Ruby master - Bug #5871
regexp \W matches some word characters when inside a case-insensitive character class
01/10/2012 01:25 AM - garethadams (Gareth Adams)

Status: Rejected
Priority: Normal
Assignee: 
Target version: ruby -v:
ruby 1.9.2p290 (2011-07-09 revision 32553) [x86_64-darwin10.8.0]

Description
=end
The following replacement, which should do nothing, has removed the upper- and lower-case "K"s and "S"s from the result:

> "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz".gsub(/\[\W\]/i,"")
=> "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"

The result is correct (the same as the input string) if I remove either the character class:

> "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz".gsub(/\W/i,"")
=> "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"

or the case insensitive flag:

> "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz".gsub(/\[\W\]/,"")
=> "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"

This has been observed in two separate ruby 1.9 installs:

- ruby 1.9.2p290 (2011-07-09 revision 32553) [x86_64-darwin10.8.0]
- ruby 1.9.3p0 (2011-10-30 revision 33570) [x86_64-darwin11.2.0]

but works correctly in 1.8
=end

Related issues:
Is duplicate of Ruby master - Bug #4044: Regex matching errors when using \W ...
Closed

History
#1 - 01/10/2012 01:40 AM - garethadams (Gareth Adams)
=end
As a simpler test case:

> "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz".scan(/\[\W\]/i)
=> ["K", "S", "k", "s"] # should be []
=end

#2 - 01/10/2012 01:46 AM - garethadams (Gareth Adams)
I've now also seen at least one report that this doesn't affect 1.9.3p0 (win32)

#3 - 01/10/2012 01:55 AM - kyrylo (Kyrylo Silin)
This happens to me too with ruby 1.9.3p0 (2011-10-30 revision 33570) [i686-linux]

#4 - 01/10/2012 02:06 AM - garethadams (Gareth Adams)
=end
Thanks to investigation from #ruby-lang, It seems this issue only occurs with UTF-8 strings

ruby-1.9.2-p290> "KSks".encode("UTF-8").scan(/\W/) != "KSks".encode("US-ASCII").scan(/\W/)
=> true
#5 - 01/10/2012 08:12 AM - naruse (Yui NARUSE)

- Status changed from Open to Rejected

It is spec as written at #4044.

#6 - 01/10/2012 10:55 AM - shyouhei (Shyouhei Urabe)

Quite generally speaking you are advised not to use /i in Unicode. The reason? because Babylonians did something wrong.

In this specific case the [\W], which equals to [^A-Za-z]; includes K and ß. So ![\W]i includes k and SS.

#7 - 01/10/2012 04:02 PM - duerst (Martin Dürst)

- Status changed from Rejected to Open

Shouhei Urabe writes:

Quite generally speaking you are advised not to use /i in Unicode.

Are there other examples where /i is advised against? If yes, please let's look at them and try to fix them, too.

The reason? because Babylonians did something wrong.

Many problems can be (figuratively) blamed on the Babylonians, but not this one.

In this specific case the ![\W], which equals to [^A-Za-z]; includes K and ß. So ![\W]i includes k and SS.

Let's look at this in detail. At https://bugs.ruby-lang.org/issues/4044#note-9, Yui Naruse writes:

Unicode ignore case breaks it.

http://unicode.org/reports/tr21/

That link says “Superseded Unicode Standard Annex”. It gives three locations for the information, http://www.unicode.org/versions/Unicode4.0.0/ch03.pdf#G33992, http://www.unicode.org/versions/Unicode4.0.0/ch04.pdf#G124722, and http://www.unicode.org/versions/Unicode4.0.0/ch05.pdf#G21180. In the archival version of tr21, at http://www.unicode.org/reports/tr21/tr21-5.html, I find the word "ignore" just two times, and I didn’t find a definition of "ignore case". Can somebody tell me exactly what is meant?

I don’t assume that the Unicode Standard would define or imply that ‘k’ or ‘S’ are non-word characters. However, if indeed there is some data or text in the Unicode Standard that defines or implies this, then that would need to be fixed urgently, and I’d like to help.

212A; C; 006B; # KELVIN SIGN
00DF; F; 0073 0073; # LATIN SMALL LETTER SHARP S
http://www.unicode.org/Public/UNIDATA/Casemap.txt

\W includes U+212A and U+00DF
\i adds U+006B (k) and U+0073 (S) to ![\W]
^ reverses the class; it doesn't include k & S.

Because of “the Babylonians”, it is frequently the case that some property that applies in a limited character set (e.g. the character set of US-ASCII) doesn't apply directly in a wider character set (e.g. the Unicode character set). In that case, rather than blaming the problem on "the Babylonians", what needs to be done is: 1) Analyse the problem, to figure out what assumptions are no longer guaranteed. 2) Think about what programmers/users would most reasonably expect. 3) Figure out how to fix the implementation so that expectations are met even without the previously valid assumptions.

In our case, we have the assumption that the negation of a character class does not include any characters of that class. For ASCII, that’s true. For Unicode, as currently implemented, it’s not true, but that’s only because the Unicode case tables haven’t been used correctly. When it comes to "the Babylonians", there isn’t a one-to-one case mapping, and as a consequence, one-way case mapping and case equivalence behave somewhat differently. I think what should be implemented is that the \w (Word character) class is defined on round-trip case equivalence (which would include U+212A and U+00DF), not as apparently currently the case on one-way case mappings. The use of round-trip case equivalence may also be appropriate for other operations in the regular expression implementation, but this needs to be checked.

Anyway, an implementation that claims that ‘k’ and ‘S’ are non-word characters is fundamentally broken, and we have to fix it. I have therefore reopened the bug. (Sorry, I was not aware of https://bugs.ruby-lang.org/issues/4044, otherwise I'd have explained things then.)

The question of whether to use round-trip case equivalence (which is appropriate e.g. for search) or only some more limited case operation also comes up in other circumstances. As an example, IDNA 2003 defines that β (U+00DF) mapps to ‘ss’, but in the context of domain names, that turned
out to be the wrong choice, because it means that it is impossible to use β in internationalized domain names. This was fixed in IDNA 2008.

#8 - 01/10/2012 04:26 PM - naruse (Yui NARUSE)
- Status changed from Open to Rejected

Please suggest concrete plan.
And if you reopen, please write it to #4044.

#9 - 01/10/2012 05:18 PM - shyouhei (Shyouhei Urabe)
Martin Dürst wrote:

Shouhei Urabe writes:

Quite generally speaking you are advised not to use /i in Unicode.

Are there other examples where /i is advised against? If yes, please let's look at them and try to fix them, too.

/Dijkstra/i.match("DIJKSTRA") or something like that.

#10 - 01/10/2012 05:51 PM - duerst (Martin Dürst)
Shouhei Urabe writes:

Martin Dürst wrote:

Shouhei Urabe writes:

Quite generally speaking you are advised not to use /i in Unicode.

Are there other examples where /i is advised against? If yes, please let's look at them and try to fix them, too.

/Dijkstra/i.match("DIJKSTRA") or something like that.

What about /Dijkstra/.match("Dijkstra")?
$ ruby -e "puts /Dijkstra/.match("Dijkstra").inspect"
nil

If this doesn't match without case equivalence, why should it match with case equivalence?
(I'm assuming that matching is transitive and that matching by /i should be a superset of matching without.)

#11 - 01/10/2012 06:07 PM - naruse (Yui NARUSE)
Martin Dürst wrote:

Shouhei Urabe writes:

Martin Dürst wrote:

Shouhei Urabe writes:

Quite generally speaking you are advised not to use /i in Unicode.

Are there other examples where /i is advised against? If yes, please let's look at them and try to fix them, too.

/Dijkstra/i.match("DIJKSTRA") or something like that.

What about /Dijkstra/.match("Dijkstra")?
$ ruby -e "puts /Dijkstra/.match("Dijkstra").inspect"
nil

It is not an issue of case equivalence.

If this doesn't match without case equivalence, why should it match with case equivalence?
(I'm assuming that matching is transitive and that matching by /i should be a superset of matching without.)
So regular expressions don't offer level 1 basic unicode support?
See http://unicode.org/reports/tr18/

On Tue, Jan 10, 2012 at 06:07:13PM +0900, Yui NARUSE wrote:

Issue #5871 has been updated by Yui NARUSE.

Martin Dürst wrote:

Shohei Urabe writes:

Martin Dürst wrote:

Shohei Urabe writes:

Quite generally speaking you are advised not to use /i in Unicode.

Are there other examples where /i is advised against? If yes, please let's look at them and try to fix them, too.

/Dijkstra/i.match("DIJKSTRA") or something like that.

What about /Dijkstra/.match("Dijkstra")?

$ ruby -e "puts /D\u0133kstra/.match('Dijkstra').inspect"
nil

It is not an issue of case equivalence.

If this doesn't match without case equivalence, why should it match with case equivalence?
(I'm assuming that matching is transitive and that matching by /i should be a superset of matching without.)

Bug #5871: regexp \W matches some word characters when inside a case-insensitive character class
https://bugs.ruby-lang.org/issues/5871

Author: Gareth Adams
Status: Rejected
Priority: Normal
Assignee:
Category:
Target version:
ruby -v: ruby 1.9.2p290 (2011-07-09 revision 32553) [x86_64-darwin10.8.0]

=begin
The following replacement, which should do nothing, has removed the upper- and lower-case "K"s and "S"s from the result:

> "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz".gsub(/\W/i,"")
=> "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"

The result is correct (the same as the input string) if I remove either the character class:

> "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz".gsub(/\W/i,"")
=> "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"

or the case insensitive flag:

> "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz".gsub(/\W/i,"")
=> "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
=end
This has been observed in two separate ruby 1.9 installs:

- ruby 1.9.2p290 (2011-07-09 revision 32553) [x86_64-darwin10.8.0]
- ruby 1.9.3p0 (2011-10-30 revision 33570) [x86_64-darwin11.2.0]

but works correctly in 1.8
=end

--
http://bugs.ruby-lang.org/

--

old inkjet cartridges emanate barium-based fumes

#13 - 01/14/2012 08:46 AM - naruse (Yui NARUSE)

Ondrej Bilka wrote:

So regular expessions dont offer level1:basic unicode support?
See http://unicode.org/reports/tr18/

We don't target on tr18 level 1 now.
But Ruby may support some parts of tr18.
You can request a feature with use case.