Ruby trunk - Feature #7704

Add a list of enabled (experimental) language features.

01/16/2013 10:54 AM - mpapis (Michal Papis)

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

Description

With multiple Ruby implementations and "experimental" features like "refinements" it would be nice to have an array or hash including list of enabled language features so developers could check it instead of auto-discovering code with some hacks.

Additionally a new keyword like require_features :refinements, ... could be introduced to allow easy validation via either exception or return status.

History

#1 - 01/16/2013 12:51 PM - naruse (Yui NARUSE)

mpapis (Michal Papis) wrote:

> With multiple Ruby implementations and "experimental" features like "refinements" it would be nice to have an array or hash including list of enabled language features so developers could check it instead of auto-discovering code with some hacks.

Such feature list system is used on many languages/platforms like W3C DOM.
As far as my understand such experiment are failed because such feature list is too rough to use.
Features are not all or nothing in real world, and they often have bugs.
So auto discovery won't work.

> Additionally a new keyword like require_features :refinements, ... could be introduced to allow easy validation via either exception or return status.

Use defined?(define_method) or defined?(using).

#2 - 01/16/2013 01:25 PM - phluid61 (Matthew Kerwin)

naruse (Yui NARUSE) wrote:

> mpapis (Michal Papis) wrote:
>
> > With multiple Ruby implementations and "experimental" features like "refinements" it would be nice to have an array or hash including list of enabled language features so developers could check it instead of auto-discovering code with some hacks.
>
> Such feature list system is used on many languages/platforms like W3C DOM.
> As far as my understand such experiment are failed because such feature list is too rough to use.
> Features are not all or nothing in real world, and they often have bugs.
> So auto discovery won't work.

Not offering an opinion as such, just adding to the discussion.

I have in mind a related system: the OpenGL Extension Library. The registry of official language extensions is managed by an Architecture Review Board, and any OpenGL program can query the framework to detect the presence of an extension.

If Ruby were to support a similar system someone would be responsible for maintaining a registry of language features, including an explicit spec describing each feature (to which an implementation must fully adhere if it says it supports it). For example, the current "refinements" feature could be bundled under the name :REFINEMENTS_2_0 and match the current spec exactly; where a future version (e.g. :REFINEMENTS_2_1) would have a different spec.

That way a particular implementation may choose to implement some 2.0 features (e.g. kwargs) without others (e.g. refinements).

However, I imagine this would quickly become an unwieldy list, as per the OpenGL Extensions.

#3 - 01/26/2013 06:34 AM - drbrain (Eric Hodel)

- Target version set to 2.6

#4 - 02/22/2013 09:20 AM - ko1 (Koichi Sasada)

- Category set to core

- Assignee set to matz (Yukihiro Matsumoto)

#5 - 02/22/2013 10:59 AM - duerst (Martin Dürst)

I can only second Yui and Matthew. Directly checking is possible because Ruby is a dynamic language, it avoids management overhead, and tests on the actual feature, not some intention that may be out of sync with actual facts. There are many organizations that have tried to do this, and most if not all have failed; I don't see any reason why we can't learn from them and use our time for more productive work on other stuff.

#6 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)