Ruby - Bug #16169

rescue in a method argument

09/17/2019 11:38 AM - zverok (Victor Shepelev)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:		**Backport:**	2.5: UNKNOWN, 2.6: UNKNOWN

Description

At 2.4, #12686 was introduced, allowing code like this:

```
foo (Integer(ENV['FOO']) rescue nil)
```

Though, I noticed that in current Ruby correctness of this syntax depends on space after the method name:

```
foo (Integer(ENV['FOO']) rescue nil) # => OK

foo(Integer(ENV['FOO']) rescue nil)
# SyntaxError ((irb):4: syntax error, unexpected modifier_rescue, expecting ')')
# foo(Integer(ENV['FOO']) rescue nil)
#                         ^~~~~~
```

I wonder, whether it is just a bug or a parser limitation (I can't guess which ambiguity the space-less version produces, but I could be missing something)?..

History

#1 - 09/17/2019 04:09 PM - Dan0042 (Daniel DeLorme)

This is the same thing as #16092, where I submitted a documentation patch a little while ago.

#2 - 09/17/2019 06:49 PM - shevegen (Robert A. Heiler)

mame explained that the behaviour is correct (in the context).

Perhaps the documentation could explain that, because I agree
with zverok at the least initially to assume that there should
not be a difference in behaviour between the two. At the least
I think that it would be better to have this explained in the
documentation as-is, similar to how mame explained that.

#3 - 09/17/2019 11:41 PM - mame (Yusuke Endoh)

I agree that Ruby syntax is very subtle. Ruby's parser is a unspeakable monolith of compromise between human intuition and parser limitation.

Consider the following code:

```
foo(x, y, z = ary)
```

It may have two possible interpretations: foo(x, y, (z = ary)) (foo accepts three arguments and the last one is an assignment expression) and foo((x, y, z = ary)) (foo accepts one argument that is a multiple assignment statement). To prevent the second interpretation, only an expression is allowed as an argument. Integer(ENV['FOO']) rescue nil is a statement, so it cannot be written as an argument.

By the way,

- A statement can be converted to an expression by surrounding parentheses: (Integer(ENV['FOO']) rescue nil) is an expression. So you can write foo((Integer(ENV['FOO']) rescue nil)).
- Ruby allows to omit parentheses of method calls: foo x is considered foo(x). So you can write foo (Integer(ENV['FOO']) rescue nil). Note that the space is required to distinguish from a normal method call foo(x).

This is just my understanding. The source code of the parser says "what it does" but doesn't say "why it does (or doesn't)", so we need to surmise. My guess could be wrong.

#4 - 09/18/2019 11:49 AM - zverok (Victor Shepelev)

@mame (Yusuke Endoh)

Ruby allows to omit parentheses of method calls

Oh, thanks! In fact, that was the piece of understanding I missed trying to catch how space- and spaceless-versions are different from PoW of the interpreter.

The ticket could be closed, I just wanted to clarify things a bit.

#5 - 10/02/2019 03:18 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Open to Closed