Ruby - Bug #7827

Allow users to yield a block in string format

02/12/2013 02:47 AM - shevegen (Robert A. Heiler)

Status:	Rejected		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	1.9.3p385	**Backport:**	

Description

Hi.

Not long ago I realized that you can use "invalid" syntax inside a ruby block.

For instance:

def foo; end

foo() { ABC } # ABC is not known.

This works, until you yield that.

And if you yield it, that would fail, unless the ABC constant would be known.
(Or perhaps a global variable instead, so let's consider:

foo() { $abc }

So far so good.

Now I actually had the idea that a ruby user might want to
return the content of a block in string format, modify that
string, and then somehow eval it (or .call it)

Would that be useful? Or even possible?

It probably is not so useful, but it would be kind of
fun to be able to put whole classes inside a block,
which later could get known, or turned into an unbound
object again... Ok, I have no idea if this is useful
at all. :)

If it is just a crazy idea, just close it :D

So my suggestion would be:

def foo; end
result = foo() { $abc }
$abc = 5

And here, either yield externally on result,

or turn it into a proc somehow, and then

before evaluating it, perhaps add a new

method like:

result.proc.proc_string # This here would return '$abc'
which could then manually be eval()ed at a later

time.

I know, I have absolutely no use case for this, I just thought it would be cool to use invalid code inside {} and then perhaps lateron turn this into valid ruby code (or use it in some other way, such as **END** and DATA.read)

Thanks for reading at least!

History

#1 - 02/12/2013 09:50 AM - matz (Yukihiro Matsumoto)

- Status changed from Open to Rejected

Adding it as a standard feature requires Ruby to hold source information, that leads to memory consumption.
You might be interested in ruby2ruby for the purpose.

Matz.