Ruby master - Feature #17579

[Proposal] A suggestion for newline-separated shorthand notation, for the creation of Arrays containing strings that may contain ' ' (space) characters

01/25/2021 11:15 AM - shevegen (Robert A. Heiler)

Status: Open
Priority: Normal
Assignee:
Target version:

Description

New year, new proposal! :)  

Summary (TL;DR) of the proposal first: It is here proposed that ruby adds functionality to allow ruby users to create arrays, with (string) members inside that array that may contain ' ' (space) characters.

This is akin to, as a shorthand notation, to well known variants like the following one here:

```ruby
%w( foo bar ) # => ["foo", "bar"]
```

(Note that this creates two members in the array, "foo" and "bar"); my suggestion here is about creating a single String with ' ' one space, e. g. "foo bar" in the example above. Note that I do NOT suggest to change the existing %w() syntax; we would need another syntax for this in my opinion, ideally also just one character.)

Since ruby code is often significantly simpler to understand than a long sentence, let me next show you what I mean with the proposal above.

Consider the following array:

```ruby
array = ['foo bar',
         'bar foo',
         'tom the cat',
         'jerry the mouse']
```

Here we have an array with four members; four string objects, each containing at least one ' ' (space) character. It may also include strings that do not contain any ' ' though.

Ruby offers several shorthand notations to create arrays.

A very popular one, and in my (not systematic) opinion probably the most commonly used variant, found in many gems, ruby users use this pattern here commonly:

```ruby
array = %w( one two three )  # or other "container" tokens used, such as:
array = %w[ one two three ]
array = %w{ one two three }
```

Today I had a use case to create the following array:

```ruby
array = ['choice 1',
         'choice 2',
         'choice 3',
         'choice 4',
         'choice 5',
         'choice 6',
         'choice 7',
         'choice 8']
```

This is ok, syntax-wise; also efficient, so the suggestion here is not that important actually.
I also wondered whether we ruby users COULD actually use a shorter variant for precisely situations like the above. (For anyone wondering why I made such a strange array, I actually wanted to create simple radio-buttons for ruby-gtk, as a "drop-down" choice, so I needed a simple array where the user can select one of these different variants, before hitting the "send" or "submit" button. So the above acted mostly as a place holder for prototyping code. In general I make use of %w() a lot though, so perhaps that is why it felt "natural" for me to want to try to look for simpler ways here. %w() is really very efficient IMO.)

Of course my first "logical" attempt would not work:

```ruby
array = %w(
  choice 1
  choice 2
  choice 3
  choice 4
  choice 5
  choice 6
  choice 7
  choice 8
)
```

This of course won't work due to the ' ' and %w() splitting on ' '. But the syntax is quite elegant, yes? Short and succinct - it is nice and efficient.

So, I was then wondering whether we could have a variant that also includes the ' ' and splits on newlines rather than ' '

(For spacing reasons all leading ' ' are also ignored on the newline. This could be argued about either way, but I probably would prefer to be able to retain such a uniform styling in my .rb files, sort of like how %w() can be used. So my whole idea is really mostly aimed at %w(), just with newline splitting rather than '' splitting.)

I wondered whether we could re-use EOF with there here-doc token. Example what I mean here:

```ruby
heredoc = <<-EOF
  choice 1
  choice 2
  choice 3
  choice 4
  choice 5
  choice 6
  choice 7
  choice 8
EOF
```

Just an array, rather than a string. Would be quite efficient. (Evidently heredoc will NOT ignore leading ' ' so it is a bit different anyway.)

But then I was thinking ... I am already using newlines there, for readability purposes, see above. So my array will be spread over multiple new lines. So, one new line, one new array element - quite simple! And this is the gist of my suggestion here really:

- A way to create arrays in ruby based on newlines.

If a variant already exists that splits on newlines by default, please disregard this suggestion then. My mind was thinking mostly about shorthand-notation syntax here.

As for the syntax for this suggestion ... I am not sure ... this is perhaps the hardest part, because the syntax should be short too, as otherwise the advantage of succinctness would be lessened.

So, what do we have available in ruby? Let's look.

We have, for array of strings:

```ruby
%w() # I'll ignore the {} and [], so this is just to compile what we have right now.
```

Array of Symbols:

```ruby
%i( foo bar ) # see the old SO discussion here, if anyone is curious https://stackoverflow.com/post/queue
```
For regular expressions we have:

\%r()

For single-quoted strings and multi-line strings we have:

\%q()

Double-quoted string:

\%Q()

Shell command:

\%x()  # I don't think I ever used this one actually ... I always use system() or `` and sometimes the popen-family

\%s()  # strange one here ... turns foo into a symbol (:foo). Peculiar ...

I quickly compared the last one with \%i():

\%s( foo )  # => :"foo"
\%i( foo )  # => [:foo]

I did not know about \%s. How odd. :)

Anyway. My proposal here is about making the "split" based on newlines, just as the array example above (e. g. array = ['choice 1', etc..) showed.

I am not entirely sure which shorthand syntax should be used. Ideally we should not use too many shorthand notations in general, because ruby users may have a hard time remembering all of this; I can never remember the old perly $ variables, always had to look up in a cheat sheet file or had to back when they were more commonly used.

But, since these elements in the array are still strings, perhaps they should remain with the \%w() syntax family.

So, I was thinking, based on that ... something like \%wn( ) for a newline?

Like:

array = \%wn(
  choice 1
  choice 2
  choice 3
  choice 4
  choice 5
  choice 6
  choice 7
  choice 8
)  # = ['choice 1','choice 2','choice 3','choice 4','choice 5','choice 6','choice 7','choice 8']

Although, I am not sure ... it would be a simple change perhaps, just append a "n" character.

I much prefer single letters, though. So I am not convinced about \%wn. The "n" is meant to stand for "newline" or "newlines".

Perhaps \%n( ) for newline could be used? Or \%W( ) could be used? Hmmmmm.

Syntax choice is sooooo difficult if you want to make a good choice that people could use. Perhaps \%W() is not so good because people may make a typo, and confuse it with \%w. So \%n may be better. Or \%wn. (I am trying to "remain" in a syntax "family", so that it may not be too obscure, although we could use another letter.)

But anyway, I believe it may be better to first decide whether this feature is good, useful, or totally pointless. And, based on that, e. g. if it is considered useful (possibly so), to then think more about the syntax choice for it. After all, if the idea is not useful, there is little point in wanting to find a useful syntax for it.

Please anyone feel free to comment if you would like to, and give ideas, pros and cons as to whether this may be useful to have or
not. I think this can only be potentially useful IF other ruby users may have had a similar use case (because otherwise nobody would use it, so then it would be a mostly "dead" feature), but I am actually not even absolutely certain that the proposal is that useful.

It is probably quite a niche case idea; but who knows, perhaps at some later time, suggestions and ideas may be picked up again, so I think it would be useful to comment on this suggestion anyway, if only for future references alone. (Or perhaps it was already suggested years ago, I do not know.)

Hopefully I managed to make this suggestion readable; I was writing this in my local editor, as I have a hard time using webforum-based HTML interfaces in general.

At any rate, thanks for reading!

History
#1 - 01/25/2021 01:00 PM - decuplet (Nikita Shilnikov)
Didn't have time to read it fully but you can escape spaces:

```ruby
2.7.1 :001 > %w(foo bar baz)
=> ["foo bar", "baz"]
```

Another option

```ruby
options = <<~EOF.lines.map(&:strip)
  choice 1
  choice 2
  choice 3
  choice 4
  choice 5
  choice 6
  choice 7
  choice 8
EOF
# => ["choice 1", "choice 2", "choice 3", "choice 4", "choice 5", "choice 6", "choice 7", "choice 8"]
```

Ignore if it's irrelevant.

#2 - 01/25/2021 03:11 PM - marcadre (Marc-Andre Lafontune)
I'm curious what is your use case. I can't recall needing an array of such strings. If I did, I would probably put these in a YAML file.

#3 - 01/25/2021 04:33 PM - Dan0042 (Daniel DeLorme)
I would also use heredoc for this, maybe just with one tweak:

```ruby
options = <<~EOF.chomp: true
  choice 1
  choice 2
  choice 3
EOF
# => ["choice 1", "choice 2", "choice 3"]
```

There's a few things combined here that I believe are not so commonly known

- you can put ruby code after the heredoc start token
- `<<~EOF` (with tilde) will strip the leading indentation
- `lines` has a `chomp` argument to minimize object allocations