Ruby - Bug #18441

Fix inconsistent parentheses with anonymous block forwarding

12/27/2021 04:31 PM - bkuhlmann (Brooke Kuhlmann)

Status:	Rejected		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 3.1.0p0 (2021-12-25 revision fb4df44d16) [arm64-darwin21.2.0]	**Backport:**	2.6: UNKNOWN, 2.7: UNKNOWN, 3.0: UNKNOWN, 3.1: UNKNOWN

Description

Overview

One of the best qualities of Ruby, syntactically, is the optional use of parenthesis (or lack there of). I've enjoyed this for almost two decades. With Ruby 3.1.0, things have become a bit more inconsistent with the introduction of anonymous block forwarding.

I'm wondering if it would be possible to allow optional parenthesis for this feature? I understand if this can't be done but having consistent behavior would be most welcome.

Steps to Recreate

To demonstrate, consider the following code:

```
def demo positional, &block
  other positional, &block
end
```

Notice parentheses are optional. If saving the above as snippet.rb and running as ruby snippet.rb, there will be syntax errors. ...but if you modify the above implementation to use anonymous block forwarding as follows:

```
def demo positional, &
  other positional, &
end
```

...and then run the above as ruby snippet.rb, you'll end up with the following syntax error:

```
snippet.rb:16: syntax error, unexpected local variable or method, expecting ';' or '\n'
  other positional, &
```

In order to fix the error, you'll need to modify the implementation as follows:

```
def demo(positional, &)
  other positional, &
end
```

Should you need to send multiple messages to similar methods within your implementation, you'll be forced to use parentheses within the body as well. Example:

```
def demo(positional, &)
  other(positional, &)
  another(positional, &)
end
```

It seems odd that the style of code you write needs to differ based on whether you use anonymous or explicit use of block syntax.

Notes

If additional context is helpful, this behavior is similar in nature to an issue with punning as described in this issue.

Anyway, thanks and am enjoying the new Ruby 3.1.0 features! □□

History

#1 - 12/27/2021 04:53 PM - jeremyevans0 (Jeremy Evans)

I don't think this is a bug. The issue with not using parentheses is that & is going to keep looking for the name for the block variable:

```
def a &
  b
  b
end
```

is parsed as:

```
def a(&b)
  b
end
```

So your code:

```
def demo positional, &
  other positional, &
end
```

is parsed as:

```
def demo(positional, &other positional, &)
end
```

Which isn't valid syntax. I'm not sure how we could keep backwards compatibility and also allow for & to work the way you want. Note that it is already possible to have anonymous block forwarding work without parentheses, just not exactly the way you want:

```
def demo positional, &;
  other positional, &
end

def demo positional, &;
  other positional, &;
  another positional, &
end
```

Note that anonymous block parameter parsing is consistent with anonymous rest and keyword rest parameter parsing from previous versions of Ruby:

```
def a *
  b, **
  c, &
  d
  [b, c, d]
end

a(1, b: 2){}
# => [[1], {:b=>2}, #<Proc:0x00000aa8c0582140 (irb):22>]
```

#2 - 12/27/2021 08:16 PM - bkuhlmann (Brooke Kuhlmann)

> It is already possible to have anonymous block forwarding work without parentheses, just not exactly the way you want:

True, but use of semicolon is definitely not desired in this situation even though that would get me slightly closer to what I want.

> Note that anonymous block parameter parsing is consistent with anonymous rest and keyword rest parameter parsing from previous versions of Ruby

That code example tripped me up, initially, because I'm not used to reading parameters broken up across multiple lines (especially with oddly placed newlines) but I see your point. There are times where you might have arguments span multiple lines which is also valid. Even use of a carriage return wouldn't help so I guess there's not an easy way to parse when a method definition stops and starts or when arguments in a message start and stop without the use parenthesis or semicolons. Alas. □□

Thanks for the explanation and clarification. □□□□□□ I'll have to fall back to letting Rubocop auto-correct for me when I veer astray. I don't think I can close this issue but feel free to do so when you have a chance.

#3 - 12/27/2021 08:28 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Open to Rejected