Ruby master - Bug #15876

1.to_s.encoding != Encoding.default_internal

05/25/2019 05:40 PM - grosser (Michael Grosser)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	2.6.3	**Backport:**	2.4: UNKNOWN, 2.5: UNKNOWN, 2.6: UNKNOWN

Description

I ran into strange looking test output when I compared .to_s with an expected text, saying that the encoding was different, which is confusing/annoying especially to users that don't know how encodings work in ruby.
1.to_s.encoding should be the same as "".encoding

History

#1 - 05/25/2019 10:25 PM - shevegen (Robert A. Heiler)

> which is confusing/annoying especially to users that don't know
> how encodings work in ruby.

I personally finally switched into UTF-8 (oddly enough, primarily due to emoji and unicode-symbols that can be used for simple indications both on the commandline and www), but I think one problem (for me) was from ruby 1.8.x to later ruby versions that there was not that much documentation available.

Judging from your comment encoding may still pose a problem for some ruby users (or potentially new ruby users).

Some time ago, I think, jeremy evans wrote a document about symbols, which was added (my apologies if I misremember). If anyone feels like writing some document about encoding in ruby, and how to deal with it ... :) (could be in wiki-style or perhaps gist-github or some other place; I am in no way suggesting that only a single person should do so, it could be a collaborative effort).

To the issue at hand, I just tested in irb:

```
1.to_s.encoding  #should be the same as "".encoding # => #<Encoding:US-ASCII>
"".encoding # => #<Encoding:UTF-8>
```

This is indeed a little surprising (to me). There may be valid reasons for this, perhaps default external encoding, or something like this, but I can see why people may be confused about it. Actually what surprises me is that .to_s on the number leads to US-ASCII encoding by default.

I think looking back when I used an ISO-encoding, the most surprising result I had encountered was actually in regards to regexp-engine and encodings used there. I do not remember exactly how I found it, but I think I reported it back then; still not entirely sure how it came, but regexes may also be an area where users may be a little bit confused - so documentation may be of some help.

#2 - 05/26/2019 12:34 AM - mame (Yusuke Endoh)

grosser (Michael Grosser), could you elaborate your problem? I cannot reproduce the warning. What warning did you see? And how?

```
s1 = 1.to_s
p s1.encoding #=> #<Encoding:US-ASCII>

s2 = "1"
p s2.encoding #=> #<Encoding:UTF-8>

p s1 == s2 #=> true with no warning
```

#3 - 05/26/2019 02:30 AM - duerst (Martin Dürst)

mame (Yusuke Endoh):

What grosser (Michael Grosser) is saying is that

```
p s1.encoding == s2.encoding #=> false
```

but he expects the result to be true. But you are right that what counts is the equality of the strings, not the encodings.

#4 - 05/26/2019 03:08 AM - mame (Yusuke Endoh)

grosser (Michael Grosser) said

> I ran into strange looking test output when I compared .to_s with an expected text, saying that the encoding was different

I thought that some string-comparison assertions (maybe attributed to an external testing framework?) emitted a spurious warning like "the encoding was different" or something.

#5 - 05/31/2019 04:10 PM - naruse (Yui NARUSE)

- Status changed from Open to Feedback

What is the problem you are actually troubled with?

If it is just a testing problem, I feel it should just use correct assertions.
But if there's a frequent pitfall, I may reconsider it.

#6 - 05/31/2019 05:37 PM - Hanmac (Hans Mackowiak)

There is Encoding.compatible? which might help to check if two strings/symbols has a common encoding

naruse (Yui NARUSE) i don't know if you are the right contact person for this, but is there a way to see if two encoding objects are compatible or can that only be checked on the string?

#7 - 05/31/2019 08:29 PM - Eregon (Benoit Daloze)

Hanmac (Hans Mackowiak) wrote:

> is there a way to see if two encoding objects are compatible or can that only be checked on the string?

Encoding.compatible? can take Encoding arguments too:

```
> Encoding.compatible?(Encoding::UTF_8, Encoding::US_ASCII)
=> #<Encoding:UTF-8>
```

#8 - 08/14/2019 03:08 AM - jeremyevans0 (Jeremy Evans)

- Status changed from Feedback to Closed