Ruby - Feature #11105

ES6-like hash literals

04/29/2015 12:32 AM - shugo (Shugo Maeda)

Status:	Rejected
Priority:	Normal
Assignee:	
Target version:	

Description

Why not support ECMAScript6-like hash literals?

For example,

```
{x, y}
```

is equivalent to:

```
{x: x, y: y}
```

For convenience, the prefix of global, instance, and class variables should be removed from the key name as follows:

```
a = 1
B = 2
$c = 3
@d = 4
@@e = 5
p({a, B, $c, @d, @@e, f: 6})
#=> {:a=>1, :B=>2, :c=>3, :d=>4, :e=>5, :f=>6}
```

Related issues:

Related to Ruby - Feature #13137: Hash Shorthand	**Rejected**	
Related to Ruby - Feature #11104: ES6-like hash literals	**Rejected**	**04/29/2015**
Related to Ruby - Feature #14579: Hash value omission	**Closed**	
Related to Ruby - Feature #16095: 2 Features: remove (simplify) 'new' keyword...	**Rejected**	
Has duplicate Ruby - Feature #15236: add support for hash shorthand	**Rejected**	

History

#1 - 04/29/2015 12:38 PM - hsbt (Hiroshi SHIBATA)

Hi Martin.

I fixed this issue. it is caused by latest version of redmine.
Please try to report again.

Thanks.

On Wed, Apr 29, 2015 at 2:02 PM, "Martin J. Dürst"
duerst@it.aoyama.ac.jp wrote:

> I tried to reply to Shugo on bugs.ruby-lang.org. However, I got the
> following validation message:
>
> "% Done is not included in the list"
>
> I do not know why I would have to indicate "% Done" on a feature that we
> only just started discussing. I also don't know where/how I would be
> indicating this (I could easily set the percentage to 0 if I knew where to
> do that).
>
> Regards, Martin.

--
SHIBATA Hiroshi shibata.hiroshi@gmail.com
http://www.hsbt.org/

#2 - 04/29/2015 01:06 PM - shugo (Shugo Maeda)

2015-04-29 10:19 GMT+09:00 Matthew Kerwin matthew@kerwin.net.au:

> Shugo Maeda wrote:
>
> > Why not support ECMAScript6-like hash literals?
>
> Does it make code easier to read, or just easier to write? Personally I find it a bit confusing/obfuscated.

The proposed syntax contributes to readability because it reduces redundancy and tells us that a key has the same name as the corresponding variable.

For example, it is obvious that only name has a different name from the corresponding variable in the following code:

```
h = {
  name: username,
  password,
  e_mail
}
```

#3 - 04/29/2015 05:38 PM - bruka (besnik ruka)

It does look easier to read but feels like it would be harder to write and debug.

What if you're typing in a hurry, or doing a lot of copy/paste and somehow you accidentally omit a key? Currently the interpreter would raise a syntax error immediately, but with this you would get cryptic logic bugs where the size of the hash is correct, but the key you want is missing.

Also this pretty much means you can't have an expression as the value i.e.

```
h = {
  name.upcase,
  email
}
```

Another issue might be that this basically ties your hash to the naming of your variables:

```
def sanitize_name(name, email)
  name = name.upcase
  return { name, email }
end
```

Which looks fine, but if later on i decide to go back and rename my local variable, I'd have to go find everywhere i've used that hash and make sure the key is changed properly. Not sure I want that kind of coupling.

#4 - 04/29/2015 09:32 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

I use that a lot with CoffeeScript and would love to be able to do the same with Ruby. +1

#5 - 04/30/2015 01:36 AM - mame (Yusuke Endoh)

-1.

{x, y} is a conventional notation of a mathematical set.
When I read it, I expect it to be equivalent to { x => true, y => true }.

--
Yusuke Endoh mame@ruby-lang.org

#6 - 04/30/2015 02:36 AM - matz (Yukihiro Matsumoto)

Question:

What if we have variables with same name and different prefixes? e.g.

```
a = 1
@a = 2
```

```
$a = 3
```

Matz.

#7 - 04/30/2015 02:48 AM - abracu (Alfredo Bravo Cuero)

Hola matz

Alfredo Bravo Cuero (@abracu)
Open Source culture advocate and Ruby on Rails Developer
| Skype: yo@alfredobravocuero.co | Cel +573192837240

Enviado desde mi iPad

#8 - 04/30/2015 02:50 AM - shugo (Shugo Maeda)

Yukihiro Matsumoto wrote:

> What if we have variables with same name and different prefixes? e.g.
>
> ```
> a = 1
> @a = 2
> $a = 3
> ```

The last one is used with warnings:

```
lexington:ruby$ cat x.rb
a = 1
@a = 2
$a = 3
p({a, @a, $a})
lexington:ruby$ ./ruby x.rb
x.rb:4: warning: duplicated key at line 4 ignored: :a
x.rb:4: warning: duplicated key at line 4 ignored: :a
{:a=>3}
```

#9 - 04/30/2015 02:55 AM - shugo (Shugo Maeda)

Yusuke Endoh wrote:

> {x, y} is a conventional notation of a mathematical set.
> When I read it, I expect it to be equivalent to { x => true, y => true }.

{x, y} looks like a set of variable bindings, so it's reasonable that each key is a variable name and each value is its value.

#10 - 04/30/2015 01:38 PM - agarie (Carlos Agarie)

As Yusuke Endoh said, I'd expect this notation to create a Set, not a Hash...

Carlos Agarie
+55 11 97320-3878 | @carlos_agarie

#11 - 05/05/2015 12:36 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

Yukihiro Matsumoto wrote:

> Question:
>
> What if we have variables with same name and different prefixes? e.g.
>
> ```
> a = 1
> @a = 2
> $a = 3
> ```

CoffeeScript would compile to this:

```
@a=1; a=2; {@a, a} => {a: this.a, a: a}
```

If the same happens in Ruby, this is what I get in IRB:

```
> {a: 1, a: 2}
```

```
(irb):2: warning: duplicated key at line 2 ignored: :a
=> {:a=>2}
```

Sounds good to me.

#12 - 05/08/2015 07:02 AM - nobu (Nobuyoshi Nakada)

Shugo Maeda wrote:

```
lexington:ruby$ cat x.rb
a = 1
@a = 2
$a = 3
p({a, @a, $a})
lexington:ruby$ ./ruby x.rb
x.rb:4: warning: duplicated key at line 4 ignored: :a
x.rb:4: warning: duplicated key at line 4 ignored: :a
{:a=>3}
```

Why they make the same symbol?
a, @a and $a are irrelevant, separate variables.
I'd expect {:a=>1, :@a=>2, :$a=>3} without any warnings.

#13 - 05/14/2015 09:15 AM - shugo (Shugo Maeda)

Nobuyoshi Nakada wrote:

> lexington:ruby$ ./ruby x.rb
> x.rb:4: warning: duplicated key at line 4 ignored: :a
> x.rb:4: warning: duplicated key at line 4 ignored: :a
> {:a=>3}

> Why they make the same symbol?
> a, @a and $a are irrelevant, separate variables.
> I'd expect {:a=>1, :@a=>2, :$a=>3} without any warnings.

Because I don't come up with any use case of such ugly key names.
I believe code like {a, @a} should not be used in real world applications.

#14 - 05/14/2015 09:30 AM - matz (Yukihiro Matsumoto)

- Status changed from Open to Rejected

I am not positive about this syntax mostly because it appears to be set syntax, or old style hash in 1.8.
Once ES6 syntax become more popular, there will be chance for this change in the future.

Matz.

#15 - 01/20/2017 12:49 AM - znz (Kazuhiro NISHIYAMA)

- Related to Feature #13137: Hash Shorthand added

#16 - 01/20/2017 12:50 AM - znz (Kazuhiro NISHIYAMA)

- Related to Feature #11104: ES6-like hash literals added

#17 - 06/19/2017 10:32 PM - tleish (Tony Fenleish)

+1

While it might be odd or new for some, using this in ES6 has been very nice. I am often wishing this was supported in Ruby. The cognitive load is so much nicer and less redundant.

I wish it could be reconsidered.

#18 - 08/24/2017 03:55 AM - knu (Akinori MUSHA)

This syntax is now widely known and popular in the JavaScript/ES world. It is frequently used in everyday code and we've grown used to it.

#19 - 08/24/2017 10:26 PM - kernigh (George Koehler)

This ES6 syntax for hash literals looks strange to me. I have never seen it before today, but I have not written JavaScript for a few years, and I am not using Ruby for web development.

Two things puzzle me:

1. What is the key in {@a}? I want :@a as the key, others want :a. I don't expect Ruby to change a into @a. For example, object.instance_variable_get :a raises NameError, but object.instance_variable_get :@a works.
2. Should {print} call the method print if there's no local variable? That's what {print: print} would do.

#20 - 08/25/2017 01:55 AM - nobu (Nobuyoshi Nakada)

It makes many conflicts with the current syntax.
I don't think it is easy to resolve.

#21 - 03/06/2018 01:50 PM - shugo (Shugo Maeda)

- Related to Feature #14579: Hash value omission added

#22 - 10/19/2018 02:04 PM - nobu (Nobuyoshi Nakada)

- Has duplicate Feature #15236: add support for hash shorthand added

#23 - 08/11/2019 12:16 AM - znz (Kazuhiro NISHIYAMA)

- Related to Feature #16095: 2 Features: remove (simplify) 'new' keyword and Property Shorthand added

Files

0001-support-ES6-like-hash-literals.patch	3.88 KB	04/29/2015	shugo (Shugo Maeda)