Hi,

I have been using this String#squish method so many time when I'm using Rails, and I think it should be a useful addition to core.

Here's the method on Rails' documentation: http://api.rubyonrails.org/v4.2.5/classes/String.html#method-i-squish

This method is very useful when you have to write a multi-line string using heredoc, but you actually does not care about the white-spaces before, after, and in-between the string.

For example:

```
<<-SQL.squish
  SELECT *
  FROM users
  WHERE users.username = 'sikachu'
SQL
#=> "SELECT * FROM users WHERE users.username='sikachu'"
```

Another example usage is when you are on the project that have a line length code standard, and you have to write a long warning message that needs to be printed to stdout:

```
puts <<-WARNING.squish
  Unable to connect to the server. Please double-check that you are currently connecting to the internet and your proxy server is working.
WARNING
#=> Unable to connect to the server. Please double-check that you are currently connecting to the internet and your proxy server is working.
```

By the way, this is my first patch and my first time writing something in C, so there might be something that does not look right to you. I'll happy to revise this patch (and learn about C in the process!) from your feedback.

Thank you,
Prem

I also just created a GitHub PR in case that will be easier to give feedback to the code: https://github.com/ruby/ruby/pull/1113

Thank you,
Prem
You could memoize these Regexps as static variables and use `rb_gc_register_mark_object` to keep them around so GC won't eat them.
Allocating 3 regexps and 3 strings every call seems like a waste.
You may also use the same

Writing the equivalent Ruby code would only allocate the Regexps once.

You can also auto-dedupe "" and " " strings with the magic "frozen_string_literal: true" comment in Ruby or `rb_fstring_cstr` function in C.

By the way, this is my first patch and my first time writing something in C, so there might be something that does not look right to you. I'll happy to revise this patch (and learn about C in the process!) from your feedback.

No worries; but personally (not speaking for the rest of ruby-core):
I would prefer we use prelude.rb more and implement more things in Ruby rather than C.

Also (definitely not speaking for anybody else in ruby-core):
but the Redmine <-> ruby-core ML integration is the only reason I've been willing to participate in Ruby development. I'm not touching proprietary websites or running any GUI/JavaScript at all to work on Ruby or any other Free Software.

---

#4 - 11/25/2015 04:20 AM - nobu (Nobuyoshi Nakada)
- Description updated

First, indented heredoc will be achieved by [Feature #9098](https://github.com/ruby/ruby/compare/trunk...nobu:feature/11735-squish), this will not be needed.

Second, squish! feels to return nil if no white spaces, to me, like as sub! and so on.

Last, it's prohibited in C89 to initialize an aggregate type by dynamic values.

#5 - 11/25/2015 05:03 PM - deivid (David Rodríguez)

I don't think the two features are the same, and would find both of them useful.

#6 - 12/07/2015 07:50 AM - ko1 (Koichi Sasada)
- Assignee set to matz (Yukihiro Matsumoto)

#7 - 12/07/2015 07:51 AM - nobu (Nobuyoshi Nakada)

https://github.com/ruby/ruby/compare/trunk...nobu:feature/11735-squish

#8 - 12/07/2015 02:19 PM - sikachu (Prem Sichanugrist)

nobu's patch seems to be the better way to implement this feature without having to use regular expression. Much more efficient.

I guess I should try to think outside the box the next time I try to write something in C.

+1 to nobu's patch. Thank you for your hard work.

#9 - 05/18/2016 01:22 AM - shyouhei (Shyouhei Urabe)
We looked at it on yesterday's developer meeting but didn't reach a consensus. This might be because the attendees are mostly Hanzi culture
residents (they treat newlines and spaces differently). This doesn't mean an immediate NG but frankly, we need to study real-world use cases more.

Note #1: recent (2.3+) ruby has squiggly heredoc ("<<~"). This can save someone who originally used squish.

Note #2: as we discuss, I reached a conclusion that squish over SQL is dangerous. It can destroy SQL's string literals with spaces.

#10 - 05/19/2016 05:57 PM - gorgeous (Adam Doppelt)
It would be great to include squish in String. I've been writing production Ruby code for years and I often pull in ActiveSupport just to get squish.

Getting input from a user? squish
Cleaning up an iffy array.join? squish
Pulling data from a web crawl? squish
Normalizing concatenated data? squish

Squish squish squish. 60k hits on github for Ruby squish. Just squish it. Squish it into core, please.

#11 - 05/20/2016 02:38 AM - shyouhei (Shyouhei Urabe)
Adam Doppelt wrote:

Getting input from a user? squish

I guess you have never had a user from CJKV cultures.

Cleaning up an iffy array.join? squish

This is a huge NO. It destroys JSON.

Pulling data from a web crawl? squish

Also NO. It destroys <pre>.

Normalizing concatenated data? squish

This could be OK depending on the "normalization" you want.

Squish squish squish. 60k hits on github for Ruby squish. Just squish it. Squish it into core, please.

I had no pro et contra to the proposal until now. From what you said I started thinking squish can be a bad smell of indiscreet data treatment.

I hope I'm wrong.

#12 - 06/23/2016 05:33 PM - gorgeous (Adam Doppelt)
Shyouhei Urabe wrote:

I had no pro et contra to the proposal until now. From what you said I started thinking squish can be a bad smell of indiscreet data treatment.

I think you're missing the point here. Squish is used to cleanup whitespace. If you want to preserve whitespace, don't call it. There are many, many, many (many) situations where cleaning up whitespace is desirable and squish is perfect.

Yes, I actually need to destroy newlines, smash spaces together, and annihilate tags. This is incredibly common. That's why I use squish. If I didn't want to do those things I would not use squish.

#13 - 06/24/2016 08:02 AM - shyouhei (Shyouhei Urabe)
I don't stop you to do what you want.

But you are requesting it into core. There are so many miserably misused APIs around the world. Isn't it just another example of such thing? Isn't it "too easy to fail"? For instance is it OK say that applying this method to user input wont introduce SQL injection or that sort of things?