In 2.4, new useful method Hash\#transform_values was added. I would like to propose also adding matching method Hash\#transform_keys.

\{
  a: 1, b: 2
\}.transform_keys {{|k| k.to_s}}
  \-> {"a"=>1, "b"=>2}

What needs to be considered is what to do in case of two keys mapping to the same new key

\{
  a: 1, b: 2
\}.transform_keys {{|_, same_key| # what should happen?}}

I think using Hash[] as model behaviour is a good idea.

Hash[{ a: 1, b: 2 }].map {{|key, value| [:s, value]}}
  \-> {[:s=>2]}

it's also how Hash\#transform_keys works in rails (afaict).

This is a follow up feature request to #9970, which seems to be stalled. If the behaviour can be agreed upon, I can try putting together a patch (if no one else wants to step up).

Associated revisions
Revision 14051117 - 07/14/2017 06:44 AM - mrkn (Kenta Murata)
hash.c: Add Hash\#transform_keys and Hash\#transform_keys!
  - hash.c (transform_keys_i, rb_hash_transform_keys): Add Hash\#transform_keys. [Feature #13583] [ruby-core:81290]
  - hash.c (rb_hash_transform_keys_bang): Add Hash\#transform_keys!. [Feature #13583] [ruby-core:81290]
  - test/ruby/test_hash.rb: Add tests for above changes.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@59328 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 59328 - 07/14/2017 06:44 AM - mrkn (Kenta Murata)
hash.c: Add Hash\#transform_keys and Hash\#transform_keys!
  - hash.c (transform_keys_i, rb_hash_transform_keys): Add Hash\#transform_keys. [Feature #13583] [ruby-core:81290]
  - hash.c (rb_hash_transform_keys_bang): Add Hash\#transform_keys!. [Feature #13583] [ruby-core:81290]
  - test/ruby/test_hash.rb: Add tests for above changes.

Revision 59328 - 07/14/2017 06:44 AM - mrkn (Kenta Murata)
hash.c: Add Hash\#transform_keys and Hash\#transform_keys!
  - hash.c (transform_keys_i, rb_hash_transform_keys): Add Hash\#transform_keys. [Feature #13583] [ruby-core:81290]
  - hash.c (rb_hash_transform_keys_bang): Add Hash\#transform_keys!. [Feature #13583] [ruby-core:81290]
  - test/ruby/test_hash.rb: Add tests for above changes.
• hash.c (rb_hash_transform_keys_bang): Add Hash#transform_keys!.
  [Feature #13583] [ruby-core:81290]

• test/ruby/test_hash.rb: Add tests for above changes.

Revision 59328 - 07/14/2017 06:44 AM - mrkn (Kenta Murata)
hash.c: Add Hash#transform_keys and Hash#transform_keys!

• hash.c (transform_keys_i, rb_hash_transform_keys): Add Hash#transform_keys.
  [Feature #13583] [ruby-core:81290]

• hash.c (rb_hash_transform_keys_bang): Add Hash#transform_keys!.
  [Feature #13583] [ruby-core:81290]

• test/ruby/test_hash.rb: Add tests for above changes.

Revision ae1c9f13 - 07/19/2017 01:59 PM - kazu
NEWS: Add Hash#transform_keys and Hash#transform_keys!
[Feature #13583] [ruby-core:81290] [ci skip]
git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@59369 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 59369 - 07/19/2017 01:59 PM - znz (Kazuhiro NISHIYAMA)
NEWS: Add Hash#transform_keys and Hash#transform_keys!
[Feature #13583] [ruby-core:81290] [ci skip]

Revision 59369 - 07/19/2017 01:59 PM - kazu
NEWS: Add Hash#transform_keys and Hash#transform_keys!
[Feature #13583] [ruby-core:81290] [ci skip]

Revision 59369 - 07/19/2017 01:59 PM - kazu
NEWS: Add Hash#transform_keys and Hash#transform_keys!
[Feature #13583] [ruby-core:81290] [ci skip]

History
#1 - 05/19/2017 06:52 PM - graywolf (Gray Wolf)
  - Description updated

#2 - 05/19/2017 06:53 PM - graywolf (Gray Wolf)
  - Description updated

#3 - 05/20/2017 01:49 AM - shyouhei (Shyouhei Urabe)
Thank you for issuing this. I see there is an obvious needs for this transformation (stringify_keys) so I'm +1 to the feature.

Let's have a concrete definition of this requested method:

• Its name is Hash#transform_keys.
• It returns a newly created Hash instance.
• It has zero arity.
• It yields,
  • with only one block parameter (which is a key of the original hash),
  • the evaluated value is the new key for the entry.
• When the new key conflicts, later entry silently discards former entry (see the description of this issue).

Is it okey? Am I missing something? Do people have any opinion?

#4 - 05/20/2017 08:56 AM - graywolf (Gray Wolf)
I don't think you missed anything, except I would just point out to also add Hash#transform_keys!. I don't know if it's worth mentioning or just kinda
I think the names are good, both `#transform_keys` and `#transform_values`.

Seem quite clear to me from the names.

On the linked older issue (~3 years), the names were different, `Hash#map_keys` and `Hash#map_values`. Matz said that the names may be confusing. Perhaps `#transform_keys` and `#transform_values` are better names. (I have not checked if the proposal is the very same; shyouhei provided a very specific definition here, including behaviour such as arity and yield-situations, which I think the other proposal did not have). Guess matz will have a look.

graywolf, could you perhaps show some example documentation for the two methods?

#6 - 07/14/2017 05:33 AM - matz (Yukihiro Matsumoto)
Looks good to me.

Matz.

#7 - 07/14/2017 06:44 AM - mrkn (Kenta Murata)
- Status changed from Open to Closed

Applied in changeset trunk|r59328.

hash.c: Add `Hash#transform_keys` and `Hash#transform_keys!`
- [Feature #13583] [ruby-core:81290]
- [Feature #13583] [ruby-core:81290]
- test/ruby/test_hash.rb: Add tests for above changes.

#8 - 12/11/2017 09:28 PM - marcandre (Marc-Andre Lafortune)
- Assignee set to matz (Yukihiro Matsumoto)
- Status changed from Closed to Open

I'm not sure I like the current behavior of `transform_keys!`. Two possibilities: `transform_keys!` is each_key { delete(old_key), set(new_key) } (as is currently) or replace(transform_keys) (I think I prefer this).

Matz, could you confirm what behavior you want?

Current:

```ruby
h = {1 => :hello, 2 => 'world'}
h.transform_keys(&:succ) # => {2 => :hello, 3 => 'world'}
h.transform_keys!(&:succ) # => {3 => :hello}
```

With using replace, we'd get the same results.

The current behavior allows partial updates though:

```ruby
h = {1 => :hello, 2 => :world}
h.transform_keys! { |k| k == 1 ? :one : break }  
h # => {2 => world, :one => :hello}
```

With the replace version, `h` would be unchanged (or else we'd have to write an ensure to do the partial update)

#9 - 12/11/2017 10:07 PM - marcandre (Marc-Andre Lafortune)
Nevermind, I just remembered that ActiveSupport also defines `transform_keys!`, so best match its behavior.