Ruby trunk - Bug #8316

Can't pass hash to first positional argument; hash interpreted as keyword arguments

04/24/2013 07:41 AM - TylerRick (Tyler Rick)

Status:	Closed		
Priority:	Normal		
Assignee:	mame (Yusuke Endoh)		
Target version:			
ruby -v:	ruby 2.2.2p95 (2015-04-13 revision 50295) [x86_64-darwin13]	**Backport:**	

Description

I'm able to pass any other type of object to my first argument:

```
def foo(hash, opt: true)
puts "hash: #{hash}, opt: #{opt.inspect}"
end

foo 'a'               # => hash: a, opt: true
foo [{a:1}]             # => hash: [{:a=>1}], opt: true
foo [{a:1}], opt: false # => hash: [{:a=>1}], opt: false
```

But when I try to pass a hash, it raises an ArgumentError:

```
foo({a:1})  # Raises ArgumentError: unknown keyword: a
# Expected behavior: hash: {:a=>1}, opt: true
```

I tried to work around the "unknown keyword" error by using ** but ended up getting a "wrong number of arguments (0 for 1)" error instead.

```
def foo_with_extra(hash, **extra)
puts "hash: #{hash}, extra: #{extra.inspect}"
end

foo_with_extra 'a'               # hash: a, extra: {}
foo_with_extra [{a:1}]             # hash: [{:a=>1}], extra: {}
foo_with_extra [{a:1}], opt: false # hash: [{:a=>1}], extra: {:opt=>false}

foo_with_extra({a:1})  # Raises ArgumentError: wrong number of arguments (0 for 1)
# Expected behavior: hash: {:a=>1}, extra: {}
```

This behavior is surprising and I haven't seen it mentioned anywhere before. Is it really intentional?

Related issues:

Related to Ruby trunk - Feature #14183: "Real" keyword argument	**Open**	
Is duplicate of Backport200 - Backport #8040: Unexpect behavior when using ke...	**Closed**	**03/08/2013**

History

#1 - 04/24/2013 07:44 AM - TylerRick (Tyler Rick)

=begin
http://jp.rubyist.net/magazine/?Ruby200SpecialEn-kwarg#f01 said:

Be careful when passing hashes to methods with both variable length argument lists and keyword arguments.

but in this example, the argument list is ((*not*)) variable length.
=end

#2 - 04/24/2013 07:52 AM - phluid61 (Matthew Kerwin)

TylerRick (Tyler Rick) wrote:

> I'm able to pass any other type of object to my first argument:
>
> ```
> def foo(hash, opt: true)
> puts "hash: #{hash}, opt: #{opt.inspect}"
> end
>
> foo 'a'                # => hash: a, opt: true
> foo [{a:1}]            # => hash: [{:a=>1}], opt: true
> foo [{a:1}], opt: false # => hash: [{:a=>1}], opt: false
> ```
>
> But when I try to pass a hash, it raises an ArgumentError:
>
> ```
> foo({a:1})  # Raises ArgumentError: unknown keyword: a
> # Expected behavior: hash: {:a=>1}, opt: true
> ```

Additionally, calling foo({opt:1}) throws ArgumentError: wrong number of arguments (0 for 1)

This issue can be worked around by calling: foo({a:1},{}) , but it is unexpected.

#3 - 04/24/2013 02:58 PM - nobu (Nobuyoshi Nakada)

- Category set to core

- Status changed from Open to Assigned

- Assignee set to mame (Yusuke Endoh)

#4 - 05/30/2013 07:50 PM - mame (Yusuke Endoh)

- Status changed from Assigned to Closed

This issue was solved with changeset r40992.
Pablo, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- vm_insnhelper.c (vm_callee_setup_keyword_arg,
  vm_callee_setup_arg_complex): consider a hash argument for keyword
  only when the number of arguments is more than the expected
  mandatory parameters. [ruby-trunk - Bug #8040]

- test/ruby/test_keyword.rb: update a test for above.

#5 - 01/07/2016 02:34 AM - ozydingo (Andrew Schwartz)

- ruby -v changed from ruby 2.0.0p0 (2013-02-24 revision 39474) [x86_64-linux] to ruby 2.2.2p95 (2015-04-13 revision 50295) [x86_64-darwin13]

This is unfortunately still an issue with default values in positional arguments:

```
2.2.2 > def foo(hash={}, opt: true); p hash; p opt; end
=> :foo
2.2.2 > foo({a: 1})
ArgumentError: unknown keyword: a
```

Expected behavior is that foo can be called with a hash argument in the first position without needing to specify the optional keyword args.

Yusuke Endoh wrote:

> This issue was solved with changeset r40992.
> Pablo, thank you for reporting this issue.
> Your contribution to Ruby is greatly appreciated.
> May Ruby be with you.
>
> ---
>
> - vm_insnhelper.c (vm_callee_setup_keyword_arg,
>   vm_callee_setup_arg_complex): consider a hash argument for keyword
>   only when the number of arguments is more than the expected
>   mandatory parameters. [ruby-trunk - Bug #8040]
>
> - test/ruby/test_keyword.rb: update a test for above.

#6 - 01/30/2016 08:20 PM - avit (Andrew Vit)

- Backport deleted (1.9.3: UNKNOWN, 2.0.0: UNKNOWN)

Andrew Schwartz wrote:

> This is unfortunately still an issue with default values in positional arguments

See #11967 for the explanation.

#7 - 12/14/2017 07:23 AM - hsbt (Hiroshi SHIBATA)

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

#8 - 06/06/2018 01:49 AM - TylerRick (Tyler Rick)

ozydingo (Andrew Schwartz) wrote:

> This is unfortunately still an issue with default values in positional arguments:
>
> 2.2.2 > def foo(hash={}, opt: true); p hash; p opt; end
> => :foo
> 2.2.2 > foo({a: 1})
> ArgumentError: unknown keyword: a
>
> Expected behavior is that foo can be called with a hash argument in the first position without needing to specify the optional keyword args.

Ay! Just ran into this today.

Looks like this case is tracked in #12717 and #11967 but won't be fixed until Ruby 3 (#14183).

I wish there were a good workaround. Best I was able to come up with was:

```
def foo(hash={}, options = {opt: true})
  opt = options[:opt]
  [hash, opt]
end
```

```
 > foo({a: 1})
=> [{:a=>1}, true]
 > foo({a: 1}, opt: false)
=> [{:a=>1}, false]
```