Ruby master - Bug #7566

Escape (\u{}) forms in Regexp literals

12/15/2012 10:06 AM - brixen (Brian Shirai)

Status:	Rejected		
Priority:	Normal		
Assignee:			
Target version:	2.0.0		
ruby -v:	ruby 1.9.3p327 (2012-11-10 revision 37606) [x86_64-darwin10.8.0]	**Backport:**	

Description

Why are \u{} escape sequences in Regexp literals not converted to bytes like they are in String literals?

https://gist.github.com/4290155

Thanks,
Brian

History

#1 - 12/15/2012 10:53 AM - drbrain (Eric Hodel)

- Category set to core

- Target version set to 2.0.0

=begin
Converting any of the regexp special characters could cause a syntax error or warning if the user tries to round-trip the regexp, so I think this is not a bug:

```
$ ruby20 -ve 'p("\u{5d}", /[\u{5d}]/)'
ruby 2.0.0dev (2012-12-15 trunk 38385) [x86_64-darwin12.2.1]
"]"
/[\u{5d}]/

$ ruby20 -ve 'p(/[]]/)'
ruby 2.0.0dev (2012-12-15 trunk 38385) [x86_64-darwin12.2.1]
-e:1: warning: character class has ']' without escape: /[]]/
/[]]/
```

=end

#2 - 12/16/2012 03:13 AM - brixen (Brian Shirai)

I'd argue that's a malformed Regexp and "round-tripping" shouldn't be expected to work.

```
sasha:rubinius brian$ irb
1.9.3p327 :001 > re = /[\\u{5d}]/
=> /[\\u{5d}]/
1.9.3p327 :002 > re2 = Regexp.new re
=> /[\\u{5d}]/
1.9.3p327 :003 > re3 = Regexp.new re.source
=> /[\\u{5d}]/
1.9.3p327 :004 > "ab]c" =~ re
=> 2
1.9.3p327 :005 > "ab]c" =~ re2
=> 2
1.9.3p327 :006 > "ab]c" =~ re3
=> 2
```

The consequence of storing the source with escape sequences and the fact that 7-bit clean source even using UTF escapes is encoded as US-ASCII is that the underlying Oniguruma data must be maintained separately and the string potentially unescaped every match. At least, that is the best understanding I have of the MRI source code. AFAIK, this is not defined anywhere.

Thanks,
Brian

#3 - 12/17/2012 11:12 AM - naruse (Yui NARUSE)

- Status changed from Open to Rejected

Because Regexp Literals are not String Literals, and escapes in them have different meanings.
For example \b, it is word boundary in Regexp but BEL in String.
People will need to distingish word boundary from BEL, so \b must be showed as \b.
\uXXXX follows such style.

#4 - 12/17/2012 11:38 AM - brixen (Brian Shirai)

Are you saying you can represent \b as a \u{} escape sequence in a Regexp?

#5 - 12/17/2012 11:49 AM - naruse (Yui NARUSE)

brixen (Brian Ford) wrote:

> Are you saying you can represent \b as a \u{} escape sequence in a Regexp?

No.
(1) \b (word boundary), \s (spaces and tabs) and so on are can't expressed as bytes
(2) so escapes are not converted to bytes, kept as is
(3) \u{} is also escape, so kept as is

#6 - 01/03/2013 03:37 AM - brixen (Brian Shirai)

But as my example shows, if the bytes were in a literal String used to create the Regexp, they are already converted. And everything works just fine.

What's the rationale for not converting \u{}? Just because it is *an* escape sequence doesn't mean it is a *Regexp* escape sequence. Why are they treated the same? It creates inconsistency between two identical Regexps except that one came from a String or Regexp literal with interpolation.

#7 - 01/03/2013 05:42 AM - phluid61 (Matthew Kerwin)

brixen (Brian Ford) wrote:

> But as my example shows, if the bytes were in a literal String used to create the Regexp, they are already converted. And everything works just fine.

No it doesn't. There are no literal strings in your example. The closest I can see is you extracting a source string from the Regexp, but I don't think that's doing what you think it is.

```
irb(main):001:0> re = /[\\u{5d}]/
=> /[\\u{5d}]/
irb(main):002:0> re.source
=> "[\\\u{5d}]"
```

If you meant this:

```
irb(main):003:0> s = "[\\u{5d}]"
=> "[\]]"
irb(main):004:0> re2 = Regexp.new s
=> /[]]/
```

You get an entirely different Regexp. They will both match the string "ab]c" because they both include the ']' character in their character class.
Incidentally:

```
irb(main):005:0> re =~ "ab\c"
=> 2
irb(main):006:0> re2 =~ "ab\c"
=> nil
```

> What's the rationale for not converting \u{}? Just because it is *an* escape sequence doesn't mean it is a *Regexp* escape sequence. Why are they treated the same?

They aren't. If it helps, consider that <u>no</u> Regexp escape sequences are treated the same as String escapes.

\ is a String literal escape sequence that is interpolated to the byte \x5C
\ is a Regexp literal escape sequence that instructs the engine to match the byte \x5C

\u{} is a String literal escape sequence that is interpolated to a codepoint
\u{} is a Regexp literal escape sequence that instructs the engine to match a codepoint

\b is a String literal that is interpolated to the byte \x08
\b is a Regexp literal that instructs the engine to match a word boundary

> It creates inconsistency between two identical Regexps except that one came from a String or Regexp literal with interpolation.

No, if the Regexps were identical they would be identical. As you can see above, re and re2 are not identical, and no one should expect them to be.