Ruby master - Feature #14585

Array#each_pair

03/07/2018 09:37 PM - iamvery (Jay Hayes)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

Abstract

I propose we add the method #each_pair to Array. It would effectively be a name for the common case each_cons(2).

Background

A few times now, I have wanted to do something pairwise on an array of values. One example where this has come up is to display a list of values of consecutive ranges:

```
arr = [1,2,3,4]
arr.each_cons(2) do |(a,b)|
  puts "#{a}-#{b}"
end
# 1-2
# 2-3
# 3-4
```

Proposal

I see the value of Array#each_pair in being able to more clearly express a common use case where you wish do something with each overlapping pair of values in an array. It also mirrors Hash's similarly named interface well.

Implementation

The implementation could be as simple as aliasing each_cons(2):

```
class Array
  def each_pair
    each_cons(2)
  end
end
```

Evaluation

One drawback I see is that it may not be clear that Array#each_pair groups overlapping pairs vs. chunking elements, e.g. [1, 2, 3, 4] => 1, 2], [2, 3], [3, 4 vs. 1, 2], [3, 4. This could be addressed with an argument, but it may also be a feature killer, e.g. [1,2,3].each_pair(overlapping: true).

Discussion

[empty]

Summary

Overall I'd say that having an interface like this, named well, might make it easier to figure out how to access overlapping pairs of array elements. I've ending up needing to do this a few times now and both times I struggled to remember each_cons(2) is the way to do it as the name wasn't very intention revealing to me. (my brain kept saying "chunk").

This is my first feature suggestion. I'm looking forward to your feedback :). Thanks for all you do! <3

History

#1 - 03/07/2018 11:32 PM - phluid61 (Matthew Kerwin)

-1

If I saw a method Array#each_pair I'd expect it to behave like:

```
class Array
  def each_pair
    each_with_index do |element, index|
      yield index, element
    end
  end
end
```

Further, if you were to propose a method on Array that lets you iterate pair-wise I'd expect it to be like each_slice, not each_cons.

#2 - 03/08/2018 01:29 AM - sikachu (Prem Sichanugrist)

> it may not be clear that Array#each_pair groups overlapping pairs vs. chunking elements

Yeah, I think that's one reason I'm against an ambiguous method like this. I feel like each_cons(2) is already good for the job and more fitting for this. I also feel like the data structure of Array makes it so that each item actually stands on its own, unlike Hash where you know that there's always a key-value pair so grabbing "each pair" make sense.

#3 - 03/08/2018 04:33 AM - nobu (Nobuyoshi Nakada)

I though same behavior as phluid61 wrote.

#4 - 03/08/2018 04:44 AM - sawa (Tsuyoshi Sawada)

I think you have the wrong data structure in the first place. I think you either should have an array of arrays of length two, or a hash.

#5 - 03/08/2018 09:02 AM - Hanmac (Hans Mackowiak)

i don't like it because i think each_pair should react like each_slice(2)

#6 - 03/10/2018 09:05 PM - iamvery (Jay Hayes)

Thank you all for the feedback! I agree, and I'm glad your responses matched my intuition. Do you have any suggestions for how one might better remember each_cons? Ruby has a history of introducing aliases for the purpose of clarity. Can you think of such an alias for this method? For some reason my brain always defaults to "chunk", but perhaps that term is already too overloaded on Enumerable. Maybe it's just a matter of really committing it to memory, but I've certainly found it difficult to track down a few times now. Thanks again for taking the time to respond :)

#7 - 03/10/2018 10:16 PM - phluid61 (Matthew Kerwin)

iamvery (Jay Hayes) wrote:

> Thank you all for the feedback! I agree, and I'm glad your responses matched my intuition. Do you have any suggestions for how one might better remember each_cons? Ruby has a history of introducing aliases for the purpose of clarity. Can you think of such an alias for this method? For some reason my brain always defaults to "chunk", but perhaps that term is already too overloaded on Enumerable. Maybe it's just a matter of really committing it to memory, but I've certainly found it difficult to track down a few times now. Thanks again for taking the time to respond :)

It's not really chunking, either, which means grouping elements into fewer larger chunks (like each_slice turning an array of 10 items into an array of 5 pairs)

What each_cons does is expose **cons**ecutive sub-sequences.

#8 - 03/11/2018 11:38 PM - iamvery (Jay Hayes)

Great explanation, Matthew. Thank you! Sounds like I just need to practice this some more to really get the terminology into my vocabulary :)