Ruby master - Bug #12295

Ripper not emitting on_parse_error for global variable name syntax errors

04/17/2016 07:48 PM - lsegal (Loren Segal)

Status:	Rejected		
Priority:	Normal		
Assignee:	aamine (Minero Aoki)		
Target version:			
ruby -v:	ruby 2.3.0p0 (2015-12-25 revision 53290) [x86_64-linux-gnu]	**Backport:**	2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: UNKNOWN

Description

Ripper is not emitting the on_parse_error event for certain types of syntax errors, specifically for the following snippet of code:

```
:~$
```

Here is the Ruby syntax error:

```
$ ruby -v
ruby 2.3.0p0 (2015-12-25 revision 53290) [x86_64-linux-gnu]
$ ruby -e ':~$'
-e:1: `$' without identifiers is not allowed as a global variable name
```

Here is the expected Ripper result for a syntax error emitting on_parse_error:

```
$ ruby -rripper -e 'class X < Ripper; def on_parse_error(m) puts "ERROR #{m}" end end; X.parse "cl
ass;end"'
ERROR syntax error, unexpected ';'
```

Here is reproduction for the omitted on_parse_error event:

```
$ ruby -rripper -e 'class X < Ripper; def on_parse_error(m) puts "ERROR #{m}" end end; X.parse ":~
$"'
$
```

Note the lack of ERROR message. The expectation is that this parse error will emit the on_parse_error event in Ripper.

I reproduced this in Ruby 2.3.0, I have not tested 2.2 or dev, so I don't know if it needs porting, though I imagine it does.

History

#1 - 04/17/2016 07:54 PM - lsegal (Loren Segal)

After looking into this a little more it looks like the Ruby error is not a "parse error", though it probably should be? This might no longer be Ripper specific. Ruby just seems to give up on the above code no matter where it is placed in source:

```
$ ruby -e 'puts "one"; :~$ ; puts "two"'
-e:1: `$' without identifiers is not allowed as a global variable name
```

I would expect that if there is no syntax error that Ruby should print "one\ntwo".

#2 - 04/17/2016 09:55 PM - usa (Usaku NAKAMURA)

You can handle the case by using compile_error instead of on_parse_error as below:

```
$ ruby -rripper -e 'class X < Ripper; def compile_error(m) puts "ERROR #{m}" end end; X.parse ":~$"'
ERROR `$' without identifiers is not allowed as a global variable name
```

IMO, ripper expects that all syntax errors are "compile errors", not "parse errors".
But current implementation of the ruby parser is not so.
Then, we must define both on_parse_error and compile_error to handle syntax errors at this time.

#3 - 06/13/2016 07:06 AM - shyouhei (Shyouhei Urabe)

- Assignee set to aamine (Minero Aoki)

- Status changed from Open to Assigned

#4 - 06/13/2016 07:11 AM - aamine (Minero Aoki)

Ripper is just a thin wrapper over ruby parser, so the original parser differentiate them, ripper just inherits difference directly.
Extra work such as unifying two errors should be done by upper layer.

#5 - 06/13/2016 07:30 AM - aamine (Minero Aoki)

- Status changed from Assigned to Closed

#6 - 08/04/2016 06:07 AM - usa (Usaku NAKAMURA)

- Status changed from Closed to Rejected