Ruby trunk - Feature #14022

String#surround

10/18/2017 04:38 AM - sawa (Tsuyoshi Sawada)

Status:	Rejected	
Priority:	Normal	
Assignee:		
Target version:		

Description

After joining the elements of an array into a string using Array#join, I frequently need to put substrings before and after the string. In such case, I would have to use either of the following:

```
[1, 2, 3].join(", ").prepend("<").concat(">") # => "<1, 2, 3>"
"<#{[1, 2, 3].join(", ")}>"                   # => "<1, 2, 3>"
"<" + [1, 2, 3].join(", ") + ">"              # => "<1, 2, 3>"
```

but none of them is concise enough. I wish there were String#surround that works like this:

```
[1, 2, 3].join(", ").surround("<", ">") # => "<1, 2, 3>"
```

Related issues:

Related to Ruby trunk - Feature #15024: Support block in Array#join	**Open**

History

#1 - 10/18/2017 04:43 AM - sawa (Tsuyoshi Sawada)

I would like both destructive and non-destructive versions of the method.

#2 - 10/18/2017 06:07 AM - mame (Yusuke Endoh)

IMO "<#{ foo }>" is more concise than foo.surround("<", ">").

#3 - 10/18/2017 06:20 AM - sawa (Tsuyoshi Sawada)

mame (Yusuke Endoh) In addition to conciseness, I often need to do this kind of string formatting after having done a long method chaining on an array. In that case, having to do string format from the beginning is not convenient.

```
"<#{some_array.some_very_long_method_chain.join(", ")}>"
```

It would be easier to read if String#surround were introduced.

```
some_array.some_very_long_method_chain.join(", ").surround("<", ">")
```

Also, in these use cases, the join(", ") operation and surrounding by "<" and ">" are a single logical operation. It makes more sense to do a chaining of join(...).surround(...) than to use a combination of join and string interpolation of "<" and ">".

#4 - 10/18/2017 09:10 AM - zverok (Victor Shepelev)

+1 for that (and exactly for the method chains).
Always define String#surround in my internal projects.

#5 - 10/18/2017 12:41 PM - Hanmac (Hans Mackowiak)

+1

i thought i have seen something like that before, but i don't remember where
ah now i remember, it was for JQuery#wrap http://api.jquery.com/wrap/

i think such a surround method might be used for xml stuff and other similar ones

#6 - 10/18/2017 07:04 PM - Eregon (Benoit Daloze)

+1, I often do "<" + long_chain + ">" because "<#{long_chain}>" tends to be harder to read, and wished there was such a method.

Making it part of #join might be slightly more efficient, but it would make the signature more complex, like [1, 2, 3].join(", ", left: "<", right: ">").

#7 - 10/19/2017 05:31 AM - shevegen (Robert A. Heiler)

> After joining the elements of an array into a string using Array#join,
> I frequently need to put substrings before and after the string.

I do not need to do this often, but I have had a need to do this, largely
due to file names on the *nix commandline that have ' ' characters (space),
so I pad them via "" like:

```
foo bar.mp3
```

to become:

```
"foo bar.mp3"
```

In particular when I then do system() invocation, e. g. to play via
mplayer/mpv.

So I can definitely see from which point Tsuyoshi Sawada is coming.

I also think that the name .surround() for String objects is concise
and may make sense, so I am also in +1 support.

So while I am not entirely sure whether this is extremely common, I
think it may be common enough to make this useful. I also agree on
the explanation given by Benoit Daloze, makes a lot of sense what he
wrote to ruby hackers I think. :)

#8 - 10/20/2017 12:56 AM - avit (Andrew Vit)

An alternate (short but cryptic) way:

```
str = "one\ntwo"
str.gsub(/^.*/m, '<\0>')
```

- gsub! can do it destructively
- using /m can control if it wraps each line, or all

(A similar usage for wrapping characters in a string is shown in the String#gsub documentation)

Out of curiosity, can someone explain why the ^ is needed in my regex?

Update: I just realized I could use sub instead, for some reason it doesn't need the ^ anchor.

I'm not against the idea of this method, just pointing out that there is already a way to do it. Also, should there be an equivalent "unquote" method to perform (essentially) str[1..-2]?

#9 - 10/20/2017 09:50 AM - knu (Akinori MUSHA)

I thought yield_self was about solving problems like this:

[1, 2, 3].join(", ").yield_self { |s| "<#{s}>" }

A nice-to-have in addition would be a shorter name, a special syntax, or a default block parameter (it, _, or whatever).

#10 - 10/20/2017 10:47 PM - Eregon (Benoit Daloze)

knu (Akinori MUSHA) wrote:

> I thought yield_self was about solving problems like this:
>
> [1, 2, 3].join(", ").yield_self { |s| "<#{s}>" }
>
> A nice-to-have in addition would be a shorter name, a special syntax, or a default block parameter (it, _, or whatever).

Interesting idea.
It is very long though.
It also is not as expressive as .surround("<", ">"), which makes the intent easier to read in my opinion.

#11 - 10/22/2017 10:49 PM - shevegen (Robert A. Heiler)

I guess it all ends up to how matz feels about .surround() :)

#12 - 10/23/2017 08:52 AM - duerst (Martin Dürst)

Two comments/ideas:

1. If the starting string and the ending string in surround are the same, it should be enough to give them only once:

   ```
   "Hello World!".surround("'") #=> "'Hello World!'"
   ```

2. As the examples above mention join a lot, it may also be possible to add two additional arguments to join:

   ```
   [1, 2, 3, 4].join(", ", "<", ">") #=> "<1, 2, 3, 4>"
   ```

   I would definitely use something like this, e.g. in

   ```
   array_of_lines.join("\n", "", "\n") #=> lines concatenated with newlines, ending with newline
   ```

#13 - 12/12/2017 02:12 PM - matz (Yukihiro Matsumoto)

I see ary.join.surround("<",">") to be no better than "<#{ary.join}>" or "<"+ary.join+">".
If the wrapped expression is long, you can format("<%s>", long_expression). I am not sure why you are so eager to chain method calls here.

Note: I am not rejecting the proposal (yet).

Matz.

#14 - 12/12/2017 02:28 PM - zverok (Victor Shepelev)

matz (Yukihiro Matsumoto)

Basically, in **my** practice (I can't speak for everyone of course) chaining is almost always a better way to construct value than operators, or interpolation, or something. Mostly because it follows "natural" flow of data, and therefore makes code more maintainable.

```
# Not that much difference
ary.join(',').surround('<', '>')
"<#{ary.join(',')}>"

# More difference:
File.read('some/source/path.txt')
 .split("\n")
 .map(&:strip)
 .grep_v(/^; /)
 .join(" ; ")
 .surround('(', ')')

"(#{File.read('some/source/path.txt')
 .split("\n")
 .map(&:strip)
 .grep_v(/^; /)
 .join(" ; ")})"
# of course, any sane developer rewrites the latter a
result = File.read('some/source/path.txt')
 .split("\n")
 .map(&:strip)
 .grep_v(/^; /)
 .join(" ; ")
"(#{result})"
```

But, as for **me** I always become frustrated when I need a new var because my "chain of thought" is broken by absence of methods. So, if we want optimize for happiness...

Well, that was the reason I fought for yield_self (still hate the name!), so in 2.5.0 you can do:

```
File.read('some/source/path.txt')
 .split("\n")
 .map(&:strip)
 .grep_v(/^; /)
 .join(" ; ")
 .yield_self { |res| "(#{res})" }
```

But for this really frequent case surround() still feels more elegant.

#15 - 12/28/2017 10:45 PM - shevegen (Robert A. Heiler)

> But for this really frequent case surround() still feels more elegant.

Agreed. It is not so frequent for my case, to be honest; but I like
the use case that sawa described since that is similar to ones I
experienced too, in regards to filenames (you know, file names which
may have empty spaces or ' characters and similar, but no " character).

"(#{variable})" works just fine or even '"'+filename+'"' :D but
string.surround('"') may feel more elegant (or perhaps .pad() but
I guess the name .pad() may be semi-reserved or refer to whitespace ...
.surround() seems less problematic)

The wiki lists that it was discussed or mentioned in a developer meeting
in late November 2017:

https://bugs.ruby-lang.org/projects/ruby/wiki/DevelopersMeeting20171129Japan

Not sure if anything has been decided - some meetings seem to have LOTS
of issues, I wonder if the japanese devs can discuss all of these in
less than 4 hours. :)

Perhaps it could be brought up again in 2018 at the next developer
meeting, if time allows?

#16 - 03/15/2018 08:14 AM - sorah (Sorah Fukumori)

- Status changed from Open to Feedback

It appears like yield_self or %s formatting can satisfy the use cases noted here.

Changing this ticket to Feedback for now. sawa (Tsuyoshi Sawada), could you update your opinion by taking a look into this discussion?

#17 - 03/16/2018 12:14 PM - sawa (Tsuyoshi Sawada)

I admit that now we can use yield_self. I didn't think interpolation was elegant enough, but I think I can live with the combination of yield_self and %.

```
["foo", "bar"]
.join(", ")
.yield_self{|s| '<%s>' % s}
# => "<foo, bar>"
```

I am not against closing this issue.

#18 - 04/02/2018 03:34 AM - nobu (Nobuyoshi Nakada)

- Status changed from Feedback to Rejected

#19 - 08/29/2018 09:15 AM - duerst (Martin Dürst)

- Related to Feature #15024: Support block in Array#join added