

[Home](#)

[Pages](#) [Classes](#) [Methods](#)

Search

Table of Contents

[Character Selectors](#)
[Character Selector](#)
[Multiple Character Selectors](#)

Show/hide navigation

Pages

[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](#)

Character Selectors

Character Selector

A *character selector* is a string argument accepted by certain Ruby methods. Each of these instance methods accepts one or more character selectors:

- [`String#tr`](#)(selector, replacements): returns a new string.
- [`String#tr!`](#)(selector, replacements): returns `self` or `nil`.
- [`String#tr_s`](#)(selector, replacements): returns a new string.
- [`String#tr_s!`](#)(selector, replacements): returns `self` or `nil`.
- [`String#count\(*selectors\)`](#): returns the count of the specified characters.
- [`String#delete\(*selectors\)`](#): returns a new string.
- [`String#delete!\(*selectors\)`](#): returns `self` or `nil`.
- [`String#squeeze\(*selectors\)`](#): returns a new string.
- [`String#squeeze!\(*selectors\)`](#): returns `self` or `nil`.

A character selector identifies zero or more characters in `self` that are to be operands for the method.

In this section, we illustrate using method [`String#delete\(selector\)`](#), which deletes the selected characters.

In the simplest case, the characters selected are exactly those contained in the selector itself:

```
'abracadabra'.delete('a')      # => "brcdbr"
'abracadabra'.delete('ab')    # => "rcdr"
'abracadabra'.delete('abc')   # => "rdr"
'0123456789'.delete('258')    # => "0134679"
'!@#$%&*()_+'.delete('+&#')  # => "!@#$%*()*_"
'tecτ'.delete('τ')            # => "ec"
'こんにちは'.delete('に')     # => "こんちは"
```

Note that order and repetitions do not matter:

```
'abracadabra'.delete('dcab')  # => "rr"
'abracadabra'.delete('aaaa')  # => "brcdbr"
```

In a character selector, these three characters get special treatment:

- A leading caret (`'^'`) functions as a “not” operator for the characters to its right:

```
'abracadabra'.delete('^bc')   # => "bcb"
'0123456789'.delete('^852')   # => "258"
```

- A hyphen (' - ') between two other characters defines a range of characters instead of a plain string of characters:

```
'abracadabra'.delete('a-d') # => "rr"
'0123456789'.delete('4-7') # => "012389"
'!@#%&*()_+'.delete(' - /') # => "@^_"

# May contain more than one range.
'abracadabra'.delete('a-cq-t') # => "d"

# Ranges may be mixed with plain characters.
'0123456789'.delete('67-950-23') # => "4"

# Ranges may be mixed with negations.
'abracadabra'.delete('^a-c') # => "abacaaba"
```

- A backslash (' \ ') acts as an escape for a caret, a hyphen, or another backslash:

```
'abracadabra^'.delete('\^bc') # => "araadara"
'abracadabra-'.delete('a\ -d') # => "brcbbr"
"hello\r\nworld".delete("\r") # => "hello\nworld"
"hello\r\nworld".delete("\ \r") # => "hello\r\nwold"
"hello\r\nworld".delete("\ \ \r") # => "hello\nworld"
```

Multiple Character Selectors

These instance methods accept multiple character selectors:

- [String#count\(*selectors\)](#): returns the count of the specified characters.
- [String#delete\(*selectors\)](#): returns a new string.
- [String#delete!\(*selectors\)](#): returns `self` or `nil`.
- [String#squeeze\(*selectors\)](#): returns a new string.
- [String#squeeze!\(*selectors\)](#): returns `self` or `nil`.

In effect, the given selectors are formed into a single selector consisting of only those characters common to *all* of the given selectors.

All forms of selectors may be used, including negations, ranges, and escapes.

Each of these pairs of method calls is equivalent:

```
s.delete('abcde', 'dcbfg')
s.delete('bcd')

s.delete('^abc', '^def')
s.delete('^abcdef')

s.delete('a-e', 'c-g')
s.delete('cde')
```

[Validate](#)

Generated by [RDoc](#) 6.4.0.

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

[Ruby-doc.org](#) is a service of [James Britt](#) and [Neurogami](#), purveyors of fine [dance noise](#)