Ruby master - Feature #15024

Support block in Array#join

08/24/2018 05:11 PM - graywolf (Gray Wolf)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

I think it could be handy to have block support in Array#join.

For example

```
> puts %w{a b c d}.join { |_, _, i| i % 2 == 1 ? "\n" : ', ' }
a, b
c, d
```

not sure what arguments exactly the block should take, atm I'm thinking of

```
> %w{a b c d}.join { |before, after, i| puts "#{before}:#{after}:#{i}" }
a:b:0
b:c:1
c:d:2
```

Would appreciate some feedback before I try putting together patch for this.

Related issues:

Related to Ruby master - Feature #14022: String#surround	**Rejected**

History

#1 - 08/24/2018 05:45 PM - jeremyevans0 (Jeremy Evans)

Your examples are both possible to implement using existing Array methods:

```
puts %w{a b c d}.each_slice(2).map{|a| a.join(", ")}.join("\n")
a, b
c, d

%w{a b c d}.each_cons(2).with_index.each{|a, i| puts (a << i).join(":")}
a:b:0
b:c:1
c:d:2
```

In the first case, your example has the block return the join separator. In the second example you don't appear to be returning the separator, you appear to be using the block for the side effect of iterating over the equivalent of each_cons(2).with_index (with before and after as separate block arguments).

I don't think adding block support to Array#join improves either example, and I think it makes the code more difficult to understand.

#2 - 08/24/2018 07:54 PM - graywolf (Gray Wolf)

Doesn't #each_slice create temporary array for each pair? Doesn't seem very efficient. But assuming that does not matter, can I use something similar to produce %{a b c d}.join { |_,_,i| i.to_s } == "a0b1c2d", is there oneliner for this too?

The second example was just illustrating what would be passed into the block.

I still think having possibility of block returning the separate is nice thing.

#3 - 08/24/2018 08:55 PM - jeremyevans0 (Jeremy Evans)

For %{a b c d}.join { |_,_,i| i.to_s } == "a0b1c2d", you could do:

```
*a, l = %w{a b c d}; a.each.with_object('').with_index{|(v, str), i| str << v << i.to_s} << l
```

That definitely isn't as nice looking. I can see the benefit of having the block return the separator, but I'm not sure how common such a need is. It's fairly straightforward to build the string manually in the cases you would need a separate separator per pair of items.

Could you share a practical example that would benefit from Array#join block support?

#4 - 08/25/2018 03:23 AM - nobu (Nobuyoshi Nakada)

Array#join concatenates array elements recursively.
What do you expect as the index between different level elements?

#5 - 08/25/2018 05:05 AM - mame (Yusuke Endoh)

I'd like somewhat to agree with the motivation. Indeed, I sometimes feel I want to insert separators between each pair of elements. However, I cannot remember the concrete situation, and how often I have encountered the situation. The motivation examples in this ticket look too artificial, so not convincing to me. Could you show us a more real-world use case?

Personally, I don't think it is a good idea to extend Array#join. It is specialized to generation of a string, but I'm unsure if it is enough. And, the block parameters |before, after, index| look too ad-hoc to me. Passing an index is a role of each_with_index. It is not a responsibility of join, I think.

#6 - 08/29/2018 09:15 AM - duerst (Martin Dürst)

- Related to Feature #14022: String#surround added