Ruby trunk - Feature #13309

Locale paramter for Integer(), Float(), Rational()

03/13/2017 04:28 AM - sawa (Tsuyoshi Sawada)

Status:	Rejected	
Priority:	Normal	
Assignee:		
Target version:		

Description

Matz' comment to my proposal on Kernel#Boolean() (https://bugs.ruby-lang.org/issues/13260) led me to the idea of introducing an optional locale parameter (called under the name allow in #13260). And now, I think that that idea can be applied generally to the Kernel#Integer, Kernel#Float, and Kernel#Rational methods.

There seems to be constant demand to parse numbers written in different languages, and there is no official clean way to handle the problem. I thought we can let these methods take an optional locale parameter to allow inputs in different languages.

```
Float("4,294,967,295.0") # => 4294967295.0 English, by default
Float("4,294,967,295.0", locale: "en") # => 4294967295.0
Float("4 294 967.295,0", locale: "de") # => 4294967295.0
Float("4 294 967 295,0", locale: "fr") # => 4294967295.0
```

Cf. https://docs.oracle.com/cd/E19455-01/806-0169/overview-9/index.html

---

History

#1 - 03/13/2017 06:33 AM - duerst (Martin Dürst)

I agree that it would be great to have such a facility. I'm not sure we need locale:, we could just allow it as a second parameter (once we have made sure that there are no other suitable candidates for further parameters). I would personally start with 'output' functions (i.e. to_s), because they are way more deterministic.

Please note that currently, Float("4,294,967,295.0") throws an error, we may want (or not) to keep it that way, and only allow commas if a locale is explicitly set.

Also, we may want to create some kind of Locale object and allow this as a parameter. Also, we may want to allow more direct specification (e.g. decimal_separator: ',', thousands_separator: '.'; but note that this isn't complete because in other locales (e.g. India), the separator will not separate thousands).

#2 - 05/19/2017 09:00 AM - naruse (Yui NARUSE)

- Status changed from Open to Rejected

It should be done by gem.
This will depends Unicode CLDR, and it's beyond Ruby itself's scope.