Ruby master - Feature #10463

`:~@` and `!:@` are not parsed correctly

11/01/2014 02:13 AM - sawa (Tsuyoshi Sawada)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>matz (Yukihiro Matsumoto)</td>
</tr>
<tr>
<td>Target version:</td>
<td></td>
</tr>
</tbody>
</table>

**Description**
The at mark in literal symbols `:~@` and `!:@` are ignored.

```
:~@ # => :~
!:@ # => :!
```

**Related issues:**
Has duplicate Ruby master - Bug #18246: send does not work for unary `!` operat... Closed

**History**

#1 - 11/01/2014 02:37 AM - silverhammermba (Max Anselm)
That's because bash is trying to interpolate the string.

```bash
$ ruby -e 'p :!@'
!:!
```

#2 - 07/09/2019 03:58 AM - jeremyevans0 (Jeremy Evans)
I did some research, and this is related to the fact that Ruby allows `!@` and `~@` as method names (back to the initial SVN revision for `~@`), silently dropping the `@` from the method name:

```ruby
class A
  def !@; @ end
  def ~@; @ end
end

!A.new
# :!
~A.new
# :~
A.instance_methods(false)
# => [:!, :~]
```

This diff would remove this behavior:

```diff
diff --git a/parse.y b/parse.y
index 33f1ef072e..ed7dae8955 100644
--- a/parse.y
+++ b/parse.y
@@ -8772,6 +8772,7 @@ parser_yylex(struct parser_params *p)
     if (IS_AFTER_OPERATOR()) {
       SET_LEX_STATE(EXPR_ARG);
       if (c == ' ') {
-         pushback(p, c);
+         pushback(p, c);
           return '!';
       }
     }
@@ -9184,9 +9185,6 @@ parser_yylex(struct parser_params *p)
     case '-':
       if (IS_AFTER_OPERATOR()) {
         if ((c = nextc(p)) != '0') {
-           pushback(p, c);
         }
         SET_LEX_STATE(EXPR_ARG);
       } else {
```

However, it breaks using `~@` and `!@` as method names, which is breaks one test in bootstraptest. This is because both the symbol and the method
name use fname in the parser. There is probably a way to remove support for using these in symbols but keeping the support in method names that I am not currently aware of. However, I'm sure if we want to keep supporting ~@ and !@ in method names.

FWIW, JRuby handles :~@ and :!@ the same way as CRuby.

#3 - 07/09/2019 03:30 PM - nobu (Nobuyoshi Nakada)
If :!@ and :) are different things, also !foo and foo.!) are different things. This means a backward incompatibility.

#4 - 07/09/2019 03:56 PM - jeremyevans0 (Jeremy Evans)
nobu (Nobuyoshi Nakada) wrote:

If :!@ and :) are different things, also !foo and foo.!) are different things.

I don't believe that is true. With the above patch:

```ruby
class A
  def !; :) end
  def ~; :) end
end

A.new
# :!
A.new.
# :)
~A.new
# :~
A.new.
# :~
```

The @ in :@ and def @; end was ignored during lexing before, it doesn't affect the semantics. This is different than +@ and +:

```ruby
class A
  def +; :+: end
  def +@; :+: end
end

+A.new
# :+:
A.new.
# :+:
```

This means a backward incompatibility.

The backward incompatibility should be limited to making the following invalid syntax:

```ruby
def !@; end
def ~@; end
alias foo !@
alias bar ~@
!:@
:@
```

Currently, :@ is different than :!@", which very much appears to be a bug. With the above patch :@ is a syntax error (:"!@" is still valid).

#5 - 07/09/2019 04:40 PM - nobu (Nobuyoshi Nakada)
jeremyevans0 (Jeremy Evans) wrote:

nobu (Nobuyoshi Nakada) wrote:

If :!@ and :) are different things, also !foo and foo.!) are different things.

I don't believe that is true. With the above patch:

Sorry, forgot @, I wanted to mean !foo and foo.!:@.

#6 - 07/09/2019 04:53 PM - jeremyevans0 (Jeremy Evans)
nobu (Nobuyoshi Nakada) wrote:

Jeremy Evans wrote:

nobu (Nobuyoshi Nakada) wrote:

If :!@ and :! are different things, also !foo and foo.! are different things.

I don't believe that is true. With the above patch:

Sorry, forgot @, I wanted to mean !foo and foo.!@.

Well, foo.(@ would be a syntax error with the patch. Is there a reason other than backwards compatibility to keep this automatic aliasing of !@ to ! and ~@ to ~? If not, maybe we could deprecate this in 2.7 and remove it in Ruby 3 (if Matz approves).

#7 - 07/10/2019 09:27 AM - shevegen (Robert A. Heiler)
Interesting - I did not know this. sawa finds stuff. :)

Personally I would be in favour of changing the behaviour as Jeremy described (I also think this may be a bug or perhaps an oddity), but I guess it depends on a) whether Matz wants to change it in the first place and b) then when to change it, if a) evaluates to a change.

A secondary consideration may be to query how many Ruby users depend on this backwards behaviour.

I have no statistical dataset to make a statement either way, but from intuition, I would venture that this change would not affect a lot of Ruby code out there; I could be wrong though.

#8 - 07/18/2019 07:22 PM - Jeremy Evans

-Matzo confirmed in the last developer meeting that he wants def !@; end to continue to work. However, I still think we should fix it so that :!@ and :~@ are not treated as :! and :~. This is a bit tricky to do as the parser currently uses the same lex state for both cases (EXPR_FNAME).

The simplest way I can think to fix this is in the attached patch. It changes the :! and :~ cases to use lex state EXPR_FNAME|EXPR_FITEM, and changes the code to check on whether EXPR_FITEM is one of the lex states. If EXPR_FITEM is one of the lex states, then it doesn't ignore the @.

#9 - 07/30/2019 07:40 AM - Koichi Sasada

- Assignee set to nobu (Nobuyoshi Nakada)

#10 - 09/02/2019 06:30 AM - Koichi Sasada

- Tracker changed from Bug to Feature

- Assignee changed from nobu (Nobuyoshi Nakada) to Matz (Yukihiro Matsumoto)

- ruby -v deleted (2.1.4)

- Backport deleted (2.0.0: UNKNOWN, 2.1: UNKNOWN)

#11 - 12/20/2019 08:12 AM - Matz (Yukihiro Matsumoto)

- Status changed from Open to Rejected

I don't see the practical benefit of this proposal. Besides that incompatibility is a clear drawback.

Matz.

#12 - 10/07/2021 06:04 PM - Jeremy Evans

- Has duplicate Bug #18246: send does not work for unary ! operator when operator isn't a literal symbol added

Files

| tilda-bang-at-symbols.patch | 2.22 KB | 07/18/2019 | Jeremy Evans |