Ruby master - Feature #9047

Alternate hash key syntax for symbols

10/24/2013 12:17 AM - jamonholmgren (Jamon Holmgren)

Status:	Closed	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:	2.6	

Description

=begin

In Ruby, if you can create a symbol with ((|:"symbolname"|)), it seems consistent to allow moving the colon to the right side in a hash and dropping the hash rocket (=>).

{
:str => "v",   # symbol
str: "v",   # symbol
:"str" => "v", # symbol
"str": "v",   # should also be a symbol
}

It would look like this:

h = {
"mykey": "value",
"otherkey": "othervalue",
regular_symbol: "value"
}

String and other non-symbol keys would retain the hash rocket syntax to avoid ambiguity.

{
"string" => "v",
MyObj.new => "v",
@my_var => "v"
}

Thoughts?

=end

Related issues:

Is duplicate of Ruby master - Feature #4276: Allow use of quotes in symbol sy...	**Closed**	**01/13/2011**

History

#1 - 10/24/2013 12:29 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

I'd prefer to reserve this syntax as a short hash syntax for string keyed hashes:

```
{
 'string': 'v' # equivalent to 'string' => 'v'
}
```

If your proposal is accepted, then it wouldn't be possible add the short syntax support for strings.

Do you have any real-world use case where a symbol would be useful but you need the :'xxx' constructor to generate it? I don't think this is a common thing to happen...

#2 - 10/24/2013 12:37 AM - jamonholmgren (Jamon Holmgren)

I would be okay with your idea, Rodrigo, although it's less consistent (if you look at my first code block in the description). It would result in symbols looking like this:

{
:'symbol': 'v'

}

Not the end of the world, though. Would you then also allow other literals to use the colon, not just strings / symbols?

{

}

I realize these are edge cases, but it's worth considering.

#3 - 10/24/2013 12:40 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

yes, it makes sense to me to accept anything as a key. The only problem is that we can't use names in variables with this syntax :(

```
key = 'a'
{key: 1} # will be {:key => 1}, not {'a' => 1}
```

Alternatively we could do:

```
{"#{key}": 1}
```

But it wouldn't be shorter in such case :P

#4 - 10/24/2013 12:54 AM - mohawkjohn (John Woods)

This may or may not be related, but we here at NMatrix (part of SciRuby) would love to be able to index ranges in NMatrix using a 1:3 notation. This can be accomplished with a hash, but only if it will allow numeric (rather than symbolic keys) before the colon. Currently we have to type:

```
n[1=>3, 2=>4]
```

or use the .../.. notation, but we would prefer

```
n[1:3,2:4]
```

#5 - 10/24/2013 02:03 AM - jamonholmgren (Jamon Holmgren)

I still think my original suggestion is more consistent and has fewer implications, but would like further input.

#6 - 10/24/2013 11:21 AM - matz (Yukihiro Matsumoto)

- Assignee set to matz (Yukihiro Matsumoto)

- Target version set to 2.6

mohawkjohn (John Woods) let us separate the issue. There may be a chance to introduce num:num literals for your purpose (just maybe).

jamonholmgren (Jamon Holmgren) having Symbol GC, it is realistic to have that kind of notation to cope well with JSON. But I am still not sure how much useful this is. Any use-case?

Matz.

#7 - 10/24/2013 11:56 AM - mohawkjohn (John Woods)

matz (Yukihiro Matsumoto) Very well, and thank you for the consideration. I opened a new issue on that topic: #9049.

#8 - 11/23/2013 03:53 PM - jamonholmgren (Jamon Holmgren)

matz (Yukihiro Matsumoto) -- sorry, I didn't receive an email notification, so I didn't realize you had responded.

This isn't MRI, I realize, but in RubyMotion this notation would come in handy.

https://github.com/clearsightstudio/ProMotion/issues/345#issuecomment-29115788

```
add UIButton.new, {
  "setTitle:forState:" => ['Register', UIControlStateNormal],
  "addTarget:action:forControlEvents:" => [self, 'register', UIControlEventTouchUpInside]
}

# becomes:

add UIButton.new, {
  "setTitle:forState:": ['Register', UIControlStateNormal],
  "addTarget:action:forControlEvents:": [self, 'register', UIControlEventTouchUpInside]
}
```

Another situation is when you want to map strings:

```
def map_string(source_string)
  {
    "jamon-holmgren": "Jamon A. Holmgren",
    "matzumoto-yukihiro": "Yukihiro Matsumoto"
  }[source_string]
end
```

Perhaps somewhat contrived, but you get the point. The first (RubyMotion) example I do deal with on a regular basis, being the creator of ProMotion.

#9 - 01/10/2014 12:58 AM - jamonholmgren (Jamon Holmgren)

I should also mention this allows for similar syntax between JavaScript, Python, and Ruby. In this case, all three languages could translate the same dictionary/hash in a more or less compatible way.

#10 - 03/14/2014 11:02 PM - jamonholmgren (Jamon Holmgren)

Looks like there is a patch already for this:

https://bugs.ruby-lang.org/issues/4276

#11 - 03/15/2014 02:21 AM - nobu (Nobuyoshi Nakada)

- Is duplicate of Feature #4276: Allow use of quotes in symbol syntactic sugar for hashes added

#12 - 09/20/2014 01:48 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

Applied in changeset r47649.

---

parse.y: quoted ID key

- parse.y (assoc): allow quoted ID as a key of a hash literal. [ruby-core:34453] [Feature #4276]