Ruby master - Feature #8970

Array.zip and Array.product

10/01/2013 03:57 AM - sawa (Tsuyoshi Sawada)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

=begin
Most of the time when I use Array#zip or Array#product, I feel cumbursome that I have to take out the first array and pass it as a receiver. For example, if I have

```
a = [[:a, :b, :c], [:d, :e, :f], [:g, :h, :i]]
```

I have to do something like this:

```
a.first.zip(*a.drop(1)){...}
a.first.product(*a.drop(1)){...}
```

Sometimes, the receiver (i.e., the first array) has significance, but most other times, that breaks asymmetry, making the code look ugly.

I would be happy if we had Array.zip and Array.product in addition so that we can do it like this:

```
Array.zip(*a){...}
Array.product(*a){...}
```

=end

Related issues:

Has duplicate Ruby master - Feature #6499: Array::zip	**Rejected**	**05/26/2012**
Is duplicate of Ruby master - Feature #7444: Array#product_set	**Open**	

History

#1 - 10/01/2013 09:53 AM - akr (Akira Tanaka)

2013/10/1 sawa (Tsuyoshi Sawada) sawadatsuyoshi@gmail.com:

> Feature #8970: Array.zip and Array.product
> https://bugs.ruby-lang.org/issues/8970
>
> Most of the time when I use Array#zip or Array#product, I feel cumbursome that I have to take out the first array and pass it as a receiver. For example, if I have
>
> ```
> a = [[:a, :b, :c], [:d, :e, :f], [:g, :h, :i]]
> ```
>
> I have to do something like this:
>
> ```
> a.first.zip(*a.drop(1)){...}
> a.first.product(*a.drop(1)){...}
> ```
>
> Sometimes, the receiver (i.e., the first array) has significance, but most other times, that breaks asymmetry, making the code look ugly.
>
> I would be happy if we had Array.zip and Array.product in addition so that we can do it like this:
>
> ```
> Array.zip(*a){...}
> Array.product(*a){...}
> ```

How different with Array#transpose ?

% ruby -e '
a = [:a, :b, :c], [:d, :e, :f], [:g, :h, :i
p a.first.zip(*a.drop(1))
p a.transpose

'
:a, :d, :g], [:b, :e, :h], [:c, :f, :i
:a, :d, :g], [:b, :e, :h], [:c, :f, :i
--
Tanaka Akira

#2 - 10/01/2013 02:03 PM - sawa (Tsuyoshi Sawada)

akr, The difference between Array#transpose and Array.zip is just the same as with Array#transpose and Array#zip. That is, when any non-first array is shorter than the first, it is complemented with nil.

#3 - 03/01/2014 02:40 PM - sowieso (So Wieso)

+1
This would make code more readable by not breaking the symmetry.
Also would be nice if the block version wouldn't return nil, but instead the concatenation of all return values, like map does. (not that an explicit map would hurt, but this is a waste of a return value)

> Array.zip([1,2,3],[4,5,6]){|left,right| left + right} => [5,7,9]

#4 - 03/01/2014 06:40 PM - marcandre (Marc-Andre Lafortune)

This is a duplicate of https://bugs.ruby-lang.org/issues/6499 and https://bugs.ruby-lang.org/issues/7444.

#5 - 03/01/2014 06:41 PM - marcandre (Marc-Andre Lafortune)

- Has duplicate Feature #6499: Array::zip added

#6 - 03/01/2014 06:41 PM - marcandre (Marc-Andre Lafortune)

- Is duplicate of Feature #7444: Array#product_set added

#7 - 08/29/2019 07:44 AM - osyo (manga osyo)

If you use https://bugs.ruby-lang.org/issues/15955, you can write as follows.

```
class UnboundMethod
  # apply or other name
  def apply(receiver, *args)
    bind(receiver).call(*args)
  end
end

arrays = [["a", "b"], ["c"], ["d", "e"]]

p Array.instance_method(:product).apply(*arrays)
# => [["a", "c", "d"], ["a", "c", "e"], ["b", "c", "d"], ["b", "c", "e"]]

p Array.instance_method(:zip).apply(*arrays)
# => [["a", "c", "d"], ["b", nil, "e"]]
```

Need a syntax to call instance_method like .: ?

#8 - 08/30/2019 06:09 AM - sawa (Tsuyoshi Sawada)

osyo (manga osyo) wrote:

> If you use https://bugs.ruby-lang.org/issues/15955, you can write as follows.
>
> ```
> class UnboundMethod
>   # apply or other name
>   def apply(receiver, *args)
>     bind(receiver).call(*args)
>   end
> end
>
> arrays = [["a", "b"], ["c"], ["d", "e"]]
>
> p Array.instance_method(:product).apply(*arrays)
> # => [["a", "c", "d"], ["a", "c", "e"], ["b", "c", "d"], ["b", "c", "e"]]
>
> p Array.instance_method(:zip).apply(*arrays)
> # => [["a", "c", "d"], ["b", nil, "e"]]
> ```

> Need a syntax to call instance_method like .: ?

Thank you for the suggestion, but that looks a little too long. It can also be written as:

```
:product.to_proc.(*arrays)
# => [["a", "c", "d"], ["a", "c", "e"], ["b", "c", "d"], ["b", "c", "e"]]

:zip.to_proc.(*arrays)
# => [["a", "c", "d"], ["b", nil, "e"]]
```

But the proposal here seems to be favored more at least by mame-san (#16102).