Ruby master - Feature #12512
Import Hash#transform_values and its destructive version from ActiveSupport

06/21/2016 02:02 PM - mrkn (Kenta Murata)

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

Description
I think value transformation is a fundamental feature of Hash.

Related issues:
- Related to Ruby master - Feature #7793: New methods on Hash
- Related to Ruby master - Feature #6669: A method like Hash#map but returns hash
- Related to Ruby master - Feature #8951: Please add a hash-to-hash alternative...
- Related to Ruby master - Feature #9635: Map a hash directly to a hash
- Related to Ruby master - Feature #9970: Add `Hash#map_keys` and `Hash#map_val...
- Related to Ruby master - Feature #10208: Passing block to Enumerable#to_h

Associated revisions
Revision ea5184b9 - 08/09/2016 08:54 AM - mrkn (Kenta Murata)
hash.c: implement Hash#map_v and Hash#map_v!

- hash.c (rb_hash_map_v, rb_hash_map_v_bang): implement Hash#map_v and Hash#map_v! [Feature #12512] [ruby-core:76095]

- test/ruby/test_hash.rb: add tests for above change.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@55847 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 55847 - 08/09/2016 08:54 AM - mrkn (Kenta Murata)
hash.c: implement Hash#map_v and Hash#map_v!

- hash.c (rb_hash_map_v, rb_hash_map_v_bang): implement Hash#map_v and Hash#map_v! [Feature #12512] [ruby-core:76095]

- test/ruby/test_hash.rb: add tests for above change.

Revision 55847 - 08/09/2016 08:54 AM - mrkn (Kenta Murata)
hash.c: implement Hash#map_v and Hash#map_v!

- hash.c (rb_hash_map_v, rb_hash_map_v_bang): implement Hash#map_v and Hash#map_v! [Feature #12512] [ruby-core:76095]

- test/ruby/test_hash.rb: add tests for above change.

Revision 55847 - 08/09/2016 08:54 AM - mrkn (Kenta Murata)
hash.c: implement Hash#map_v and Hash#map_v!

- hash.c (rb_hash_map_v, rb_hash_map_v_bang): implement Hash#map_v and Hash#map_v! [Feature #12512] [ruby-core:76095]

- test/ruby/test_hash.rb: add tests for above change.

Revision 55847 - 08/09/2016 08:54 AM - mrkn (Kenta Murata)
hash.c: implement Hash#map_v and Hash#map_v!

- hash.c (rb_hash_map_v, rb_hash_map_v_bang): implement Hash#map_v and Hash#map_v! [Feature #12512] [ruby-core:76095]
- test/ruby/test_hash.rb: add tests for above change.

Revision eaa0a27f - 09/08/2016 02:33 AM - mrkn (Kenta Murata)

hash.c: map_v -> transform_values

- hash.c (rb_hash_transform_values, rb_hash_transform_values_bang): Rename map_v to transform_values.
  [Feature #12512] [ruby-core:76095]
- test/ruby/test_hash.rb: ditto.

Revision 56099 - 09/08/2016 02:33 AM - mrkn (Kenta Murata)

hash.c: map_v -> transform_values

- hash.c (rb_hash_transform_values, rb_hash_transform_values_bang): Rename map_v to transform_values.
  [Feature #12512] [ruby-core:76095]
- test/ruby/test_hash.rb: ditto.

Revision 56099 - 09/08/2016 02:33 AM - mrkn (Kenta Murata)

hash.c: map_v -> transform_values

- hash.c (rb_hash_transform_values, rb_hash_transform_values_bang): Rename map_v to transform_values.
  [Feature #12512] [ruby-core:76095]
- test/ruby/test_hash.rb: ditto.

Revision 56099 - 09/08/2016 02:33 AM - mrkn (Kenta Murata)

hash.c: map_v -> transform_values

- hash.c (rb_hash_transform_values, rb_hash_transform_values_bang): Rename map_v to transform_values.
  [Feature #12512] [ruby-core:76095]
- test/ruby/test_hash.rb: ditto.

Revision 56099 - 09/08/2016 02:33 AM - mrkn (Kenta Murata)

hash.c: map_v -> transform_values

- hash.c (rb_hash_transform_values, rb_hash_transform_values_bang): Rename map_v to transform_values.
  [Feature #12512] [ruby-core:76095]
- test/ruby/test_hash.rb: ditto.

Revision 56099 - 09/08/2016 02:33 AM - mrkn (Kenta Murata)

hash.c: map_v -> transform_values

- hash.c (rb_hash_transform_values, rb_hash_transform_values_bang): Rename map_v to transform_values.
  [Feature #12512] [ruby-core:76095]
- test/ruby/test_hash.rb: ditto.

History

#1 - 06/21/2016 02:02 PM - mrkn (Kenta Murata)
- Related to Feature #7793: New methods on Hash added
#2 - 06/21/2016 02:03 PM - mrkn (Kenta Murata)
- Description updated

#3 - 06/21/2016 02:04 PM - mrkn (Kenta Murata)
- Description updated

#4 - 06/22/2016 12:14 AM - shyouhei (Shyouhei Urabe)
- Related to Feature #8669: A method like Hash#map but returns hash added

#5 - 06/22/2016 12:14 AM - shyouhei (Shyouhei Urabe)
- Related to Feature #8951: Please add a hash-to-hash alternative of the map method to Hash added

#6 - 06/22/2016 12:15 AM - shyouhei (Shyouhei Urabe)
- Related to Feature #9635: Map a hash directly to a hash added

#7 - 06/22/2016 12:15 AM - shyouhei (Shyouhei Urabe)
- Related to Feature #9970: Add `Hash#map_keys` and `Hash#map_values` added

#8 - 06/22/2016 12:18 AM - shyouhei (Shyouhei Urabe)
CF: https://github.com/rails/rails/pull/15819
I believe no one is against the feature itself.

#9 - 07/20/2016 02:26 AM - shyouhei (Shyouhei Urabe)
We looked at this issue at yesterday's developer meeting and no one there was against this API, including the value passed to its block, but not the name. This name is not rejected, though. The word "transform" had no usage in core before so we were wary about introducing it.

#10 - 08/09/2016 01:15 AM - mrkn (Kenta Murata)
I don't persist with the name transform_value.
map_v or map_values are also acceptable for me.

#11 - 08/09/2016 07:14 AM - matz (Yukihiro Matsumoto)
map_v accepted.

Matz.

#12 - 08/09/2016 07:16 AM - mrkn (Kenta Murata)
- Assignee changed from matz (Yukihiro Matsumoto) to mrkn (Kenta Murata)
- Status changed from Open to Assigned

#13 - 08/09/2016 08:54 AM - mrkn (Kenta Murata)
- Status changed from Assigned to Closed

Applied in changeset r55847.

---

hash.c: implement Hash#map_v and Hash#map_v!

- hash.c (rb_hash_map_v, rb_hash_map_v_bang): implement Hash#map_v and Hash#map_v! [Feature #12512] [ruby-core:76095]
- test/ruby/test_hash.rb: add tests for above change.

#14 - 08/09/2016 07:05 PM - janfri (Jan Friedrich)
Yukihiro Matsumoto wrote:

    #map_v accepted.

At first I thought it was an "adapted form" of Hash#map like Enumerable#grep_v is an "inverted" Enumerable#grep. Maybe the ",v" could be
confusing for other people?

Why not Hash#map_values - we have Enumerable#each_with_index, Enumerable#each_with_object, Hash#keys, Hash#values, Hash#values_at and other "full forms" of method names?

#15 - 08/10/2016 11:34 AM - Eregon (Benoit Daloze)

Yukihiro Matsumoto wrote:

> #map_v accepted.

Matz.

Will all due respect,
I think the name #map_v is not very Ruby-like.
Why such a short name?

#map_values seems more consistent with existing methods (#values, #values_at, #each_value, #has_value?, #value?).
It also conveys the intention explicitly, while I am fairly certain many people will be confused by #map_v.
From the documentation and standard terminology: "A Hash is a dictionary-like collection of unique keys and their values.".
Most of the duplicated tickets also mentioned #map_values and I believe this is the name most people would expect.
The discussion in https://bugs.ruby-lang.org/issues/7793 also converges towards #map_values (even nobu agreed :)).

Matz, could you please reconsider the name?

#16 - 08/10/2016 11:53 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

Yes, I would say the same a few days ago about map_v being a bad name... I also prefer map_values. I just didn't say anything because I prefer a map_v over not having such a method at all and didn't want to delay this feature :P

#17 - 08/10/2016 01:31 PM - matz (Yukihiro Matsumoto)

Benoit,
The name and behavior haven't fixed yet. So we welcome discussion.

Your statement about "Ruby-like" is pretty interesting. Even though I have been designing Ruby for last 20+ years, I am not sure if I could define "Ruby-like". Could you please suggest a better "Ruby-like" name than "map_v"?

And Rodrigo, we have decided to introduce a method, so you don't have to worry about the delay.

Matz.

#18 - 08/10/2016 04:43 PM - marcandre (Marc-Andre Lafortune)

map_values is good, succinct and clear. 'v' is an unclear abbreviation, made worse by the existence of other methods with the long form like values_at.

I guess I'm just repeating what Jan and Rodrigo said...

#19 - 08/10/2016 08:36 PM - Eregon (Benoit Daloze)

Yukihiro Matsumoto wrote:

> Benoit,
>
The name and behavior haven't fixed yet. So we welcome discussion.
>
> Your statement about "Ruby-like" is pretty interesting. Even though I have been designing Ruby for last 20+ years, I am not sure if I could define "Ruby-like". Could you please suggest a better "Ruby-like" name than "map_v"?
>
> Matz.
>
> Dear Matz,
>
> Thank you for the quick reply!
> I think #map_values is intuitive and consistent with other methods as well as the general terminology used in Hash.
> It seems almost everyone agrees on the name for once :
>
> About defining "Ruby-like" for names, that's though indeed.
> I would think it's often concise yet clear and intuitive, generally avoiding abbreviations and using well-established names from e.g. Unix and functional programming if there are.
> They also have their own syntactic particularities with _, ?, ! and =.
Matz, could you close it if you finish the discussion?

Matz.

Yukihiro Matsumoto wrote:

From our discussion, we concerned map._values is too close to map.values, which has totally different semantics. Besides that, we are thinking about adding other methods like map_k and map_kv.

Matz.

From our discussion, we concerned map_values is too close to map.values, which has totally different semantics.

But isn't map.values non sensical?

Moreover, it usually will be map_values { |k,v| ... } (i.e. followed by a block), while we never see h.values { ... }.

I don't see how there could be confusion here.

Yukihiro Matsumoto wrote:

map_values, map_keys and map_pairs seems consistent.

In addition, we are considering adding a method to create a hash from an enumerable in a similar way to map_pairs or map_kv. From my point of view, adding map_kv to Enumerable is acceptable, but map_pairs looks weird. You may feel differently, and if so, what should be the name of a method we are going to add to Enumerable?

Matz.

Yukihiro Matsumoto wrote:

map_values, map_keys and map_pairs seems consistent.

In addition, we are considering adding a method to create a hash from an enumerable in a similar way to map_pairs or map_kv.

What would be the semantics?
For simple cases there is already Enumerable#to_h for an enumeration of pairs:

{1..4}.each_cons(2).map { |k,v| ... }.to_h

For more complex cases, would it be something like Enumerable#associate or #categorize (#4151) ?

Can we stick to the name of #map_v here? I don't think it's wise to expand the front line in this thread.

Right, sorry for the diversion.
I agree with Marc-Andre, I do not find map_values confusing. After all we want to take a Hash, change/map its values to something else and return the resulting Hash.

I guess matz meant hash.values.map {} but that's the reverse order of operations and naturally it ends up with a different result (take the Hash values as an Array and map it).

I do not see this as confusing. Also there is already flat_map, collect_concat, etc which of course exist because it's not "just" a composition of the two operations mentioned in the name.

Enumerable#map_pairs sounds right if it's about manipulating pairs (2-element arrays or key-value pairs).

#28 - 09/07/2016 06:13 AM - matz (Yukihiro Matsumoto)
Ok, I will introduce transform_values (not map_v nor map_values).

I wanted a Hash generation method in Enumerable (e.g. map_kv), and the proposed method name to be consistent with the name. But I found out that they are not really "map" functions, so they shouldn't be named map_* . So I gave up the idea of map_v and map_kv altogether. The original name transform_values describes the intention better.

Matz.

#29 - 09/08/2016 02:33 AM - mrkn (Kenta Murata)
- Status changed from Open to Closed

Applied in changeset r56099.

hash.c: map_v -> transform_values

- hash.c (rb_hash_transform_values, rb_hash_transform_values_bang):
  Rename map_v to transform_values.
  [Feature #12512] [ruby-core:76095]

- test/ruby/test_hash.rb: ditto.

#30 - 10/25/2016 01:36 AM - shyouhei (Shyouhei Urabe)
- Related to Feature #10208: Passing block to Enumerable#to_h added