 implementation in the Ruby ecosystem.

Adding ExperimentalFeatures would let CRuby and other implementations experiment in a shared namespace, which seems clearer for everybody. It wouldn't prevent gems to depend on it, but at least the name clearly states the intention.

So, in summary I don't really mind either way.

I think if we make RubyVM a shared namespace then CRuby will need another module/way for experimental/CRuby-specific features.

#20 - 08/03/2020 07:40 PM - Eregon (Benoit Daloze)

I think an important lesson the Ruby implementors have learned over time is that there are no CRuby-specific APIs, it's a myth. Whatever was thought once as CRuby-specific will eventually be implemented on some other Ruby implementation, because they need it for compatibility.

There is definitely a need to introduce experimental not-yet-matured APIs. RubyVM is not a great place for that, as experimental APIs might be completely unrelated to the VM.

#21 - 08/04/2020 12:19 AM - mame (Yusuke Endoh)

FYI: [Type-profiler](#), which I'm developing for an experimental type inference tool for Ruby 3 types, heavily depends upon `RubyVM::InstructionSequence` because it performs static analysis on MRI byte code.

#22 - 08/15/2020 10:20 AM - Eregon (Benoit Daloze)

mame (Yusuke Endoh) wrote in [#note-21](#):

FYI: [Type-profiler](#), which I'm developing for an experimental type inference tool for Ruby 3 types, heavily depends upon `RubyVM::InstructionSequence` because it performs static analysis on MRI byte code.

Is there a reason to use bytecode instead of the AST (e.g., from parser or `RubyVM::AbstractSyntaxTree` which can be portable), which has more information?

It seems suboptimal to me to have any "Ruby 3 types project" depend on something like `RubyVM::InstructionSequence`, that will make it practically impossible to work on alternative Ruby implementations.

Also the bytecodes change regularly, so this will probably regularly break any project depending on it.

Note that it is orthogonal to this issue, any library using RubyVM could switch to `ExperimentalFeatures` easily (by `ExperimentalFeatures = RubyVM` unless defined?(`ExperimentalFeatures`)).

#23 - 09/01/2020 08:06 AM - matz (Yukihiro Matsumoto)

- *Status changed from Open to Closed*

As far as I understand, you want a place to separate features that do not only belong to CRuby, right?

But for example, we started `RubyVM::AbstractSyntaxTree` as a CRuby specific feature, it was natural for us to place it under RubyVM.

So it may be uncertain that the feature is implementation-specific or not from the start. So we have to discuss for each feature that can be shared among implementations (like `RubyVM::AbstractSyntaxTree`). We might give it a new name. In my opinion, we don't need a place like `ExperimentalFeature`, because it is not what we really need.

Matz.