### Anchors

- `^` Start of string, or start of line in multi-line pattern
- `\A` Start of string
- `$` End of string, or end of line in multi-line pattern
- `\Z` End of string
- `\b` Word boundary
- `\B` Not word boundary
- `<` Start of word
- `>` End of word

### Character Classes

- `\c` Control character
- `\s` White space
- `\S` Not white space
- `\d` Digit
- `\D` Not digit
- `\w` Word
- `\W` Not word
- `\x` Hexadecimal digit
- `\O` Octal digit

### POSIX

- `[upper:]` Upper case letters
- `[lower:]` Lower case letters
- `[alpha:]` All letters
- `[alnum:]` Digits and letters
- `[digit:]` Digits
- `[xdigit:]` Hexadecimal digits
- `[punct:]` Punctuation
- `[blank:]` Space and tab
- `[space:]` Blank characters
- `[cntrl:]` Control characters
- `[graph:]` Printed characters
- `[print:]` Printed characters and spaces
- `[word:]` Digits, letters and underscore

### Assertions

- `?=` Lookahead assertion
- `?!` Negative lookahead
- `?<=` Lookbehind assertion
- `?!=` or `?!<` Negative lookbehind
- `?>` Once-only Subexpression
- `?()` Condition [if then]
- `?()[` Condition [if then else]
- `?#` Comment

### Quantifiers

- `*` 0 or more
- `+` 1 or more
- `?` 0 or 1
- `{3}` Exactly 3
- `{3,}` 3 or more
- `{3, 5}` 3, 4 or 5

Add a ? to a quantifier to make it ungreedy.

### Escape Sequences

- `\` Escape following character
- `\Q` Begin literal sequence
- `\E` End literal sequence

"Escaping" is a way of treating characters which have a special meaning in regular expressions literally, rather than as special characters.

### Common Metacharacters

- `^` Start of string
- `{` Start group
- `}` End group
- `+` One or more
- `?` Zero or one
- `<` Less than
- `>` Greater than

The escape character is usually `\`.

### Pattern Modifiers

- `g` Global match
- `i` Case-insensitive
- `m` Multiple lines
- `s` Treat string as single line
- `x` Allow comments and whitespace in pattern
- `e` Evaluate replacement
- `U` Ungreedy pattern
- `*` PCRE modifier

### Groups and Ranges

- `. ` Any character except new line (\n)
- `(a|b)` a or b
- `(\d)` Digit
- `(\w)` Digits and letters
- `(\S)` Not white space
- `(\D)` Not digit
- `(\s)` White space
- `(\b)` Word boundary
- `(\B)` Not word boundary
- `(\A)` Start of string
- `(\Z)` End of string
- `(\{3\})` Exactly 3
- `(\{3,\})` 3 or more
- `(\{3, 5\})` 3, 4 or 5

Ranges are inclusive.

### String Replacement

- `$n` nth non-passive group
- `$2` "xyz" in `/\(abc\(xyz\)\)/` $2
- `$1` "xyz" in `/\(\?::abc\)\(xyz\)\)/` $1
- `$` Before matched string
- `$` After matched string
- `$+` Last matched string
- `$&` Entire matched string

Some regex implementations use `\` instead of `$`.

---

By Dave Child (DaveChild)
cheatography.com/davechild/
aloneonahill.com

Published 19th October, 2011.
Page 1 of 1.

Sponsored by Readable.com
Measure your website readability!
https://readable.com