When converting an instance of Rational/Float/Fixnum/Bignum to a string with the `.to_s` method, the resulting string has the encoding US-ASCII. This happens for 1.9.3 as well as 2.0.0rc1.

```
#ENCODING#
# Encoding:UTF-8#
# Encoding:UTF-8#
# Encoding:UTF-8#

# 1.to_s.encoding#
# Encoding:US-ASCII#

# (2/1).to_r.to_s.encoding#
# Encoding:US-ASCII#

# "abc".encoding#
# Encoding:UTF-8#
```

This behavior matches Time#to_s, see #5226

Since there are no non-US-ASCII characters in the result of `to_s` on Rational, Float, Fixnum or Bignum there should be no problem with the US-ASCII encoding. Can you demonstrate one?

The only problem I see is that ruby is lying to the user. It is not severe since, as you said, there are no non-ascii characters in the resulting string, but I think ruby should respect the set encoding.

US-ASCII is a strict subset of UTF-8, so I don't think there's necessarily any lying involved.

On current policy, strings which always include only US-ASCII characters are US-ASCII. If there is a practical issue, I may change the policy in the future.

Note that US-ASCII string is faster than UTF-8 on getting length or index access.
On 2013/01/31 18:07, coffeejunk (Maximilian Haack) wrote:

Issue #7752 has been updated by coffeejunk (Maximilian Haack).

The only problem I see is that ruby is lying to the user.

There is 0% lying if one claims that an ASCII-only string is US-ASCII. There is also 0% lying if one claims it's UTF-8.

It is not severe since, as you said, there are no non-ascii characters in the resulting string, but I think ruby should respect the set encoding.

Setting Encoding.default_internal (or something else) is not a guarantee that all Strings will be in that encoding. Otherwise, it wouldn't be called "default".

Regards, Martin.

Bug #7752: Rational/Float/Fixnum/Bignum .to_s.encoding is US-ASCII
https://bugs.ruby-lang.org/issues/7752#change-35742