

Ruby - Feature #14579

Hash value omission

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

Status:	Closed	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:		
Description How about to allow value omission in Hash literals: <pre>x = 1 y = 2 h = {x:, y:} p h #=> {x=>1, :y=>2}</pre> And in keyword arguments: <pre>def login(username: ENV["USER"], password:) p(username:, password:) end login(password: "xxx") #=> {:username=>"shugo", :password=>"xxx"}</pre>		
Related issues: Related to Ruby - Feature #11105: ES6-like hash literals Rejected Related to Ruby - Feature #18124: Hash shorthands (matching constructors func... Closed Related to Ruby - Feature #14973: Proposal of percent literal to expand Hash Closed Has duplicate Ruby - Feature #17292: Hash Shorthand / Punning Closed		

Associated revisions

Revision 7899866849c1bac708e513e5d9e14467a0155a9e - 09/11/2021 10:09 AM - nobu (Nobuyoshi Nakada)

Another test for [Feature #14579]

The value of the dynamic key cannot be omitted for now.

Revision 7899866849c1bac708e513e5d9e14467a0155a9e - 09/11/2021 10:09 AM - nobu (Nobuyoshi Nakada)

Another test for [Feature #14579]

The value of the dynamic key cannot be omitted for now.

Revision 78998668 - 09/11/2021 10:09 AM - nobu (Nobuyoshi Nakada)

Another test for [Feature #14579]

The value of the dynamic key cannot be omitted for now.

Revision d05ef38865af8dd43fd583e46f46509fe29e93cb - 09/11/2021 10:20 AM - nobu (Nobuyoshi Nakada)

[DOC] NEWS for [Feature #14579] [ci skip]

Revision d05ef38865af8dd43fd583e46f46509fe29e93cb - 09/11/2021 10:20 AM - nobu (Nobuyoshi Nakada)

[DOC] NEWS for [Feature #14579] [ci skip]

Revision d05ef388 - 09/11/2021 10:20 AM - nobu (Nobuyoshi Nakada)

[DOC] NEWS for [Feature #14579] [ci skip]

Revision 297f9b8d4c4502aa2ba0eccf93dfce215a7b6dfe - 09/11/2021 11:23 AM - shugo (Shugo Maeda)

Add documentation and tests for keyword argument value omission

[Feature #14579]

Revision 297f9b8d4c4502aa2ba0eccf93dfce215a7b6dfe - 09/11/2021 11:23 AM - shugo (Shugo Maeda)

Add documentation and tests for keyword argument value omission

[Feature #14579]

Revision 297f9b8d - 09/11/2021 11:23 AM - shugo (Shugo Maeda)

Add documentation and tests for keyword argument value omission

[Feature #14579]

History

#1 - 03/06/2018 01:50 PM - shugo (Shugo Maeda)

- *Related to Feature #11105: ES6-like hash literals added*

#2 - 03/06/2018 01:51 PM - shugo (Shugo Maeda)

- *File hash_value_omission.diff added*

#3 - 03/08/2018 11:04 AM - Eregon (Benoit Daloze)

I find this syntax very confusing.

The two occurrences of password: above means two very different things (required kwarg and auto-hash creation).

For

```
x = 1
y = 2
h = {x:, y:}
```

it looks to me like the values are missing.

I'd prefer a syntax which is different than "key syntax without value", and refers to the variable name used for the value more clearly, like:

```
x = 1
y = 2
h = {x, y}
```

That would also work for the second case like so:

```
def login(username: ENV["USER"], password:)
  p({username, password})
end
```

#4 - 03/08/2018 11:09 AM - Eregon (Benoit Daloze)

Should this also work for non-Symbols keys like:

```
x = 1
y = 2
h = { "x" => , "y" => }
```

#5 - 03/08/2018 11:30 AM - phluid61 (Matthew Kerwin)

Eregon (Benoit Daloze) wrote:

I'd prefer a syntax which is different than "key syntax without value", and refers to the variable name used for the value more clearly, like:

```
x = 1
y = 2
h = {x, y}
```

Please no, this is too close to perl's weird handling of lists/hashes. To me it reads like you're trying to write:

```
h = {1=>2}
```

#6 - 03/09/2018 02:55 AM - shevegen (Robert A. Heiler)

I agree with Matthew.

I understand the suggestion trying to make the syntax even more succinct (less to type) but I think it's one step too much. Ruby already has a

quite condensed syntax.

I think this syntax here, asked by Benoit, is also problematic:

```
h = { "x" => , "y" => }
```

Has a slight "visual" problem, at the least to me. I would expect => to "point" to something on the right hand side, which the normal syntax in hashes, in ruby, requires (unless you use the foo: :bar syntax notation).

The:

```
h = {x:, y:}
```

to my brain it's indeed a bit confusing because I would normally expect something on the right side of "foo: ".

#7 - 03/09/2018 08:59 AM - shugo (Shugo Maeda)

Eregon (Benoit Daloze) wrote:

I'd prefer a syntax which is different than "key syntax without value", and refers to the variable name used for the value more clearly, like:

```
x = 1
y = 2
h = {x, y}
```

I proposed the above syntax in [#11105](#), but it was rejected, and this proposal is alternative.

#8 - 03/15/2018 06:34 AM - matz (Yukihiro Matsumoto)

I prefer this syntax to [#11105](#), but this introduces a Ruby-specific syntax different from ES6 syntax. Besides that, I don't like both anyway because they are not intuitive (for me).

Matz.

#9 - 03/15/2018 07:49 AM - shugo (Shugo Maeda)

- Status changed from Open to Rejected

matz (Yukihiro Matsumoto) wrote:

I prefer this syntax to [#11105](#), but this introduces a Ruby-specific syntax different from ES6 syntax. Besides that, I don't like both anyway because they are not intuitive (for me).

So I withdraw this proposal.

#10 - 10/29/2020 05:48 AM - mame (Yusuke Endoh)

- Has duplicate Feature #17292: Hash Shorthand / Punning added

#11 - 08/20/2021 06:18 PM - mame (Yusuke Endoh)

- Related to Feature #18124: Hash shorthands (matching constructors functionality in JS) added

#12 - 09/11/2021 09:20 AM - sorah (Sorah Fukumori)

- Status changed from Rejected to Open

- Assignee set to matz (Yukihiro Matsumoto)

#13 - 09/11/2021 09:28 AM - knu (Akinori MURASHI)

Here's a typical use case.

```
def get_user_profile(client)
  client.get_json("/current_user") => { id: }
  client.get_json("/profile", { id: }) => { nick:, bio: }

  return { id:, nick:, bio: }
end
```

#14 - 09/11/2021 09:52 AM - matz (Yukihiro Matsumoto)

After the RubyKaigi 2021 sessions, we have discussed this issue and I was finally persuaded. Our mindset has been updated (mostly due to mandatory keyword arguments). Accepted.

Matz.

#15 - 09/11/2021 09:59 AM - shugo (Shugo Maeda)

- Status changed from Open to Closed

Thank you.

Committed in c60dbcd1c55cd77a24c41d5e1a9555622be8b2b8.

#16 - 09/13/2021 02:04 AM - matz (Yukihiro Matsumoto)

I assumed the value should be a local variable. The merged patch calls the method when the local variable is not defined. I doubt this is sufficient behavior. Any opinion?

Matz.

#17 - 09/13/2021 02:23 AM - mame (Yusuke Endoh)

For the record: { "#{ str }": } is not allowed. Matz said that it is intentional.

#18 - 09/13/2021 03:23 AM - knu (Akinori MUSHASHI)

We should allow it to call a (private) method if no variable with the name defined. We use methods in RSpec or with attr_reader that look like variables, and programmers don't necessarily distinguish between methods from variables when writing a program. I believe this syntax should take methods into account.

#19 - 09/13/2021 03:53 AM - baweaver (Brandon Weaver)

knu (Akinori MUSHASHI) wrote in [#note-18](#):

We should allow it to call a (private) method if no variable with the name defined. We use methods in RSpec or with attr_reader that look like variables, and programmers don't necessarily distinguish between methods from variables when writing a program. I believe this syntax should take methods into account.

I would agree that (private) methods are very useful here, especially attr_* methods. There are a few cases I would wonder what they do:

- @var: - Would this work with instance/class/global/constant variables if they're valid symbols?
- a = 1; {a:, b: 3} - Does it support mixing omissions and regular values?
- p a:, b: 3 - Does it work with implied hashes / keywords? (I think yes).

I agree that { "#{ str }": } should not be allowed, as it presents potential for abuse and vulnerabilities.

I've [PR'd the second case on mixed values](#), but just considered the first with ivars and similar concepts. I'm not sure which way that should go.

#20 - 09/13/2021 04:03 AM - knu (Akinori MUSHASHI)

We also discussed further with Matz and concluded that quoted keys ({ "key": }) are not allowed with or without interpolation. This is simply because you don't need that when any local variable or constant can be written without quotation, and because it might make you feel it could possibly mean { "key": "key" } and that would be confusing.

#21 - 09/13/2021 04:10 AM - knu (Akinori MUSHASHI)

baweaver (Brandon Weaver) wrote in [#note-19](#):

- @var: - Would this work with instance/class/global/constant variables if they're valid symbols?

No, because we didn't change the symbol key syntax. { @var: @var } is not valid, so { @var: } isn't either. The same goes for \$var and @@var.

- a = 1; {a:, b: 3} - Does it support mixing omissions and regular values?

Yes.

- p a:, b: 3 - Does it work with implied hashes / keywords? (I think yes).

Yes, but beware when you omit the last value without the closing parenthesis. The interpreter will look further past the line end for a value.

#22 - 09/15/2021 06:13 AM - shugo (Shugo Maeda)

- Status changed from Closed to Assigned

matz (Yukihiro Matsumoto) wrote in [#note-16](#):

I assumed the value should be a local variable. The merged patch calls the method when the local variable is not defined.
I doubt this is sufficient behavior. Any opinion?

I believe a method should be called when a local variable is not defined.
Because it's convenient as knu stated, and because {x:} is a syntax sugar of {x: x} except that keywords are allowed.

#23 - 09/15/2021 06:20 AM - shugo (Shugo Maeda)

Note that constants are also allowed:

```
X = 1
p(X:) #=> { :X=>1 }
```

#24 - 09/15/2021 06:40 AM - shugo (Shugo Maeda)

shugo (Shugo Maeda) wrote in [#note-22](#):

except that keywords are allowed.

I meant that keywords are allowed as local variable or method names.
For example, {if:} is not a syntax error and {self:} doesn't access the receiver but a local variable or method self.

#25 - 09/15/2021 07:18 AM - duerst (Martin Dürst)

shugo (Shugo Maeda) wrote in [#note-24](#):

I meant that keywords are allowed as local variable or method names.
For example, {if:} is not a syntax error and {self:} doesn't access the receiver but a local variable or method self.

Ah, so {if:} means something close to {if: local_variable_get(:if)} and '{self:}' means something close to {self: local_variable_get(:self)} (and NOT {self: self}). Not sure we need this, but also not sure it hurts.

#26 - 09/15/2021 08:54 AM - shugo (Shugo Maeda)

duerst (Martin Dürst) wrote in [#note-25](#):

Ah, so {if:} means something close to {if: local_variable_get(:if)} and '{self:}' means something close to {self: local_variable_get(:self)} (and NOT {self: self}).

Yes.
Technically speaking, send(:if) is used instead of local_variable_get if the local variable is not defined.

Not sure we need this, but also not sure it hurts.

In the meeting just after RubyKaigi, someone pointed out that {if:}[:if] is faster than binding.local_variable_get(:if).

```
excelsior:ruby$ cat bm.rb
require "benchmark"
```

```
Benchmark.bmbm do |b|
  -> {if:} {
    b.report("binding.local_variable_get") do
      10000.times do
        binding.local_variable_get(:if)
      end
    end
    b.report("new hash syntax") do
      10000.times do
        {if:}[:if]
      end
    end
  }.call(if: 123)
end
excelsior:ruby$ ./ruby bm.rb
```

```

Rehearsal -----
binding.local_variable_get    0.005680    0.000211    0.005891 ( 0.005889)
new hash syntax              0.001817    0.000136    0.001953 ( 0.001965)
-----
                                total: 0.007844sec

```

```

                                user      system      total      real
binding.local_variable_get    0.003668    0.000094    0.003762 ( 0.003763)
new hash syntax              0.000829    0.000162    0.000991 ( 0.001042)

```

#27 - 09/15/2021 09:27 AM - duerst (Martin Dürst)

shugo (Shugo Maeda) wrote in [#note-26](#):

duerst (Martin Dürst) wrote in [#note-25](#):

Technically speaking, `send(:if)` is used instead of `local_variable_get` if the local variable is not defined.

Not sure we need this, but also not sure it hurts.

In the meeting just after RubyKaigi, someone pointed out that `{if:}[if]` is faster than `binding.local_variable_get(:if)`.

I don't think using `if` as the name of a local variable is a good idea, and I don't think the speed of bad ideas should concern us too much.

#28 - 09/15/2021 10:00 AM - zverok (Victor Shepelev)

[@duerst \(Martin Dürst\)](#)

I don't think using `if` as the name of a local variable is a good idea,

It is good (and widely used, BTW) name for a method parameter, in contexts like

```
validate :foo, if: :something?
```

I don't see how it is bad idea, while producing the clearest method call convention for "conditional" DSLs.

#29 - 09/15/2021 10:08 AM - shugo (Shugo Maeda)

duerst (Martin Dürst) wrote in [#note-27](#):

I don't think using `if` as the name of a local variable is a good idea, and I don't think the speed of bad ideas should concern us too much.

As zverok stated, a keyword such as `if` is used as a keyword argument (especially on Rails?).

#30 - 09/16/2021 01:41 AM - Dan0042 (Daniel DeLorme)

matz (Yukihiro Matsumoto) wrote in [#note-16](#):

I assumed the value should be a local variable.

I also assumed the same thing, but after getting over my initial surprise I found this way is quite nice, very ruby-ish. A bit like learning that `rescue => obj.attr` is valid.

#31 - 09/16/2021 06:10 AM - shugo (Shugo Maeda)

- Status changed from Assigned to Closed

Matz accepted the current behavior at [DevelopersMeeting20210916Japan](#)

#32 - 09/16/2021 09:22 AM - knu (Akinori MUSA)

...Which is that `{ symbol: }` verbosely means `{ symbol: binding.local_variable_defined?(:symbol) ? binding.local_variable_get(:symbol) : __send__(:symbol) }` with no exception, no matter if the symbol is `if`, `self`, `fork`, `return` or whatever.

#33 - 11/30/2021 05:00 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #14973: Proposal of percent literal to expand Hash added

#34 - 08/22/2023 10:35 PM - olivierlaca (Olivier Lacan)

Has it been considered to ever expand this feature to allow instance variables and global variables. Or was it strictly reserved for local variables intentionally, to avoid incompatible or dangerous behavior?

Thanks for adding this regardless, it's a wonderful feature.

Files

hash_value_omission.diff	619 Bytes	03/06/2018	shugo (Shugo Maeda)
--------------------------	-----------	------------	---------------------