Ruby master - Bug #10856

Splat with empty keyword args gives unexpected results

02/15/2015 08:47 PM - seantheprogrammer (Sean Griffin)

Status:	Closed		
Priority:	Normal		
Assignee:	nobu (Nobuyoshi Nakada)		
Target version:	2.7		
ruby -v:	ruby 2.2.0p0 (2014-12-25 revision 49005) [x86_64-darwin13]	**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN

Description

When keyword args are passed to a method with splat, and there are no keyword args, an empty hash is sent. I would expect no argument to be given, same as splat with an empty array. For example:

```
def foo
end

foo(**{})
```

This causes an argument error, as an empty hash is passed. I would expect the same behavior as

```
def foo
end

foo(*[])
```

Related issues:

Related to Ruby master - Bug #10719: empty splatting literal hash after other...	**Closed**	**01/09/2015**
Related to Ruby master - Bug #13791: `belongs_to': unknown keywords: required...	**Closed**	
Related to Ruby master - Bug #13793: Compatible issue with keyword args behavior	**Closed**	
Related to Ruby master - Feature #14183: "Real" keyword argument	**Closed**	
Related to Ruby master - Bug #15078: Hash splat of empty hash should not crea...	**Closed**	
Has duplicate Ruby master - Misc #11131: Unexpected splatting of empty kwargs	**Closed**	
Has duplicate Ruby master - Bug #13717: Calling lambda with keyword arguments...	**Closed**	

Associated revisions

Revision 26aed9c5 - 08/05/2017 06:58 AM - nobu (Nobuyoshi Nakada)

splat keyword hash

- compile.c (compile_array_keyword_arg): set keyword splat flag if explicitly splatted. [ruby-core:68124] [Bug #10856]
- vm_args.c (setup_parameters_complex): try keyword hash splat if given.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@59519 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 59519 - 08/05/2017 06:58 AM - nobu (Nobuyoshi Nakada)

splat keyword hash

- compile.c (compile_array_keyword_arg): set keyword splat flag if explicitly splatted. [ruby-core:68124] [Bug #10856]
- vm_args.c (setup_parameters_complex): try keyword hash splat if given.

Revision 59519 - 08/05/2017 06:58 AM - nobu (Nobuyoshi Nakada)

splat keyword hash

- compile.c (compile_array_keyword_arg): set keyword splat flag if explicitly splatted. [ruby-core:68124] [Bug #10856]
- vm_args.c (setup_parameters_complex): try keyword hash splat if given.

Revision 59519 - 08/05/2017 06:58 AM - nobu (Nobuyoshi Nakada)

splat keyword hash

- compile.c (compile_array_keyword_arg): set keyword splat flag if explicitly splatted. [ruby-core:68124] [Bug #10856]
- vm_args.c (setup_parameters_complex): try keyword hash splat if given.

Revision d49fca88 - 11/02/2017 07:52 AM - nobu (Nobuyoshi Nakada)

compile.c: kw splat after splat

- compile.c (setup_args): set keyword splat flag after splat arguments. [ruby-core:83638] [Bug #10856]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@60613 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 60613 - 11/02/2017 07:52 AM - nobu (Nobuyoshi Nakada)

compile.c: kw splat after splat

- compile.c (setup_args): set keyword splat flag after splat arguments. [ruby-core:83638] [Bug #10856]

Revision 60613 - 11/02/2017 07:52 AM - nobu (Nobuyoshi Nakada)

compile.c: kw splat after splat

- compile.c (setup_args): set keyword splat flag after splat arguments. [ruby-core:83638] [Bug #10856]

Revision 60613 - 11/02/2017 07:52 AM - nobu (Nobuyoshi Nakada)

compile.c: kw splat after splat

- compile.c (setup_args): set keyword splat flag after splat arguments. [ruby-core:83638] [Bug #10856]

Revision 3db2041f - 03/14/2019 09:04 AM - mame (Yusuke Endoh)

compile.c: fix the corner case of rest and keyword arguments

See https://bugs.ruby-lang.org/issues/10856#note-20 . [Bug #10856]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@67256 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 67256 - 03/14/2019 09:04 AM - mame (Yusuke Endoh)

compile.c: fix the corner case of rest and keyword arguments

See https://bugs.ruby-lang.org/issues/10856#note-20 . [Bug #10856]

History

#1 - 02/17/2015 05:36 AM - shugo (Shugo Maeda)

- Related to Bug #10719: empty splatting literal hash after other keywords causes SEGV added

#2 - 02/17/2015 05:40 AM - shugo (Shugo Maeda)

- Status changed from Open to Closed

Sean Griffin wrote:

> When keyword args are passed to a method with splat, and there are no keyword args, an empty hash is sent. I would expect no argument to be given, same as splat with an empty array. For example:

It was fixed in r49193.

#3 - 03/05/2015 03:31 PM - seantheprogrammer (Sean Griffin)

It looks like this bug still exists in 2.2.1, and was only fixed when splatting a hash literal. The following code is still broken:

```
def foo
end

h = {}
foo(**h)
```

#4 - 05/11/2015 05:12 AM - nobu (Nobuyoshi Nakada)

- Has duplicate Misc #11131: Unexpected splatting of empty kwargs added

#5 - 05/11/2015 07:20 AM - nobu (Nobuyoshi Nakada)

- Status changed from Closed to Open

#6 - 05/14/2015 07:28 AM - matz (Yukihiro Matsumoto)

It's because ** tries to pass keyword hash (this caes empty) as an argument, so that old style

```
  def foo(h)
  end
  foo(**{})
```

to work. In anyway, passing keyword arguments to a method that does not take any keyword argument can cause exception.
If you have real-world use-case, let us know.

Matz.

#7 - 08/10/2015 12:27 PM - teamon (Tymon Tobolski)

Hi Matz,

I think I just found a real-world use-case for exactly this issue - please take a look at the (simplified) example below.

```
class Dispatcher
  def call(event, args)
    public_send(event, **args)
  end

  def first_event(myarg:)
    # ...
  end

  def second_event
    # ...
  end
end
```

And the call site looks like this:

```
disp = Dispatcher.new
disp.call(params[:event], params[:args])
```

Then we can observe:

```
disp.call(:first_event, {myarg: 123}) # => passes correctly, all good

disp.call(:first_event, {})                          # => missing keyword: myarg - exactly what I'd expect
disp.call(:first_event, {myarg: 123, other: "foo"})
# => unknown keyword: other - exactly what I'd expect

disp.call(:second_event, {})
# => wrong number of arguments (1 for 0) - this /should/ just pass without error
```

So, in case the params[:args] is empty we would expect to either get a "missing keyword" exception or simply valid method execution when such param is not required.

Please let me know what do you think about it.

#8 - 08/10/2015 02:33 PM - marcandre (Marc-Andre Lafortune)

I feel this has to be fixed.

foo(**{}) should === foo(**Hash.new) in all cases, and I feel it should not raise an error.

#9 - 08/21/2016 06:09 AM - gisborne (Guyren Howe)

I believe this behavior is wrong and should be fixed.

This gets in the way of simple functional programming idioms. eg "Call each of these functions with these args until one doesn't fail"

```
class FnSeries
  def initialize(*fns)
    @fns = fns
  end

  def call(*args, **kwargs)
    @fns.each do |fn|
      begin
        return fn.call(*args, **kwargs)
        rescue Exception => e
     end
  end
end
```

If one of the fns takes no args, this will fail even if that function would otherwise succeed.

#10 - 08/21/2016 03:29 PM - Eregon (Benoit Daloze)

Guyren Howe wrote:

> I believe this behavior is wrong and should be fixed.
>
> This gets in the way of simple functional programming idioms. eg "Call each of these functions with these args until one doesn't fail"

There is a simple fix for your use-case, if you just want to fowrard arguments, don't use ** at all:
(it's not like in Python, keyword arguments are less separated form normal arguments)

```
class FnSeries
  def initialize(*fns)
    @fns = fns
  end

  def call(*args)
    @fns.each do |fn|
      begin
        return fn.call(*args)
        rescue Exception => e
     end
  end
end
```

Marc-Andre Lafortune wrote:

> I feel this has to be fixed.
>
> foo(**{}) should === foo(**Hash.new) in all cases, and I feel it should not raise an error.

I agree, it's highly inconsistent that:

```
def foo(*args); args; end
foo(**{}) # => []
h={}
foo(**h) # => [{}]
foo(h) # => [{}]
```

#11 - 07/05/2017 11:04 AM - nobu (Nobuyoshi Nakada)

- Has duplicate Bug #13717: Calling lambda with keyword arguments inconsistent behavior added

#12 - 07/05/2017 03:21 PM - nobu (Nobuyoshi Nakada)

- Description updated

https://github.com/ruby/ruby/compare/trunk...nobu:bug/10856-splat-hash

#13 - 08/05/2017 06:58 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

Applied in changeset trunk|r59519.

---

splat keyword hash

- compile.c (compile_array_keyword_arg): set keyword splat flag if explicitly splatted. [ruby-core:68124] [Bug #10856]
- vm_args.c (setup_parameters_complex): try keyword hash splat if given.

#14 - 08/09/2017 08:17 AM - nobu (Nobuyoshi Nakada)

- Related to Bug #13791: `belongs_to': unknown keywords: required, anonymous_class (ArgumentError) since Revision 59519 added

#15 - 08/09/2017 04:08 PM - nobu (Nobuyoshi Nakada)

- Related to Bug #13793: Compatible issue with keyword args behavior added

#16 - 11/01/2017 05:02 PM - marcandre (Marc-Andre Lafortune)

- Target version set to 2.5

- Assignee set to nobu (Nobuyoshi Nakada)

- Status changed from Closed to Open

This is not actually fixed.

```
def foo
  puts "OK"
end

options = {}
foo(**options) # => OK  (In 2.5.0preview1)
args = []
foo(*args, **options) # => ArgumentError: wrong number of arguments (given 1, expected 0)
```

The second call should also output "Ok".

Hopefully Nobu can crack this before 2.5.0

#17 - 11/02/2017 07:53 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

Applied in changeset trunk|r60613.

---

compile.c: kw splat after splat

- compile.c (setup_args): set keyword splat flag after splat arguments. [ruby-core:83638] [Bug #10856]

#18 - 12/14/2017 07:24 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #14183: "Real" keyword argument added

#19 - 09/05/2018 04:22 PM - marcandre (Marc-Andre Lafortune)

- Related to Bug #15078: Hash splat of empty hash should not create a positional argument. added

#20 - 03/14/2019 05:06 AM - mame (Yusuke Endoh)

- Target version changed from 2.5 to 2.7

- Status changed from Closed to Assigned

marcandre (Marc-Andre Lafortune) wrote:

> This is not actually fixed.
>
> ```
> def foo
>   puts "OK"
> end
>
> options = {}
> foo(**options) # => OK  (In 2.5.0preview1)
> args = []
> foo(*args, **options) # => ArgumentError: wrong number of arguments (given 1, expected 0)
> ```
>
> The second call should also output "Ok".
>
> Hopefully Nobu can crack this before 2.5.0

This is not completely fixed yet:

```
$ ruby -v
ruby 2.6.0p0 (2018-12-25 revision 66547) [x86_64-linux]
$ ruby -e 'def foo; end; options = {}; args = []; foo(*args, **options)'
$ ruby -e 'def foo(z); end; options = {}; args = []; foo(*args, 1, **options)'
Traceback (most recent call last):
    1: from -e:1:in `<main>'
-e:1:in `foo': wrong number of arguments (given 2, expected 1) (ArgumentError)
```

I go for the exception. opt = {}; foo(**option) should consistently pass an empty hash instead of ignoring it. It is not intuitive, but it IS the current spec of keyword argument. This is a design flaw in the current spec. I believe that it must be fixed by complete separation between keyword arguments and positional arguments (#14183).

#21 - 03/14/2019 08:45 AM - mame (Yusuke Endoh)

Another presentation of the bug:

```
def foo; end
foo(*[], {}) #=> does not raise an exception in 2.6
```

#22 - 03/14/2019 09:05 AM - mame (Yusuke Endoh)

- Status changed from Assigned to Closed

Applied in changeset trunk|r67256.

---

compile.c: fix the corner case of rest and keyword arguments

See https://bugs.ruby-lang.org/issues/10856#note-20 . [Bug #10856]