Array#zip_with

Inspired by Haskell's zipWith function, I hacked on together for Ruby:

```
[1, 2, 3].zip_with([6, 5, 4], :+) #=> [7, 7, 7]
[1, 2, 3].zip_with([6, 5, 4]) { |a, b| 3*a+2*b } #=> [15, 16, 17]
```

So far I only have a Ruby version of it:

https://gist.github.com/731702b90757e21cadcb

My questions:

1. Would this method be considered a worthwhile addition to Array?
2. I've never hacked on the C side of Ruby (read some parts of the source though) and my C is quite rusty. I'd like to change that, would somebody be willing to help me turn this into a proper patch?
I'm answering here since redmine won't answer to my requests (even got a 500).

I think this new method would be redundant with Array#zip. But I agree doing `c.zip(d).map { |a,b| a+b }` is a bit long.

So I propose to change the return value of Array#zip (and Enumerable#zip) with a block from nil to the result #map would give (so an Array of all yielded elements).

An unconditional nil is anyway not useful here, the only concern I see might be the cost of creating this Array. But I think many cases with block simulate #map and it would avoid creating the intermediate Array in `a.zip(b).map {}`.

This would not address the case with a Symbol, which could be easily detected as last argument as it is not Enumerable. But blocks simplified by Symbol have rarely been accepted and I think only changing the return value would already improve #zip a lot.

Here is a gist with a Ruby implementation of the modifications for Array#zip: [https://gist.github.com/903388](https://gist.github.com/903388)

I'm wishing to do a C implementation if this feels right for others, but I'd like to have opinions first.

What do you think?

---

#3 - 04/09/2011 06:23 PM - mrkn (Kenta Murata)

=begin

Hi,

On 2011年4月5日火曜日 at 19:50, Benoit Daloze wrote:

Here is a gist with a Ruby implementation of the modifications for Array#zip: [https://gist.github.com/903388](https://gist.github.com/903388)

I'm wishing to do a C implementation if this feels right for others, but I'd like to have opinions first.

I implemented the features in C, and wrote tests for them.

Please see the following diffs:

[https://github.com/mrkn/ruby/commit/9c7ead0e385b6a17dafa5bc8b4389e1ba2e3040](https://github.com/mrkn/ruby/commit/9c7ead0e385b6a17dafa5bc8b4389e1ba2e3040)

I will commit this if matz will approve.

... Kenta Murata
Sent with Sparrow

=end

#4 - 04/09/2011 10:23 PM - matz (Yukihiro Matsumoto)

=begin

Hi,

In message "Re: [ruby-core:35673] Re: [Ruby 1.9 - Feature #4539][Assigned] Array#zip_with" on Sat, 9 Apr 2011 17:29:28 +0900, Kenta Murata muraken@gmail.com writes:

I implemented the features in C, and wrote tests for them.

Please see the following diffs:

[https://github.com/mrkn/ruby/commit/9c7ead0e385b6a17dafa5bc8b4389e1ba2e3040](https://github.com/mrkn/ruby/commit/9c7ead0e385b6a17dafa5bc8b4389e1ba2e3040)

I will commit this if matz will approve.

I am not sure whether adding new zip_with or adding zip with symbol at last would be better. Any opinion?

matz.

=end

#5 - 04/09/2011 11:23 PM - Eregon (Benoit Daloze)

=begin

Hi,

On 9 April 2011 10:29, Kenta Murata muraken@gmail.com wrote:

=end
Hi,

I implemented the features in C, and wrote tests for them. Please see the following diffs:
https://github.com/mrkn/ruby/commit/9c7ead0e385b6a17dafa5bc8b4389e1ba12e3040

Thank you, nice diff.

It would be nice to add the two examples from Michael Kohl:

```
[1,2,3].zip([6,5,4], :+) #=> [7, 7, 7]
[1,2,3].zip([6,5,4]) { |a,b| 3*a+2*b } #=> [15, 16, 17]
```

And of course do the same for Enumerable.zip.

Just a quick notice: in the tests, the arguments are reversed (it is assert_equal(expected, actual, msg = nil)). It is always hard to remember with test/unit, that is a reason why I prefer the spec syntax in general. It is definitely just a detail, but it could mislead someone reading a failing test.

=begin

#6 - 04/10/2011 12:23 AM - Eregon (Benoit Daloze)

=begin

Hello,

On 9 April 2011 15:13, Yukihiro Matsumoto matz@ruby-lang.org wrote:

| Hi,
| |
| I am not sure whether adding new zip with or adding zip with symbol at
| last would be better. Â Any opinion?
| |
| Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â Â matz.

Although I already presented my opinion, I'd like to clarify it.

I think adding a new method for just one special form is not worth it. Also, a method name ending with a preposition is rather unusual in core/stdlib.

I feel honored to see you here,
B.D.

=end

#7 - 04/10/2011 12:23 PM - sorah (Sorah Fukumori)

=begin

HI,

On Sat, Apr 9, 2011 at 10:13 PM, Yukihiro Matsumoto matz@ruby-lang.org wrote:

I am not sure whether adding new zip_with or adding zip with symbol at
last would be better. Â Any opinion?

I vote to adding zip with symbol at last.

Because zip_with method name is so long and hard to understand behavior.

--
Shota Fukumori a.k.a. @sora_h - http://codnote.net/

=end

#8 - 04/10/2011 12:29 PM - mame (Yusuke Endoh)

=begin

Hi,

2011/4/9 Yukihiro Matsumoto matz@ruby-lang.org:

I am not sure whether adding new zip_with or adding zip with symbol at
last would be better. Â Any opinion?

I'm neutral for adding zip with symbol at last, but I
object to letting zip with block return a new array.

2011/4/5 Benoit Daloze eregonptp@gmail.com:

       An unconditional nil is anyway not useful here, the only concern I see might be the cost of creating this Array.

It is actually a problem. I have used Array#zip with block for iteration many times.

       big_ary1.zip(big_ary2) do |x, y|
       p [x, y]
       end

The change requires some people (including me) to rewrite such a code as follows:

       big_ary1.size.times do |i|
       x, y =
       #9 - 04/10/2011 10:23 PM - aprescott (Adam Prescott)

       =begin
On Sat, Apr 9, 2011 at 2:13 PM, Yukihiro Matsumoto matz@ruby-lang.org wrote:

       I am not sure whether adding new zip_with or adding zip with symbol at last would be better. Any opinion?

       matz.

       An issue I have with zip taking a symbol is that ary1.zip(ary2, :+) doesn't actually return a zipped array (of arrays). In that respect, the result and the name "zip" don't align, with the additional argument.

       Is there such a need for this in core that we need a shortcut around ary1.zip(ary2).map { |a, b| a + b }?

       =end

#10 - 04/11/2011 07:23 AM - mrkn (Kenta Murata)

=begin
On 2011年4月10日日曜日 at 11:56, Marc-Andre Lafortune wrote:

       On Sat, Apr 9, 2011 at 4:29 AM, Kenta Murata muraken@gmail.com wrote:

       I implemented the features in C, and wrote tests for them.
       Please see the following diffs:
https://github.com/mrkn/ruby/commit/9c7ead0a385b6a17dafa5bc8b4389e1ba9f2e3040

       Looks good, but is there a reason to use rb_funcall for the reduce?
       Typically MRI calls directly the C implementation wherever possible, and I would suggest doing that here too.

       Tee patch was updated:
https://github.com/mrkn/ruby/commit/fe7b7bb11e7d1173a31a7765472ac10eac9316

This changes add a new public function of libruby, rb_enum_inject_with_symbol. So we should discuss whether the function is acceptable as a public function.

--
Kenta Murata
Sent with Sparrow
=end

#11 - 04/24/2011 01:01 PM - jvoorhis (Jeremy Voorhis)

=begin
I think it's worth having Enumerable.zip_with as a new public method. zip_with is the generalization of zip (you can define zip = zipWith (,) in
Haskell). Because zip_with can be implemented directly via inject, it's possible to provide an implementation with a lower complexity than zip composed with map. With respect to the comment about a method ending in a preposition, the Haskell standard provides a good precedent, and a worthy addition to the collection of languages that influenced Ruby.

---

#12 - 07/18/2011 11:58 PM - trans (Thomas Sawyer)
@Endoh Why would you have to rewrite? You can still iterate, just don't use the return result.

#13 - 07/19/2011 12:00 AM - trans (Thomas Sawyer)
@matz (Yukihiro Matsumoto) is symbol really needed?

class Array

    def zip(a, &b1)
        if b
            r = []
            b2 = lambda{|x| r << b1.call(*x) }
            super(a, &b2)
            r
        else
            super(a)
        end
    end

end

[1,2,3].zip([5,6,7], &:+) #=> [6,8,10]

#14 - 07/19/2011 12:06 AM - trans (Thomas Sawyer)
@adam (Adam M) Perhaps you are right. Perhaps the real issue is why #map can't take optional "zipping" arguments?

#15 - 07/20/2011 12:59 PM - mame (Yusuke Endoh)
Hello,

2011/7/18 Thomas Sawyer transfire@gmail.com:
    @Endoh Â Why would you have to rewrite? You can still iterate, just don't use the return result.

I have used Array#zip in hot-path code to avoid unused array generation. So I don't want it to generate a unused and big array. But this is just my personal opinion, not a decision. If matz says ok, it is ok.

BTW, ko1 is now studying optimization to avoid unnecessary object generation by using escape analysis. If it is achieved, my concern will be pointless.

--
Yusuke Endoh mame@tsg.ne.jp

#16 - 07/21/2011 08:53 PM - aprescott (Adam Prescott)
On Mon, Jul 18, 2011 at 4:06 PM, Thomas Sawyer transfire@gmail.com wrote:
    @adam (Adam M) Perhaps you are right. Perhaps the real issue is why #map can't take optional "zipping" arguments?

I believe there is a suggestion from ruby-talk to call it map_with, instead. I agree with leaning towards #map instead of #zip, whatever happens.

#17 - 10/27/2012 11:53 PM - yhara (Yutaka HARA)
- Description updated
- Target version set to 2.6

#18 - 12/25/2017 06:14 PM - naruse (Yui NARUSE)
- Target version deleted (2.6)
#19 - 02/20/2018 08:54 AM - nobu (Nobuyoshi Nakada)
- Description updated

#20 - 10/22/2019 10:48 PM - duerst (Martin Dürst)
- Related to Feature #16261: Enumerable#each_splat and Enumerator#splat added

#21 - 10/23/2019 07:37 PM - shevegen (Robert A. Heiler)
Martin added this to the next developer meeting. I have not yet commented on this issue so I may briefly do so.

matz asked back then between zip_with, or zip with symbol. I think zip_with may be better than zip with symbol from a use-point of view.

As for zip_with versus map_with as shown by aprescott - I think zip_with may be better than putting an additional map_* name. I guess you could reason for an alias either way, but perhaps it would be better to keep it simple and start only with zip_with, see whether this may be used at all, before considering map_* changes (as a name; keep in mind that we have other use cases already with .map, such as .map.with_index(2) and such. This is also another reason why I think zip_* would be better than a map_* change here. But you could also reason either way if people don't read docs, and want to use a .map_* variant instead. :P Either way, I think it would be better to see for the potential use cases for .zip_with first).

It may also be worthwhile to ask mame for his opinion again too, years later, to see how/if any opinions changed/adapted/stayed the same or not in regards to the feature. :) (Actually, if it is a new method, then any object allocation situation may not be very important, since it would not effect more general use cases, e.g. current use of .zip() and such; but I do not know the C internals so I have no real clue).

#22 - 11/28/2019 08:23 AM - matz (Yukihiro Matsumoto)

- The name zip_with is too confusing with zip. We need a new name.
- The behavior can be described by the combination of zip and map.
- I am not sure how much we need this behavior (yet).

Matz.