Ruby master - Bug #7663
Unable to invoke a method with `**kwargs` if the receiver doesn’t define keyword arguments
01/07/2013 07:11 AM - wycats (Yehuda Katz)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td></td>
</tr>
<tr>
<td>Target version:</td>
<td>2.0.0dev</td>
</tr>
<tr>
<td>ruby -v:</td>
<td></td>
</tr>
</tbody>
</table>

Description
This works:

```ruby
def hello
  puts "hello"
end

hello([])
```

This does not:

```ruby
def hello
  puts "hello"
end

hello({})
```

I may be misunderstanding the idea behind the keyword arguments, but I would expect them to behave similarly to regular arguments when used with splat.

History

#1 - 01/07/2013 10:38 AM - matz (Yukihiro Matsumoto)
- Status changed from Open to Rejected

Unlike regular argument, empty keyword argument is not equal to empty regular list.

Matz.

#2 - 01/07/2013 12:46 PM - mame (Yusuke Endoh)

In the current design, keyword arguments are NOT omittable at a callee side.
In other words, you cannot pass keyword arguments to a method that does not support keyword arguments.

Other possible behavior is to ignore the passed arguments silently (as you expected), but I'm afraid if it is rather error-prone than useful. It tends to hide bugs.

If you really want to do so, you can use ** parameter explicitly.

```ruby
def hello(**dummy)
  puts "hello"
end
```

A bare ** ([#7662](#7662)) is indeed useful in this use case. I'm not so positive to this use case itself, though.