# Ruby - Feature #8848

## Syntax for binary strings

08/31/2013 03:59 PM - duerst (Martin Dürst)

| | |
|---|---|
| **Status:** | Open |
| **Priority:** | Normal |
| **Assignee:** | |
| **Target version:** | |

**Description**

In commit 37486, Yui (Naruse) added a String#b method as proposed in http://bugs.ruby-lang.org/issues/6767.

String#b was added to allow easy generation of binary strings; this became necessary in particular after the source file encoding was changed to UTF-8.

However, as also recognized in http://bugs.ruby-lang.org/issues/6767, in the long term (ideally starting with Ruby 2.1) it would be better to make binary strings available as part of Ruby syntax.

One reason for this efficiency. String#b creates a duplicate object, which is not at all necessary for the frequent use case of String literals.

Another reason is encoding validity. To be able to e.g. create a "\xFF" binary string, with String#b in an UTF-8 source context, it is necessary to allow "\xFF" (temporarily at least) as an (actually invalid) UTF-8 string. This may be difficult for some implementations, and isn't desirable in general.

Regarding syntax, there are mainly two solutions:

1. a '%b' prefix
2. a 'b' suffix

The preferable syntax depends on the overall future approach of Ruby to String literal suffixes (see https://bugs.ruby-lang.org/issues/8579).

**Related issues:**

| | | |
|---|---|---|
| Related to Ruby - Feature #10391: Provide %eISO-8859-1'string \xAA literal' s... | **Open** | **10/16/2014** |

### History

#### #1 - 10/27/2014 09:35 AM - duerst (Martin Dürst)

*- Related to Feature #10391: Provide %eISO-8859-1'string \xAA literal' string literals with explicit encoding added*

#### #2 - 10/27/2014 06:43 PM - normalperson (Eric Wong)

duerst@it.aoyama.ac.jp wrote:

> One reason for this efficiency. String#b creates a duplicate object,
> which is not at all necessary for the frequent use case of String
> literals.

Avoiding one allocation is easy to add to [Feature #10423]
(which avoids string literal allocations for many methods)

> Another reason is encoding validity. To be able to e.g. create a
> "\xFF" binary string, with String#b in an UTF-8 source context, it is
> necessary to allow "\xFF" (temporarily at least) as an (actually
> invalid) UTF-8 string. This may be difficult for some implementations,
> and isn't desirable in general.

We can even go farther than #10423 and move the evaluation of
"string literal".{b,encode,force_encoding} to compile time.

The downside is compatibility with people who wish to override one of
those methods, but doubt anybody overrides those...
There's no new (and strange looking, IMHO) syntax to learn,
it looks like a normal method call, and the optimization would be

usable with existing code.

#### #3 - 10/28/2014 08:33 AM - duerst (Martin Dürst)

Eric Wong wrote:

> We can even go farther than #10423 and move the evaluation of
> "string literal".{b,encode,force_encoding} to compile time.
>
> The downside is compatibility with people who wish to override one of
> those methods, but doubt anybody overrides those...

Even if nobody overrides String#encode, they may configure it in various ways.

> There's no new (and strange looking, IMHO) syntax to learn,
> it looks like a normal method call, and the optimization would be
> usable with existing code.

It's not enough to move evaluation to compile time. We may want to know the desired encoding before we start to parse the string. That by definition doesn't work when the method (or whatever) comes after the end of the literal.

#### #4 - 12/23/2021 11:43 PM - hsbt (Hiroshi SHIBATA)

*- Project changed from 14 to Ruby*