Ruby master - Feature #7793

New methods on Hash

02/07/2013 01:53 AM - dsisnero (Dominic Sisneros)

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

Description

It would be nice to have the following methods added to hash

```
h = { name: 'dominic', request: 'add the following methods', :why => 'convenience'}

h.map_v{|v| v.upcase}
#=> {:name=>"DOMINIC", :request=>"ADD THE FOLLOWING METHODS", :why=>"CONVENIENCE"}

h.map_k{|k| k.to_s}
#=> { "name"=> 'dominic', "request"=>"add the following methods', "why" => "convenience"}

h.map_kv{|k,v| [k.to_s, v.upcase]}
#=> { "name"=>"DOMINIC", "request"=>"ADD THE FOLLOWING METHODS", "why"=>"CONVENIENCE"}


class Hash

  def map_v
    reduce({}) do |result, array|
      k,v = array
      new_val = yield v
      result.merge( k => new_val)
    end
  end

  def map_k
    reduce({}) do |result, array|
      k,v = array
      new_k = yield k
      result.merge(new_k => v)
    end
  end

  def map_kv
    reduce({}) do |result, array|
      new_k,new_v = yield array
      result.merge(new_k => new_v)
    end
  end

end
```

Related issues:

Related to Ruby master - Feature #6669: A method like Hash#map but returns hash	**Feedback**	
Related to Ruby master - Feature #4151: Enumerable#categorize	**Rejected**	
Related to Ruby master - Feature #7292: Enumerable#to_h	**Closed**	**11/07/2012**
Related to Ruby master - Feature #10552: [PATCH] Add Enumerable#frequencies a...	**Open**	**11/27/2014**
Related to Ruby master - Feature #12512: Import Hash#transform_values and its...	**Closed**	
Has duplicate Ruby master - Feature #9970: Add `Hash#map_keys` and `Hash#map_...	**Open**	

History

#1 - 02/07/2013 02:00 AM - marcandre (Marc-Andre Lafortune)

- Status changed from Open to Closed

I am glad to see that more people like you take the time to propose ways to create hashes.

I completely agree that hash creation from Enumerable is lacking currently.

I will close this feature request because I am convinced it can't be accepted as is (the proposed names have no chance of being accepted) and because it is largely duplicated by the following:

https://bugs.ruby-lang.org/issues/6669
https://bugs.ruby-lang.org/issues/4151
https://bugs.ruby-lang.org/issues/7292

If you have the time, read on those and see if you can contribute.

Thanks

#2 - 02/07/2013 05:02 AM - dsisnero (Dominic Sisneros)

This should be re-opened. It is not for all enumerables but only for hash.

map_v and map_k are very useful

map_kv is similar to h.mash and others and could be eliminated by those other bugs but the other functions aren't and are specifically for hashes and thus this should be re-opened

#3 - 02/07/2013 06:51 AM - marcandre (Marc-Andre Lafortune)

- Category set to core

- Status changed from Closed to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

Fine, I'll reopen and assign this to Matz.

#4 - 02/07/2013 06:59 AM - charliesome (Charlie Somerville)

At the risk of bike shedding, I think map_k and map_v should be named map_keys and map_values. That can be for matz to decide though.

#5 - 02/07/2013 10:14 AM - nobu (Nobuyoshi Nakada)

Considering existing methods:

```
$ ruby -e 'p Hash.instance_methods(false).grep(/each_/)'
[:each_value, :each_key, :each_pair]
```

They should be map_key, may_value, and map_pair, respectively, I
think.

Anyway, why don't you make it a gem first?

#6 - 02/07/2013 10:18 AM - nobu (Nobuyoshi Nakada)

- Description updated

#7 - 02/13/2013 06:22 PM - yhara (Yutaka HARA)

- Target version set to 2.6

#8 - 03/13/2013 11:13 AM - phluid61 (Matthew Kerwin)

nobu (Nobuyoshi Nakada) wrote:

> Anyway, why don't you make it a gem first?

That's a good idea. Let's see what the uptake is, if any: https://rubygems.org/gems/hashmap

Note: I used #map_keys, #map_values and #map_pairs as my method names.

#9 - 06/16/2014 03:49 PM - Ajedi32 (Andrew M)

FYI, Rails has a method similar to the proposed map_k called transform_keys.

#10 - 06/30/2014 04:22 AM - nobu (Nobuyoshi Nakada)

- Has duplicate Feature #9970: Add `Hash#map_keys` and `Hash#map_values` added

#11 - 09/06/2014 11:14 AM - nobu (Nobuyoshi Nakada)

- Description updated

#12 - 09/06/2014 04:14 PM - trans (Thomas Sawyer)

An issue with the name is that "map" semantically means to create an Array, i.e.ahash.map{ |k,v| ... } produces an Array. So map_keys would make sense to mean ahash.map_keys{ |k| ... } and produce an Array too. Hash#map_pair would just a synonym for #map, just as #each_pair is just a synonym for #each.

Facets has long had Hash#rekey and Hash#revalue (and in-place forms Hash#rekey! and Hash#revalue!). These names are concise and do not suffer this semantic issue. Note Facets doesn't have a #remap method (though I suppose it could) because it has Enumerable#mash, and it's alias #graph, which can create a Hash from any Enumerable object.

#13 - 10/08/2014 04:17 AM - sawa (Tsuyoshi Sawada)

Just like there are map and map!, there should be both a non-desctructive and a destructive version for this method.

```
h = {a: "foo"}

h.non_destructive_one{|k, v| [k.to_s, v.upcase]} #=> {"a" => "FOO"}
h #=> {a: "foo"}

h.destructive_one!{|k, v| [k.to_s, v.upcase]} #=> {"a" => "FOO"}
h #=> {"a" => "FOO"}
```

I also have a (not that strong) opinion that the block for these methods should take a hash rather than an array. That should make more sense since the return value is a hash.

```
h.non_destructive_one{|k, v| {k.to_s => v.upcase}} #=> {"a" => "FOO"}
h.destructive_one!{|k, v| {k.to_s => v.upcase}} #=> {"a" => "FOO"}
```

#14 - 10/08/2014 07:11 AM - avit (Andrew Vit)

> the block for these methods should take a hash rather than an array.

Do you mean the input should be a single argument with a hash? I don't think that is very consistent for |k, v| expansion.

> That should make more sense since the return value is a hash.

Everything inside the block is a tuple; what type the input/output are transformed from/to happens outside the block. IMHO the array makes more sense than the hash inside the block.

#15 - 10/29/2014 01:56 PM - Ajedi32 (Andrew M)

Below is a summary of the different naming proposals so far in this thread, with links to the documentation for real-world implementations where available.

Option 1

The original proposal, uses the term map to express changing the keys or values on the hash, and keeps things terse by abbreviating the terms 'key' and 'value':

- Hash#map_k
- Hash#map_k!
- Hash#map_vs
- Hash#map_v!
- Hash#map_kv
- Hash#map_kv!

Option 2

A clearer, more verbose alternative to option 1. (Proposed by Charlie Somerville.)

- Hash#map_keys
- Hash#map_keys!

- Hash#map_values
- Hash#map_values!
- Hash#map_pairs
- Hash#map_pairs!

Option 3

Given the existing methods Hash#each_key, Hash#each_value, and Hash#each_pair, it might be better to use a sigular alternative to option 2. (Proposed by Nobuyoshi Nakada.)

- Hash#map_key
- Hash#map_key!
- Hash#map_value
- Hash#map_value!
- Hash#map_pair
- Hash#map_pair!

Option 4

Given the potential for the previous options to be confused with Hash#map, which returns an array, it might be best to use an entirely different naming convention. This one is based on Facets, a popular (485,329 downloads on Rubygems) library with the purpose of extending Ruby's core classes with useful methods. (Proposed by Thomas Sawyer.)

- Hash#rekey
- Hash#rekey!
- Hash#revalue
- Hash#revalue!
- Enumerable#graph (See #6669)
  - Aliased as Enumerable#mash
- Hash#graph!
  - Aliased as Hash#mash!

Option 5

Similar to option 4, but based on the naming convention used by Ruby on Rails.

- Hash#transform_keys
- Hash#transform_keys!
- Hash#transform_values
- Hash#transform_values!
- Hash#transform_pairs
- Hash#transform_pairs!

#16 - 11/02/2014 05:19 PM - trans (Thomas Sawyer)

I can't help but mention it, because it gave me a chuckle....

I like rekey and revalue from #4, because they make sense semantically, don't confuse the idea of map returning an array, and they are *concise*. Concision is always a big plus. However graph and mash don't really convey much in their names (mash is combination of "map" and "hash" btw), so I've always been rather ho-hum about those, but never could come up with a better, yet still concise, alternative.

Options #2 and #5 are nice for their consistency --the use of _keys, _values and _pairs-- But they lack for concision (especially #5) which sucks, and #2 has the map name issue as mentioned.

So I tried a combination of both ideas using re- as the prefix to the three suffixes and got:

- rekey
- revalue
- repair

At which point the giggles kicked in :-)

#17 - 11/02/2014 07:23 PM - Ajedi32 (Andrew M)

"repair"? Hehe, yeah that's kind of an unfortunate coincidence.

The thing I really like about Option 4's graph and mash is that they are methods on Enumerable, which means they can be used with any Enumerable object, not just hashes. As I mentioned, the creation of a method like that is being discussed in #6669. Right now, a similar effect can be achieved (for the non-destructive method anyway) by chaining map and to_h, so perhaps the full hash transform methods don't provide as big of a benefit over what we have now as rekey and revalue do.

If we do decide to base our names off of the assumption that the full hash transform methods will be on Enumerable, and not Hash, then perhaps something like this might work:

- Enumerable#associate
- Hash#associate!

#18 - 11/29/2014 03:04 AM - duerst (Martin Dürst)

- Related to Feature #10552: [PATCH] Add Enumerable#frequencies and Enumerable#relative_frequencies added

#19 - 05/13/2015 06:12 PM - rafaelfranca (Rafael França)

I'm biased here since we already implemented part of Option #5 on Ruby on Rails but I prefer its explicitness over concision. It is not clear to me what rekey and revalue does.

#20 - 06/21/2016 02:02 PM - mrkn (Kenta Murata)

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