Ruby - Feature #10118

Double splat for non-symbol keys

08/07/2014 11:45 PM - sawa (Tsuyoshi Sawada)

Status:	Closed	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:		

Description

The double splat operator ** only seems to work with hashes whose keys are symbols. It will not work when a key is a string, for example. This is true for both ways; for construction:

```
def foo **; end
foo(:a => 3) #=> nil
foo("a" => 3) #=> ArgumentError: wrong number of arguments (1 for 0)
```

and destruction:

```
def bar *; end
bar(**{:a => 3}) #=> nil
bar(**{"a" => 3}) #=> TypeError: wrong argument type String (expected Symbol)
```

This is confusing. I propose that the double splat syntax should be extended so that it works even when the keys are not symbols.

Related issues:

Related to Ruby - Bug #10699: m(*a, **b) doesn't recognize integer options.	**Closed**

History

#1 - 08/08/2014 02:48 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

Yet another issue caused by the confusion caused by the distinction between symbols and strings and another reason to revisit #7792.

#2 - 08/08/2014 03:03 PM - matz (Yukihiro Matsumoto)

- Status changed from Open to Feedback

If you are JavaScript programmer with less Ruby experience, it might be confusing.
But should we change the language for those less experienced users, with introducing incompatibility?

Matz.

#3 - 08/08/2014 03:08 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

From my point of view the same issues that come from introducing incompatibility come from keeping this confusing behavior where both strings and symbols can be used as identifiers but one have to understand that they are different and not interchangeable.

If you introduce a few incompatibilities (I don't think there are that many that couldn't be easily fixed for good) then you force a few users to update their code to not expect symbols from behaving differently from strings. I'm still to hear a good use case where this wouldn't be feasible/desired.

On the other hand, if you don't introduce this small compatibility issue you end up with an increased number of confusion caused by them being different. I counted already a ton of issues in Redmine related to confusions due to symbols and strings not being compatible to each other.

There's no free solution that would fix all confusions and avoid introducing any backward incompatibilities.

#4 - 08/08/2014 04:41 PM - sawa (Tsuyoshi Sawada)

My intent was that, if the proposal is realized, we can easily create hashes that include other hashes like this without using merge:

```
h = {"a" => 1, some_object => 2}
{"b" => 3, **h, another_object => 4}
```

but if there were restriction that the keys must be symbols, then we cannot do this.

Rodrigo Rosenfeld Rosas, Using a string was just an example. It could be anything. #7792 is totally irrelevant, which was rejected. I am not saying that a string key should be assimilated with a corresponding symbol key. I completely agree with the opinion that #7792 should be rejected. Please discuss your problems in thread #7792 if you want to, and not here.

#5 - 08/09/2014 12:49 AM - nobu (Nobuyoshi Nakada)

- File 0001-vm.c-allow-to-splat-non-symbol-keys.patch added

- Category set to syntax

- Assignee set to matz (Yukihiro Matsumoto)

- Target version set to 2.2.0

It sounds reasonable.

#6 - 08/09/2014 04:10 AM - dsferreira (Daniel Ferreira)

Hi,

As far as I know the double splat was introduced in ruby 2.0.
What was the reason behind the design decision?
Does that decision still holds?

I believe the answer to this questions can give a direction to the problem in hands.
I must say I’m very surprised that such behaviour exists and I would be very interested in understanding why it exists as it stands right now.

The differentiation between strings and symbols in hashes causes issues.
We all know that hence the discussion around HashWithIndiferentAccess

Why are we accepting only hashes with symbols as arguments?

Are we aiming to completely stop using strings as hash keys in future releases?

Thanks,

Daniel

#7 - 08/09/2014 04:52 AM - matz (Yukihiro Matsumoto)

Double splat was introduced to pass a hash given from keyword arguments.
Keyword argument hash is fundamentally a hash with symbol keys.

Matz.

#8 - 01/06/2015 03:01 AM - josh.cheek (Josh Cheek)

Yukihiro Matsumoto wrote:

> Double splat was introduced to pass a hash given from keyword arguments.
> Keyword argument hash is fundamentally a hash with symbol keys.
>
> Matz.

I've always considered the final hash to be a hash of options. I can't think of a reason to split the options hash in two based on whether the keys are symbols.

Isn't the purpose that a method can extract whatever options it cares about and then pass the remaining options to the next method? In that case, having them in one hash makes the most sense, otherwise weird things happen, here we see 3..6=>:delete got turned into its own separate hash and added to the wrong variable.

```
def m1(*nums, opts)
  nums # => [1, 2, 3, 4, 5, {3..6=>:delete}]
  opts # => {:offset=>2}
end

# do something with multiplier and pass off to m1
def m2(*nums, multiplier:, **opts)
  m1(*nums, **opts)
end

# some options for m2, some for m1
options = {
  multiplier: 5,
  offset:     2,
  3..6 => :delete
}
m2(1,2,3,4,5, options)
```

To deal with this, you would have to explicitly check the other args for hashes accidentally getting appended to their list.

For example, this causes Open3 to break when the key is the file descriptor (such as Open3.popen3 'echo message >&3', 3 => $stdout). As the caller, there is no way for me to pass my arguments such that they are passed correctly to spawn since it happens inadvertently between methods down the callstack. Depending on whether my options are symbols or integers, they'll either wind up in popen_run's opts or cmd argument. So I must edit the source https://github.com/ruby/ruby/pull/808/files or find a different implementation, because I can't know what my users' open3 is going to do.

#9 - 01/06/2015 06:10 AM - nobu (Nobuyoshi Nakada)

*- Related to Bug #10699: m(*a, **b) doesn't recognize integer options. added*

#10 - 12/01/2016 02:49 PM - akostadinov (Aleksandar Kostadinov)

It would be beneficial if one could do:

```
my_method(**hash_with_string_keys)
```

This for example you can read a hash from MongoDB and pass it as parameters to your method without additional processing. For example to implement persistent queue of some operations.

#11 - 09/01/2019 12:49 PM - sawa (Tsuyoshi Sawada)

I think this feature has already been implemented for Ruby 2.7. If I am correct, please close this issue.

#12 - 09/01/2019 03:48 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Feedback to Closed

Files

0001-vm.c-allow-to-splat-non-symbol-keys.patch	2.06 KB	08/09/2014	nobu (Nobuyoshi Nakada)