Ruby - Feature #6669

A method like Hash#map but returns hash

06/30/2012 02:23 AM - yhara (Yutaka HARA)

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

Description

Given a hash h, h.map returns an array(alist), but sometimes I hope it returned a hash.

Example:

```
class Hash
  def apply(&block)
    self.inject({}) do |h, (k, v)|
      new_k, new_v = *block.call(k, v)
      h[new_k] = new_v
      h
    end
  end
end

score = {
  taro: [1,3,2],
  jiro: [3,5,8,4],
  saburo: [2,9]
}
max_score = score.apply{|k,v| [k, v.max]}
#=> {taro: 3, jiro: 8, saburo: 9}
p max_score[:taro]
#=> 3
```

I'm not thinking "apply" is a perfect name for this. Maybe "hash_map" is better
(we already have "flat_map").

Related issues:

Related to Ruby - Feature #4151: Enumerable#categorize	**Rejected**	
Related to Ruby - Feature #7292: Enumerable#to_h	**Closed**	**11/07/2012**
Related to Ruby - Feature #7793: New methods on Hash	**Closed**	
Related to Ruby - Feature #12512: Import Hash#transform_values and its destru...	**Closed**	

History

#1 - 06/30/2012 02:27 AM - yhara (Yutaka HARA)

- File 6669.pdf added

Adding presentation slide for the feature request meeting ([ruby-dev:45708])

#2 - 06/30/2012 02:34 AM - trans (Thomas Sawyer)

Hi, I just want to mention that Facets has this using name #mash (map hash) with alias #graph, which was the original name. So those are two names to consider. Thanks.

#3 - 07/02/2012 01:35 AM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

Received, thank you!

--

Yusuke Endoh mame@tsg.ne.jp

#4 - 07/21/2012 06:12 PM - matz (Yukihiro Matsumoto)

Since Hash#reject, Hash#select does return a hash, I think it's OK for Hash#collect to return a hash.

I believe Hash#map should return an array as before, just like find_all does.

Matz.

#5 - 07/21/2012 07:53 PM - duerst (Martin Dürst)

On 2012/07/21 18:12, matz (Yukihiro Matsumoto) wrote:

> Issue #6669 has been updated by matz (Yukihiro Matsumoto).
>
> Since Hash#reject, Hash#select does return a hash, I think it's OK for Hash#collect to return a hash.
>
> I believe Hash#map should return an array as before, just like find_all does.
>
> Matz.

Wouldn't it be really confusing that for Arrays, #map and #collect are
synonyms, but for Hash, they are different?

Regards, Martin.

#6 - 07/22/2012 06:50 AM - marcandre (Marc-Andre Lafortune)

Hi,

duerst (Martin Dürst) wrote:

> Wouldn't it be really confusing that for Arrays, #map and #collect are
> synonyms, but for Hash, they are different?

It could be confusing, and would also introduce incompatibilities. Also, if the proposal for associate/categorize is accepted, then this collect would be a duplication.

I believe that methods like associate/categorize acting on all Enumerable to produce Hashes would be far more useful (since they don't only act on hashes), without introducing incompatibilities.

#7 - 07/22/2012 11:05 PM - trans (Thomas Sawyer)

It would introduce an incompatibility, but probably not as much as you might think since #map has become the more commonly used method and it is used even less frequently on a Hash.

I think it makes good sense to have a known set of methods that are *closed*, which is to say they return the same class of object. As matz points out, #select and #reject are already closed. #collect could be made closed without too much trouble since we still have the more widely used #map.

That doesn't get rid of the need for a better Array to Hash conversion method though, which has been discussed in other threads.

#8 - 07/24/2012 10:47 PM - mame (Yusuke Endoh)

- Status changed from Assigned to Feedback

Yutaka Hara,

We discussed your slide at the developer meeting (7/21).

Matz was positive to the feature itself, but there was no
method name that matz liked. Please find another good name.

Here is a discussion summary:

- hash_map, map_hash
  Matz's most favorite, but not enough to accept.
- associate
  Not bad, but not enough to accept.
- mash
  Such a created word is not good.

- apply, convert, process, graph
  Bad. They suggest something different.

- morph (Yugui suggested)
  This might be a correct terminology (?), but it is difficult
  for those unfamiliar with category theory.

--
Yusuke Endoh mame@tsg.ne.jp

#9 - 07/25/2012 10:46 PM - merborne (kyo endo)

how about #hmap.

#10 - 09/19/2012 02:30 PM - ryenus (_ ryenus)

What about #remap

#select and #reject only change the number of k/v pairs and do not change the mapping.

Compared to that, here we want to have certain keys to map to different values, so #remap clearly infers such purpose to me.

Regarding the bang version, i.e., #remap! can be used to modify self instead of returning a new hash.

#11 - 09/19/2012 03:09 PM - david_macmahon (David MacMahon)

What about #map! (or are bang methods frowned upon these days)?

#12 - 09/19/2012 04:15 PM - david_macmahon (David MacMahon)

Please ignore my previous comment; I see that you want a new Hash rather than replacing the keys/values of the existing Hash. I guess you want a more efficient alternative to:

max_score = Hash[score.map {|k,v| [k, v.max]}]

Since you want to create a new Hash from an existing object, how about a Hash::map factory method:

```
class Hash
def self.map(other, &block)
other.inject({}) do |h, (k, v)|
new_k, new_v = *block.call(k, v)
h[new_k] = new_v
h
end
end
end
```

Note that "other" doesn't even need to be a Hash (e.g. it could be an Array of two element Arrays). In fact, this:

Hash.map([[key, value], ...]) {|*kv| kv}

would be equivalent to:

Hash[[[key, value], ...]]

Perhaps if Hash::map is not given a block, the behavior could be as if the "{|*kv| kv}" block had been given.

#13 - 09/19/2012 11:43 PM - trans (Thomas Sawyer)

- mash
  Such a created word is not good.

I think when no other choices suffice one is left with two options, either created word or long explanatory term, e.g. #mash or #map_hash, respectively.

OTOH, recently I have been looking at new API related to this that uses fluent notation via an Enumerator-like "hashifier", e.g.

enum.hashify.map

This approach allows for other methods to be defined to "hashify" in variant ways.

#14 - 10/25/2012 04:36 PM - yhara (Yutaka HARA)

- Target version changed from 2.0.0 to 2.6

#15 - 02/07/2013 07:44 AM - phluid61 (Matthew Kerwin)

I might also suggest the name map_pairs, derived from each_pair. I think it's not incorrect, as the return value from each_pair is the original Hash object, so the return value from map_pairs could also be a Hash object.

Incidentally this paves the way for future possibilities, such as map_keys or map_values (as per each_key and each_value)

#16 - 03/13/2013 11:24 AM - phluid61 (Matthew Kerwin)

phluid61 (Matthew Kerwin) wrote:

> I might also suggest the name map_pairs, derived from each_pair. I think it's not incorrect, as the return value from each_pair is the original Hash object, so the return value from map_pairs could also be a Hash object.
>
> Incidentally this paves the way for future possibilities, such as map_keys or map_values (as per each_key and each_value)

This is related to #7793. I just released a gem that implements Hash#map_pairs, #map_keys, and #map_values.
https://rubygems.org/gems/hashmap

#17 - 09/06/2014 11:15 AM - nobu (Nobuyoshi Nakada)

- Description updated

#18 - 06/22/2016 12:14 AM - shyouhei (Shyouhei Urabe)

- Related to Feature #12512: Import Hash#transform_values and its destructive version from ActiveSupport added

#19 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)

#20 - 10/06/2021 07:41 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Feedback to Closed

This was implemented by Hash#to_h taking a block starting in Ruby 2.6.

Files

6669.pdf	41.7 KB	06/30/2012	yhara (Yutaka HARA)