frozen_string_literal の話

Kazuhiro NISHIYAMA

第76回 Ruby関西 勉強会

2017-01-14

自己紹介

- 西山和広
- id:znz (github, twitter など)
- Ruby コミッター

`frozen_string_literal` とは?

- ruby 2.3.0 以降
- 文字列リテラルで生成される文字列を freeze されたものにする機能
- たとえば `''.frozen?` が `true` になる

使い方

- ファイルの頭にマジックコメント (ファイルごとに指定)

```
# frozen_string_literal: true
```

- 起動時のオプション (プロセス全体のデフォルトを指定)

```
--enable=frozen_string_literal
```

優先度

- 両方指定された場合はマジックコメント優先

オプションの例

```
% ruby --enable=frozen_string_literal \
    -e 'p "".frozen?'
true
% ruby --disable=frozen_string_literal \
    -e 'p "".frozen?'
false
```

- 対応していない gem が存在する可能性があるので enable にするときは動作確認必須

マジックコメントの例

shebang, coding もつけるときはこの順番

```
% cat a.rb
#!/usr/bin/ruby
# coding: ascii-8bit
# frozen_string_literal: true
p ''.frozen?
p ''.encoding
% ruby a.rb
true
#<Encoding:ASCII-8BIT>
```

利点

オブジェクト生成が減る

```
% ruby -e "2.times { p ''.object_id }"
16501180
16501040
% cat a.rb
# frozen_string_literal: true
2.times { p ''.object_id }
% ruby a.rb
11393540
11393540
```

"".freeze

ruby 2.1.0 から "文字列リテラル".freeze は最適化が入っている

```
% ruby -e "2.times { p ''.freeze.object_id }"
16865720
16865720
```

欠点

破壊的変更が必要な場合に対処が必要

```
% cat a.rb
# frozen_string_literal: true
s = ''
s << 'x'
p s
% ruby a.rb
a.rb:3:in `<main>': can't modify frozen String (RuntimeError)
```

破壊的変更が必要な場合

String#dup を使う

```
% cat a.rb
# frozen_string_literal: true
s = ''.dup
s << 'x'
p s
% ruby a.rb
"x"
```

別の方法

String.new を使う方法もある

```
% cat a.rb
# frozen_string_literal: true
s = String.new
s << 'x'
p s
% ruby a.rb
"x"
```

違い

Encoding が違う

```
% cat a.rb
# frozen_string_literal: true
p ''.dup.frozen?
p ''.dup.encoding
p String.new.frozen?
p String.new.encoding
% ruby a.rb
false
#<Encoding:UTF-8>
false
#<Encoding:ASCII-8BIT>
```

frozen になるリテラル例 (1)

'' や "" の文字列リテラルは frozen になる

```
% cat a.rb
# frozen_string_literal: true
p 'string'.frozen?
p "string\n".frozen?
% ruby a.rb
true
true
```

frozen になるリテラル例 (2)

文字列補間 (string interpolation) があっても文字列全体は frozen になる

```
% cat a.rb
# frozen_string_literal: true
p "#{1+2}".frozen?
% ruby a.rb
true
```

frozen になるリテラル例 (3)

文字リテラルも frozen

```
% cat a.rb
# frozen_string_literal: true
p ?a.frozen?
% ruby a.rb
true
```

frozen になるリテラル例 (4)

% 記法でもヒアドキュメントでも frozen になる

```
% cat a.rb
# frozen_string_literal: true
p [%q(q), %Q[#{1+2}], <<END].all?(&:frozen?)
#{1+2}
END
% ruby a.rb
true
```

frozen になるリテラル例 (5)

%w は配列自体は frozen ではないが、文字列は frozen

```
% cat a.rb
# frozen_string_literal: true
a = %w[a b]
p a.frozen?
p a.all?(&:frozen?)
% ruby a.rb
false
true
```

frozen になるリテラル例 (6)

%W も同様

```
% cat a.rb
# frozen_string_literal: true
v = "c d"
a = %W(a\ b #{v}e\sf #{})
p a.frozen?
p a.all?(&:frozen?)
% ruby a.rb
false
true
```

frozen ではないリテラル例

外部コマンド出力は frozen ではない

```
% cat a.rb
# frozen_string_literal: true
p `echo`.frozen?, %x(echo).frozen?
% ruby a.rb
false
false
```

disasm で確認 (1)

- putstring は String を作成
- freeze があると opt_str_freeze になる

```
% ruby --dump=insns -e '""'
== disasm: #<ISeq:<main>@-e>============================================
0000 trace            1                                               (   1)
0002 putstring        ""
0004 leave
% ruby --dump=insns -e '"".freeze'
== disasm: #<ISeq:<main>@-e>============================================
0000 trace            1                                               (   1)
0002 opt_str_freeze   ""
0004 leave
```

disasm で確認 (2)

- freeze は再定義チェックが入る
- putobject はオブジェクトをスタックに積むだけ

```
% ruby --enable=frozen_string_literal --dump=insns -e '"".freeze'
== disasm: #<ISeq:<main>@-e>============================================
0000 trace            1                                               (   1)
0002 opt_str_freeze   ""
0004 leave
% ruby --enable=frozen_string_literal --dump=insns -e '""'
== disasm: #<ISeq:<main>@-e>============================================
0000 trace            1                                               (   1)
0002 putobject        ""
0004 leave
```

まとめ

- ruby 2.3.0 以降を使うなら `frozen_string_literal: true` を積極的につけましょう
- `freeze` は不要になるので ruby 2.3.0 以降だけを対象にするのなら消すと良い