Ruby trunk - Feature #15606

Precedence of -@ and +@

02/15/2019 09:24 AM - sos4nt (Stefan Schüßler)

Status:	Feedback
Priority:	Normal
Assignee:	
Target version:	

Description

-@ and +@ seem to have issues with method chaining:

```
i = 5

-i.negative?
# NoMethodError (undefined method `-@' for false:FalseClass)
```

here's another example:

```
# frozen_string_literal: true

+'foo'.upcase!
# FrozenError (can't modify frozen String)
```

I know that I can fix this by adding parentheses, i.e. (-i).negative? and (+'foo').upcase! but it feels cumbersome.

Shouldn't the above work out of the box?

Unless I'm missing a crucial use case, the precedence for -@ and +@ should be changed.

History

#1 - 02/15/2019 09:28 AM - sos4nt (Stefan Schüßler)

- Subject changed from Precendence of -@ and +@ to Precedence of -@ and +@

#2 - 02/15/2019 10:23 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Feedback

Do you expect i = -1; -i.abs to return +1?

#3 - 02/15/2019 10:39 AM - sos4nt (Stefan Schüßler)

nobu (Nobuyoshi Nakada) wrote:

> Do you expect i = -1; -i.abs to return +1?

Yes, indeed. I'd expect <something>.abs to return a positive value.

```
 1.abs #=> 1
-1.abs #=> 1

i = 1

 i.abs #=> 1
-i.abs #=> -1  <- not what I'd expect
```

Likewise:

```
i = 1

-1.succ #=> 0
-i.succ #=> -2  <- not what I'd expect
```

And a really weird one:

```
i = 1

-i.to_s #=> "1"  <- this is a frozen string
```

Again, I perfectly understand *why* this is happening and how to avoid it. I just don't think this is the way it should be, nor do I see why it could be preferable.

#4 - 02/16/2019 06:02 AM - spinute (Satoru Horie)

Compatibility is essential in Ruby. So, we need strong evidence when we break it.

I do not think the current behavior is weird.

<something>.abs returns positive value, consistently.

A programmer who knows -i.abs is evaluated as -(i.abs) can get an expected result. As a note, - in -1 is not an operator, so -1.abs == (-1).abs == 1. -i.to_s is also explained in the same way.

I agree that the behavior may be confusing for beginners.