

Pages Classes Methods

Table of Contents

Options for Case Mapping.

Show/hide navigation

argument converters

[creates_option](#)
[option_params](#)
[tutorial](#)
[packed_data](#)
[ractor](#)
[regexp](#)
[methods](#)
[unicode_properties](#)
[COPYING](#)
[COPYING.ja](#)
[LEGAL](#)
[NEWS](#)
[README.ja](#)
[README](#)
[security](#)
[signals](#)
[standard_library](#)
[strftime_formatting](#)
[syntax](#)
[assignment](#)
[calling_methods](#)
[comments](#)
[control_expressions](#)
[exceptions](#)
[literals](#)
[methods](#)
[miscellaneous](#)
[modules_and_classes](#)
[pattern_matching](#)
[precedence](#)
[refinements](#)
[timezones](#)
[windows](#)
[yjit](#)
[yjit_hacking](#)

Case Mapping

Some string-oriented methods use case mapping.

In String:

- [String#capitalize](#)
- [String#capitalize!](#)
- [String#casecmp](#)
- [String#casecmp?](#)
- [String#downcase](#)
- [String#downcase!](#)
- [String#swapcase](#)
- [String#swapcase!](#)
- [String#upcase](#)
- [String#upcase!](#)

In Symbol:

- [Symbol#capitalize](#)
- [Symbol#casecmp](#)
- [Symbol#casecmp?](#)
- [Symbol#downcase](#)
- [Symbol#swapcase](#)
- [Symbol#upcase](#)

Default Case Mapping

By default, all of these methods use full Unicode case mapping, which is suitable for most languages. See [Section 3.13 \(Default Case Algorithms\) of the Unicode standard](#).

Non-ASCII case mapping and folding are supported for UTF-8, UTF-16BE/LE, UTF-32BE/LE, and ISO-8859-1~16 Strings/Symbols.

Context-dependent case mapping as described in [Table 3-17 \(Context Specification for Casing\) of the Unicode standard](#) is currently not supported.

In most cases, case conversions of a string have the same number of characters. There are exceptions (see also `:fold` below):

```
s = "\u00DF" # => "ß"
s.upcase     # => "SS"
s = "\u0149" # => "ñ"
s.upcase     # => "'N"
```

Case mapping may also depend on locale (see also `:turkic` below):

```
s = "\u0049"      # => "I"
s.downcase         # => "i" # Dot above.
s.downcase(:turkic) # => "ı" # No dot above.
```

Case changes may not be reversible:

```
s = 'Hello World!' # => "Hello World!"
s.downcase         # => "hello world!"
s.downcase.upcase  # => "HELLO WORLD!" # Different from original s.
```

Case changing methods may not maintain Unicode normalization. See [String#unicode_normalize](#)).

Options for Case Mapping

Except for `casecmp` and `casecmp?`, each of the case-mapping methods listed above accepts optional arguments, `*options`.

The arguments may be:

- `:ascii` only.
- `:fold` only.
- `:turkic` or `:lithuanian` or both.

The options:

- `:ascii`: ASCII-only mapping: uppercase letters ('A'..'Z') are mapped to lowercase letters ('a'..'z'); other characters are not changed

```
s = "Foo \u00D8 \u00F8 Bar" # => "Foo Ø ø Bar"
s.upcase                   # => "FOO Ø Ø BAR"
s.downcase                 # => "foo ø ø bar"
s.upcase(:ascii)          # => "FOO Ø Ø BAR"
s.downcase(:ascii)        # => "foo ø ø bar"
```

- `:turkic`: Full Unicode case mapping, adapted for the Turkic languages that distinguish dotted and dotless I, for example Turkish and Azeri.

```
s = 'Türkiye'           # => "Türkiye"
s.upcase                 # => "TÜRKIYE"
s.upcase(:turkic)       # => "TÜRKİYE" # Dot above.

s = 'TÜRKIYE'           # => "TÜRKIYE"
s.downcase               # => "türkiye"
s.downcase(:turkic)     # => "türkİye" # No dot above.
```

- `:lithuanian`: Not yet implemented.
- `:fold` (available only for [String#downcase](#), [String#downcase!](#), and [Symbol#downcase](#)): Unicode case folding, which is more far-reaching than

Unicode case mapping.

```
s = "\u00DF"      # => "ß"
s.downcase         # => "ß"
s.downcase(:fold)  # => "ss"
s.upcase          # => "SS"

s = "\uFB04"      # => "ffl"
s.downcase         # => "ffl"
s.upcase          # => "FFL"
s.downcase(:fold) # => "ffl"
```

[Validate](#)

Generated by [RDoc](#) 6.4.0.

Based on [Darkfish](#) by [Michael Granger](#).

[Ruby-doc.org](#) is provided by [James Britt](#) and [Neurogami](#).

[Hack your world. Feed your head. Live curious.](#)