

Pages Classes Methods

Search

[Case Mapping.](#)  
[Default Case Mapping.](#)  
[Options for Case Mapping.](#)

Show/hide navigation

- [NEWS-1.8.7](#)
- [NEWS-1.9.1](#)
- [NEWS-1.9.2](#)
- [NEWS-1.9.3](#)
- [NEWS-2.0.0](#)
- [NEWS-2.1.0](#)
- [NEWS-2.2.0](#)
- [NEWS-2.3.0](#)
- [NEWS-2.4.0](#)
- [NEWS-2.5.0](#)
- [NEWS-2.6.0](#)
- [NEWS-2.7.0](#)
- [NEWS-3.0.0](#)
- [NEWS-3.1.0](#)
- [NEWS-3.2.0](#)
- [bsearch](#)
- [bug triaging](#)
- [case mapping](#)
- [character selectors](#)
- [command injection](#)
- [contributing](#)
- [building\\_ruby](#)
- [documentation guide](#)
- [glossary](#)
- [making changes to ruby](#)
- [making changes to stdlibs](#)
- [reporting issues](#)
- [testing ruby](#)
- [dig methods](#)
- [distribution](#)
- [dtrace probes](#)
- [encodings](#)
- [extension.ja](#)
- [extension](#)
- [fiber](#)
- [format specifications](#)
- [globals](#)
- [implicit conversion](#)
- [keywords](#)
- [maintainers](#)
- [marshal](#)
- [memory view](#)
- [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.](#)