Ruby - Bug #19877

Non intuitive behavior of syntax only applied to literal value

09/13/2023 09:22 AM - tompng (tomoya ishida)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 3.3.0dev (2023-09-08T23:08:32Z master b635a66e95) [x86_64-linux]	**Backport:**	3.0: UNKNOWN, 3.1: UNKNOWN, 3.2: UNKNOWN

Description

Non intuitive behavior of syntax only applied to literal value

Some ruby syntax is only applied to literal value.

```
def 1.foo; end # receiver is a literal, it is Syntax Error
/(?<a>)/ =~ s # receiver is regexp literal, it will assign to local variable
if cond1..cond2; end # range-like syntax appears in condition, it is flipflop
```

If it is wrapped with parenthesis, the behavior seems not intuitive for me, and YARP parses it differently.

```
def (1).foo; end # Syntax Error
def ((1;1)).foo; end # Syntax Error
def ((;1)).foo; end # Syntax OK
def ((1+1;1)).foo; end # Syntax OK
def ((%s();1)).foo; end # Syntax Error
def ((%w();1)).foo; end # Syntax OK
def ("#{42}").foo; end # Syntax Error
def (:"#{42}").foo; end # Syntax OK

(/(?<a>)/) =~ s # assigns to a
(;/(?<a>)/) =~ s # does not assigns
(%s();/(?<a>)/) =~ s # assigns to a
(%w();/(?<a>)/) =~ s # does not assigns
(1; (2; 3; (4; /(?<a>)/))) =~ s # assigns to a
(1+1; /(?<a>)/) =~ s # does not assign

if ((cond1..cond2)); end # flipflop
if (; cond1..cond2); end # range
if (1; cond1..cond2); end # flipflop
if (%s(); cond1..cond2); end # flipflop
if (%w(); cond1..cond2); end # range
if (1; (2; (3; 4; cond1..cond2))); end # flipflop
if (1+1; cond1..cond2); end # range
```

I expect YARP and parse.y parses same.
I expect all parenthesis-wrapped result same.
I think it is simple and intuitive if parenthesis-wrapped code always behaves different from non-wrapped code because there are more complex variation like this

```
def (class A; 1; end).foo; end
(break; /?<a>/) =~ s
class A; /?<a>/; end =~ s
```

Associated revisions

Revision e8896a31d48e5797df3878696dcb50aed85b87c2 - 09/14/2023 04:09 PM - nobu (Nobuyoshi Nakada)

[Bug #19877] Literals cannot have singleton methods even in blocks

Revision 864bb8680cee48a2bed85703dc2e4070728362d4 - 09/14/2023 04:09 PM - nobu (Nobuyoshi Nakada)

[Bug #19877] Named captures should take place from regexps in block

Revision e8896a31d48e5797df3878696dcb50aed85b87c2 - 09/14/2023 04:09 PM - nobu (Nobuyoshi Nakada)

[Bug #19877] Literals cannot have singleton methods even in blocks

Revision 864bb8680cee48a2bed85703dc2e4070728362d4 - 09/14/2023 04:09 PM - nobu (Nobuyoshi Nakada)

[Bug #19877] Named captures should take place from regexps in block

Revision e8896a31 - 09/14/2023 04:09 PM - nobu (Nobuyoshi Nakada)

[Bug #19877] Literals cannot have singleton methods even in blocks

Revision 864bb868 - 09/14/2023 04:09 PM - nobu (Nobuyoshi Nakada)

[Bug #19877] Named captures should take place from regexps in block

Revision 9cb33aad55e0a2cded3b09b0509b9c53fb0fa5ae - 09/14/2023 04:09 PM - nobu (Nobuyoshi Nakada)

[Bug #19877] Fix flip-flop in block

Revision 9cb33aad55e0a2cded3b09b0509b9c53fb0fa5ae - 09/14/2023 04:09 PM - nobu (Nobuyoshi Nakada)

[Bug #19877] Fix flip-flop in block

Revision 9cb33aad - 09/14/2023 04:09 PM - nobu (Nobuyoshi Nakada)

[Bug #19877] Fix flip-flop in block

Revision 1802d14ca8924bd67e0915c5ad9f1fad5dba0602 - 11/30/2023 12:40 PM - nobu (Nobuyoshi Nakada)

[Bug #19877] Assign captures for direct regexp literal only

Revision 1802d14ca8924bd67e0915c5ad9f1fad5dba0602 - 11/30/2023 12:40 PM - nobu (Nobuyoshi Nakada)

[Bug #19877] Assign captures for direct regexp literal only

Revision 1802d14c - 11/30/2023 12:40 PM - nobu (Nobuyoshi Nakada)

[Bug #19877] Assign captures for direct regexp literal only

Revision 9b7a964318d04beb82680ff1d0c6eb571f4b8b5e - 12/08/2023 03:53 AM - nobu (Nobuyoshi Nakada)

[Bug #19877] Flip-flop needs to be direct condition

Revision 9b7a964318d04beb82680ff1d0c6eb571f4b8b5e - 12/08/2023 03:53 AM - nobu (Nobuyoshi Nakada)

[Bug #19877] Flip-flop needs to be direct condition

Revision 9b7a9643 - 12/08/2023 03:53 AM - nobu (Nobuyoshi Nakada)

[Bug #19877] Flip-flop needs to be direct condition

History

#1 - 09/14/2023 05:29 PM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

Applied in changeset git|e8896a31d48e5797df3878696dcb50aed85b87c2.

[Bug #19877] Literals cannot have singleton methods even in blocks

#2 - 11/27/2023 06:18 PM - kddnewton (Kevin Newton)

We need an answer on this in order to properly understand what we need to support. In particular,

```
(1; /(?<foo>)/) =~ s
```

assigning to the local seems incorrect. It would mean the only way for a parser (prism or any other) to be correct would be to mirror CRuby dropping literal nodes from the tree. In the meantime I know @yui-knk (Kaneko Yuichiro) has been working on adding those nodes back in for the universal parser.

So either way, we need to know what is "correct" in this case.

#3 - 11/30/2023 10:17 AM - nobu (Nobuyoshi Nakada)

https://github.com/ruby/ruby/pull/9077 ?

#4 - 12/01/2023 03:34 PM - kddnewton (Kevin Newton)

Thank you @nobu (Nobuyoshi Nakada) that fixes the regex, but we still need it for flip-flops:

https://github.com/ruby/prism/issues/1923

```
RubyVM::AbstractSyntaxTree.parse 'if (1; a..b); end'
# =>
(SCOPE@1:0-1:17
 tbl: []
 args: nil
 body: (IF@1:0-1:17 (FLIP2@1:7-1:11 (VCALL@1:7-1:8 :a) (VCALL@1:10-1:11 :b)) (BEGIN@1:13-1:13 nil) nil))
```

#5 - 12/02/2023 09:58 AM - tompng (tomoya ishida)

I found a similar behavior for MATCH node (Prism::MatchLastLineNode)

```
RubyVM::AbstractSyntaxTree.parse 'if (1; //); end'
(none):1: warning: regex literal in condition
=>
(SCOPE@1:0-1:15
 tbl: []
 args: nil
 body:
   (IF@1:0-1:15
      (BEGIN@1:3-1:10
         (BLOCK@1:4-1:9 (LIT@1:4-1:5 1) (MATCH@1:7-1:9 //)))
      (BEGIN@1:11-1:11 nil) nil))
```

#6 - 12/07/2023 10:04 AM - mame (Yusuke Endoh)

In today's dev meeting, @matz (Yukihiro Matsumoto) said:

```
expr if (cond1..cond2)                # This should be a flip-flop
expr if (((cond1..cond2)))            # This should be a flip-flop
expr if begin cond1..cond2; end       # This should be a flip-flop
expr if (;;; cond1..cond2)            # Don't care. Either is ok
expr if (1; cond1..cond2)             # This should NOT be a flip-flop
expr if (method; cond1..cond2)        # This should NOT be a flip-flop

(111; /(?<a>)/) =~ s; # a should NOT be defined
(foo; /(?<a>)/) =~ s; # a should NOT be defined
(/(?<a>)/) =~ s       # a should NOT be defined
```