Ruby trunk - Bug #13939

Ruby 2.4.2 has issue supporting Seattle.rb style for define_method

09/25/2017 03:43 PM - danielpclark (Daniel P. Clark)

Status:	Rejected		
Priority:	Normal		
Assignee:	nobu (Nobuyoshi Nakada)		
Target version:			
ruby -v:	ruby 2.4.2p198 (2017-09-14 revision 59899) [x86_64-linux]	**Backport:**	2.2: UNKNOWN, 2.3: UNKNOWN, 2.4: UNKNOWN

Description

In Ruby 2.3 & 2.4.0 you can do this

```
define_method :some_method_name { "asdf" }
```

As of my trying 2.4.2 this is no longer supported. Anyone who tries to load my 'read_source' gem will get a failure message in require without explaining the specific area of code.

```
SyntaxError: /home/danielpclark/dev/read_source/test/support/example.rb:18: syntax error, unexpect
ed '{', expecting keyword_end
ethod :also_attr_method_name { "asdf" }
                             ^
/home/danielpclark/dev/read_source/test/support/example.rb:18: syntax error, unexpected '}', expec
ting keyword_end
so_attr_method_name { "asdf" }
                              ^
    from (irb):2:in `require_relative'
    from (irb):2
    from /usr/share/rvm/rubies/ruby-2.4.2/bin/irb:11:in `<main>'
```

To fix my gem for Ruby 2.4.2 I now have to wrap the method name in parenthesis ()

```
define_method(:some_method_name) { "asdf" }
```

And that will work. But I find this odd as the following will still work in 2.4.2:

```
define_method :johny, instance_method(:apple)
```

Which is still Seattle.rb style.

Related issues:

Is duplicate of Ruby trunk - Bug #13898: Block parsing regression	**Rejected**
Has duplicate Ruby trunk - Bug #13976: SyntaxError if curly brace block follo...	**Closed**

History

#1 - 09/25/2017 06:15 PM - shevegen (Robert A. Heiler)

Let's ignore the "Seattle.rb style" for the moment because I don't know of such a thing - may well be seattle-duck style. :P

But anyway, I agree with you in at the least one point, which is the warning message:

> Anyone who tries to load my 'read_source' gem will get a failure message in require without explaining the specific area of code.

Perhaps the error message could be more indicative of the error or what the exact problem is or how to solve it. I understand that the () provide additional information that is in some way useful to the parser, or whatever is responsible, so perhaps the message above could be changed somewhat.

I guess ruby core prefers short messages when possible (aka "syntax error, unexpected '}'") but this is indeed not always extremely helpful. With more recent changes such as the did-you-mean-gem, but also some other discussions about better and more fine-tuned

control over warnings/error messages, that may be more helpful to the average ruby hacker.

To the issue about differential parsing, that is actually indeed strange.

Even more surprising is that I actually thought that:

```
define_method(:some_method_name) { "asdf" }
```

Is the only way to use define_method() :D

I think I used that always ...

It reminds me a bit of:

```
get '/' do
  play_intro_music
end
```

versus

```
get '/' { play_intro_music }
# which does not work
```

versus

```
get('/') { play_intro_music }
```

which works. I always thought that in some cases the ruby parse needs the (). I'd love to be able to make them optional in the second case. Oddly enough, for method DEFINITIONS, I always use () when there are arguments, so "def foo(bar)" is what I prefer over "def foo bar". This is more an aside though, it is strange that this changed.

By the way your last example does not work as-is :D

```
define_method :johny, instance_method(:apple)
```

NoMethodError: undefined method `instance_method' for main:Object

Sorry for nitpicking there, I understand what you mean.

I guess it may be because of {} having different meanings in ruby but then again, there was probably some reason unless it was an accident. It's ~3:15 in Tokyo so I guess in a few hours perhaps some from the ruby core team can chime in. I did not even know that we could omit the () there, that was awesome if that was possible. I remember in my code though, that whenever I used define_method(), I always used () ... usually in combination with some *eval* method thing.

#2 - 09/25/2017 09:10 PM - asterite (Ary Borenszweig)

I think this was introduced in https://github.com/ruby/ruby/commit/9987109

Basically, any call foo arg { } when arg is a literal now gives an error.

I think this is a huge breaking change, for example if you had something like:

```
describe "foo" {
  it "does something" {
  }
}
```

Now it no longer works, though I don't know if someone writes code like that.

Maybe the error makes sense because { binds the thing to its left as a block, but I don't know :-)

#3 - 09/25/2017 09:19 PM - danielpclark (Daniel P. Clark)

> Perhaps the error message could be more indicative of the error or what the exact problem is or how to solve it. I understand that the () provide additional information

> that is in some way useful to the parser, or whatever is responsible, so perhaps the message above could be changed somewhat.
>
> I guess ruby core prefers short messages when possible (aka "syntax error, unexpected '}'") but this is indeed not always extremely helpful. With more recent changes such as the did-you-mean-gem, but also some other discussions about better and more fine-tuned control over warnings/error messages, that may be more helpful to the average ruby hacker.

I apologize I wrote that earlier with a different error message. Here's what importing 'read_source' does in Ruby 2.4.2 (but will work for any other version of Ruby).

```
/usr/share/rvm/rubies/ruby-2.3.5/lib/ruby/site_ruby/2.3.0/rubygems/core_ext/kernel_require.rb:55:in `require':
 cannot load such file -- read_source (LoadError)
    from /usr/share/rvm/rubies/ruby-2.3.5/lib/ruby/site_ruby/2.3.0/rubygems/core_ext/kernel_require.rb:55:in `
require'
    from /home/danielpclark/dev/faster_path/test/monkeypatches/faster_path_test.rb:3:in `<top (required)>'
    from /usr/share/rvm/rubies/ruby-2.3.5/lib/ruby/site_ruby/2.3.0/rubygems/core_ext/kernel_require.rb:55:in `
require'
    from /usr/share/rvm/rubies/ruby-2.3.5/lib/ruby/site_ruby/2.3.0/rubygems/core_ext/kernel_require.rb:55:in `
require'
    from /home/danielpclark/.rvm/gems/ruby-2.3.5/gems/rake-12.0.0/lib/rake/rake_test_loader.rb:15:in `block in
 <main>'
    from /home/danielpclark/.rvm/gems/ruby-2.3.5/gems/rake-12.0.0/lib/rake/rake_test_loader.rb:4:in `select'
    from /home/danielpclark/.rvm/gems/ruby-2.3.5/gems/rake-12.0.0/lib/rake/rake_test_loader.rb:4:in `<main>'
rake aborted!
Command failed with status (1)
/home/danielpclark/.rvm/gems/ruby-2.3.5/gems/rake-12.0.0/exe/rake:27:in `<top (required)>'
/home/danielpclark/.rvm/gems/ruby-2.3.5/bin/ruby_executable_hooks:15:in `eval'
/home/danielpclark/.rvm/gems/ruby-2.3.5/bin/ruby_executable_hooks:15:in `<main>'
```

I wasn't going to include it here because I think that problem should be a different bug report. The same error will occur for gem 'method_source' and only for Ruby 2.4.2. I'm not sure this require issue is a Ruby bug yet. It looks like it could be RVM not using the right gem set with Ruby 2.4.2.

The error shown in this bug report is not from being required but from running the test suite in the gem.

> By the way your last example does not work as-is :D

Ah, for that last example there should be a method defined within a class.

```
class Example
  def apple
    "Johny Apple Seed was here"
  end

  define_method :johny, instance_method(:apple)
end
```

#4 - 09/25/2017 09:30 PM - nobu (Nobuyoshi Nakada)

- Is duplicate of Bug #13898: Block parsing regression added

#5 - 09/25/2017 09:39 PM - nobu (Nobuyoshi Nakada)

It's a bug in 2.4.0 and 2.4.1 only.
Braces just after a literal has caused a syntax error before, like as:

danielpclark (Daniel P. Clark) wrote:

> I apologize I wrote that earlier with a different error message. Here's what importing 'read_source' does in Ruby 2.4.2 (but will work for any other version of Ruby).
>
> ```
> /usr/share/rvm/rubies/ruby-2.3.5/lib/ruby/site_ruby/2.3.0/rubygems/core_ext/kernel_require.rb:55:in `requi
> re': cannot load such file -- read_source (LoadError)
> ```

A brace block has higher precedence and is bound to the previous expression, and a literal cannot be a method call and have a block.

shevegen (Robert A. Heiler) wrote:

> I guess ruby core prefers short messages when possible (aka "syntax error, unexpected '}'") but this is indeed not always extremely helpful.

Unfortunately, this message is generated by bison, not us.

#6 - 10/05/2017 08:01 AM - shyouhei (Shyouhei Urabe)

- Has duplicate Bug #13976: SyntaxError if curly brace block follows args without parentheses, introduced in 2.4.2 added

#7 - 11/10/2017 08:01 AM - hsbt (Hiroshi SHIBATA)

- Assignee set to nobu (Nobuyoshi Nakada)

- Status changed from Open to Rejected