Ruby master - Bug #16642
Splatted empty hash literal produces frozen hash object
02/19/2020 06:22 PM - Dan0042 (Daniel DeLorme)

Status: Open
Priority: Normal
Assignee: 
Target version: 
ruby -v: 
Backport: 2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN

Description
When splatting an empty hash literal, internally it's optimized using a global frozen hash object, but this implementation detail can leak into the ruby code outside:

```ruby
ruby2_keywords def foo(*a) a.last end
h = foo(**{})
h[1] = 2
# can't modify frozen Hash: {} (FrozenError)
```

I think this can be considered a bug?

History
#1 - 02/20/2020 12:04 AM - jeremyevans0 (Jeremy Evans)
I agree it is a bug. I'm not sure it is worth fixing. Basically, the reason behind it is the parser doesn't separate hash compilation from keyword argument compilation, and the optimization for `**{}` as the only keyword argument ends up applying to hashes as well. It is simple to remove the optimization, but it will result in worse performance for keyword arguments using `**{}` (which can be used to avoid keyword argument warnings in Ruby 2.7). Separate parsing and/or compilation of keyword arguments and hashes will fix this, and it is on my todo list.

#2 - 02/20/2020 12:10 AM - jeremyevans0 (Jeremy Evans)
Actually, looks like I didn't read the bug report closely enough. This is a different issue, and suggests that we should either recognize the frozen hash in the ruby2_keywords case and dup it, or remove the optimization.

The issue I was referring to in my last comment is the fact that `{**{}}.frozen?` is true, due to the same optimization.