Ruby - Feature #17771

String#start_with? should not construct MatchData or set $~

04/01/2021 06:08 PM - headius (Charles Nutter)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

I am working on making $~ more thread-safe in JRuby and came across this unexpected behavior:

```
$ rvm ruby-3.0 do ruby -e '"foo".start_with?(/foo/); p $~'
#<MatchData "foo">
```

The start_with? method was added 11 years ago in https://bugs.ruby-lang.org/issues/3388 but I do not think the set of $~ was an intended feature. The start_with? method could be much faster and more thread-safe if it did not use the frame-local backref slot and did not allocate a MatchData.

Compare with match? which was added specifically (without MatchData or backref setting) to provide a fast way to check if a Regexp matches.

I propose that start_with? stop constructing MatchData, stop setting backref, and provide only its boolean result in the same way as match?.

History

#1 - 04/01/2021 06:13 PM - headius (Charles Nutter)

I will also point out that this method, like many others, will *not* always set $~. If you pass a string, it remains whatever it was before:

```
$ rvm ruby-3.0 do ruby -e '"foo".start_with?("foo"); p $~'
nil
```

Avoiding the use of $~ would make this behavior consistent.

#2 - 04/01/2021 06:25 PM - headius (Charles Nutter)

I see this behavior was explicitly blessed by matz in #13712 but I still believe this is not the best choice.

Around the same time as that discussion, another boolean query method match? was added that explicitly does *not* set the last match frame variable.

I feel this is inconsistent and the boolean query methods that accept a Regexp should be as fast as possible. If you want a MatchData use methods that provide it.

#3 - 04/01/2021 06:53 PM - jeremyevans0 (Jeremy Evans)

- Tracker changed from Bug to Feature

- Backport deleted (2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN, 3.0: UNKNOWN)

#4 - 04/01/2021 07:04 PM - enebo (Thomas Enebo)

It really feels like an unintended side-effect of the method. If you write this method and accept a variable then depending on the type of that variable there is either some MatchData (MD) as a side-effect or there isn't. This is inconsistent. If you wanted to explicitly use MD then you have to know what you are supplying. If you know it is a regexp then just writing str =~ /^my_pat/ is what you want.

#5 - 04/01/2021 07:06 PM - headius (Charles Nutter)

An alternative to using str =~ /^pat/ for a start_with? that provides a MatchData would be to add a start_with that is not a boolean query method.

#6 - 04/02/2021 10:28 AM - Eregon (Benoit Daloze)

I don't think there is a rule that predicate methods only return a boolean and never set $~.
It is the case for String#match vs String#match?, but it doesn't mean it holds for other Regexp methods.
I see it a bit like the use of !, which in the core library is generally only used if there is also a non-! variant (e.g., Array#delete).

String#start_with? enables to match a regexp without the need to manually build another regexp like /\A#{regexp}/ (from the user point of view, there might be internal caching depending on the regexp engine), so I think that is a valid use case for using start_with? and accessing the MatchData after.

StringScanner has a similar functionality for matching a regexp from the start, as if there was a \A, but does not expose $~ directly:
ruby -rstrscan -e 's = StringScanner.new("test string"); s.scan(/(\w)\w+/); p s[1]' => "t".

That said, I'm not against no longer setting $~ for String#start_with?, but I do worry about the compatibility issue here, especially since it might be quite hard to debug why $~ is suddenly nil or the previous MatchData in the Ruby version changing this behavior.

#7 - 04/02/2021 02:49 PM - marcandre (Marc-Andre Lafortune)

I also believe it is unintended behavior and should be removed.