Ruby trunk - Feature #729

curly brackets" and "begin end" blocks should behave syntactically and semantically exactly the same

11/09/2008 05:10 AM - tpo (Tomas Pospisek)

Status:	Rejected
Priority:	Normal
Assignee:	
Target version:	

Description

=begin
(transported over from rubyforge #16111, as I still think this would be worth while...)

Every now and then I run into a "principle of least surprise" violation wrt blocks in Ruby.

The most primitive problem i have is this: why does the following work:

```
# the code below is grouped together because it represents
# a semantic whole:
begin
do_something
and_then_something_else
end
```

but this here bails out with a syntax error:

```
# the following code is grouped together because it represents
# a semantic whole:
{
do_something
and_then_something_else
}
```

In every other language I know that has curly braces as block delimiters, the above is alowed and natural. Except Ruby.

The above code is a special case. I am not sure it can be fixed without breaking Ruby syntax as a whole. I think it can.

To me "curly brackets" and "begin end" should act semantically and syntactically exactly the same.
I.e. all the following forms should IMHO be allowed and identical:

```
if condition
end

if condition begin
end

if condition {
}
```

The same would apply to all other ruby control structures.

Unfortunately it seems it is not be possible to fix the general case without breaking Ruby since Ruby expects that "condition" above could also in itself be a block since both:

```
if { condition }
end

and

if begin condition end
end
```

are allowed and make sense, but contradict my wish above, since allowing the above proposed change would actually make the syntax more ambiguous and make it harder for the parser to help the programmer with syntax errors.
=end

History

#1 - 11/09/2008 05:49 AM - antares (Michael Klishin)

=begin

> In every other language I know that has curly braces as block delimiters, the above is alowed and natural. Except Ruby.

Python does not use curly braces for if statements, and AFAIK nether Python community, nor Ruby community really cares about curly braces for if statements. If you ask me, I'd say curly braces are ugly, so calling it natural is a matter of taste. I don't think that Ruby which historically never tried to be C-like language, really needs to become closer to C.
=end

#2 - 11/09/2008 08:43 PM - matz (Yukihiro Matsumoto)

- Status changed from Open to Rejected

=begin
You shouldn't mention principle of least surprise of YOURSELF in the proposal, for background varies for everyone. You have no rights to change the language (especially in incompatible way) to adopt your personal preference before accustoming yourself to the language, unless you're the creator of the language. Even Ruby surprised you, it is so for good reasons.

If you use Ruby for a while (e.g. at least a year of so), and you still feel it should be changed for good reasons, come again.

```
matz.
```

=end

#3 - 11/11/2008 01:09 AM - cout (Paul Brannan)

=begin
On Sun, Nov 09, 2008 at 05:08:29AM +0900, Tomas Pospisek wrote:

> To me "curly brackets" and "begin end" should act semantically and syntactically exactly the same.

Except for precedence, the curly brackets are the same as do..end, not begin..end.

I think the real question is why do and begin are different. This confused me when I first started using ruby, but now I am used to it.

Paul

=end