# Ruby - Feature #5662

## inject-accumulate, or Haskell's mapAccum*

11/23/2011 05:24 AM - EdvardM (Edvard Majakari)

| | |
|---|---|
| **Status:** | Rejected |
| **Priority:** | Normal |
| **Assignee:** | |
| **Target version:** | |

**Description**

with Ruby, we often use this idiom to build a hash out of something:

new_hash = enum.inject({}) { |h, thing| h[compute_key(thing) = compute_value(thing)]; h }

while that last h is very easy to add, it is also easy to forget and feels logically not very injectish thing to do. I'd propose this we call 'infuse' in our project:

module Enumerable

# like inject, but returns accumulator instead. Instead of writing

# [1, 2].inject({}) {|h, i| h[i] = 2*i; h }

# just say

# [1, 2].infuse({}) {|h, i| h[i] = 2*i } # -> {1 => 2, 2 => 4}

def infuse(init, &block)
inject(init) { |acc, i| block.call(acc, i); acc }
end
end

Eg. [1, 2].infuse({}) { |a, i| a[i] = 2*i } # => {1 => 2, 2 => 4}

Instead of infuse, maybe inject_accum or inject_acc would be more rubyish method name.

**Related issues:**

| | |
|---|---|
| Related to Ruby - Feature #4151: Enumerable#categorize | **Rejected** |

### History

#### #1 - 11/23/2011 05:41 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

+1

#### #2 - 11/23/2011 07:39 AM - Eregon (Benoit Daloze)

You can already do this by using Enumerable#each_with_object or Enumerator#with_object:

```
[1, 2].each_with_object({}) { |i,h| h[i] = 2*i } # => {1=>2, 2=>4}
```

#### #3 - 11/23/2011 08:58 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

Interesting, I never noticed/used this method before. My only concern is about the naming "each_with_object" when you actually want to inject/accumulate. The code intention is not clear enough when you write each_with_object. Maybe a better alias could be included.

#### #4 - 11/23/2011 04:17 PM - EdvardM (Edvard Majakari)

I also noticed mapAccum* is quite different.

I have to agree with Rodrigo. (each_)with_object seems to really do the thing, but the name is a bit funny one. Then again, that could be just simply aliased in the code for accumulating.

#### #5 - 11/23/2011 04:51 PM - neleai (Ondrej Bilka)

Why not just use
Hash[[1,2].map{|a| [a,2*a]}]

#### #6 - 11/23/2011 07:31 PM - Eregon (Benoit Daloze)

Rodrigo Rosenfeld Rosas wrote:

> Interesting, I never noticed/used this method before. My only concern is about the naming "each_with_object" when you actually want to inject/accumulate. The code intention is not clear enough when you write each_with_object. Maybe a better alias could be included.

I think accumulate implies an accumulator, which you don't have in this case. A Hash does not accumulate values like a growing Integer for example, it rather "register" the key/value entries. The alias of inject, reduce, is actually clear to the intention, you should not use inject with an Array for example (instead of map).

each_with_object is just avoiding the explicit variable definition and returns it:

h = {}
[1, 2].each { |i| h[i] = 2*i }
h

I believe the code I showed is somewhat common in 1.9 and is clear to people knowing about it.

In this particular case, you could probably also use Hash.new:

Hash.new { |h,k| h[k] = k*2 }

#### #7 - 11/23/2011 08:29 PM - Anonymous

Benoit Daloze wrote :

> h = {}
> [1, 2].each { |i| h[i] = 2*i }
> h
>
> I believe the code I showed is somewhat common in 1.9 and is clear to people knowing about it.

I would write
Hash.new.tap do |h|
...
end

Heavier, but the intention is clearer, and without an extra variable (outside of the block).

_md

#### #8 - 11/23/2011 10:18 PM - EdvardM (Edvard Majakari)

Ok.. I'll give real example to show what is typical use case for us:

hash = MyDatabaseObject.get_all.infuse({}) { |h, r| h[normalize_db_key(r.id, r.name)] = r }

after that, code can quickly access any record by id and name saying

obj = hash[normalize_db_key(myid, myname)]

Then again, I'm quite happy with this "each_with_object".

#### #9 - 11/24/2011 12:47 AM - marcandre (Marc-Andre Lafortune)

Hi,

Edvard Majakari wrote:

> Ok.. I'll give real example to show what is typical use case for us:
>
> hash = MyDatabaseObject.get_all.infuse({}) { |h, r| h[normalize_db_key(r.id, r.name)] = r }

As pointed out, you currently have the choice of:

```
get_all.each_with_object({}) { |r, h| h[normalize_db_key(r.id, r.name)] = r }
Hash[ get_all.map { |r| [normalize_db_key(r.id, r.name), r] } ]
```

ActiveSupport also gives you:
get_all.index_by { |r| normalize_db_key(r.id, r.name) }

There is a proposition for Enumerable#associate/categorize in [ruby-core:33683] which would give you:
get_all.associate { |r| [normalize_db_key(r.id, r.name), r] }

I also feel your infuse proposal is much too close to inject/each_with_object. Moreover, if you need it mostly to create hashes, it might be best to look into a good way to create hashes (like the proposal for associate/categorize).

#### #10 - 11/26/2011 06:53 AM - ujihisa (Tatsuhiro Ujihisa)

> new_hash = enum.inject({}) { |h, thing| h[compute_key(thing)] = compute_value(thing); h }
>
> while that last h is very easy to add, it is also easy to forget and feels logically not very injectish thing to do. I'd propose this we call 'infuse' in our project:

It's just because you used []=. Use merge instead.

```
new_hash = enum.inject({}) {|h, thing| h.merge compute_key(thing) => compute_value(thing) }
```

I don't think we need Enumerable#infuse only for []=.

#### #11 - 03/28/2012 12:41 AM - mame (Yusuke Endoh)

*- Status changed from Open to Rejected*

I think the answer to this original proposal is "use each_with_object".
That's all.  Closing.

Please open another ticket for an alias of the method if needed.

--
Yusuke Endoh mame@tsg.ne.jp