

Anchors

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

Character Classes

<code>\c</code>	Control character
<code>\s</code>	White space
<code>\S</code>	Not white space
<code>\d</code>	Digit
<code>\D</code>	Not digit
<code>\w</code>	Word
<code>\W</code>	Not word
<code>\x</code>	Hexadecimal digit
<code>\O</code>	Octal digit

POSIX

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

Assertions

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

Quantifiers

<code>*</code>	0 or more	<code>{3}</code>	Exactly 3
<code>+</code>	1 or more	<code>{3,}</code>	3 or more
<code>?</code>	0 or 1	<code>{3,5}</code>	3, 4 or 5

Add a `?` to a quantifier to make it ungreedy.

Escape Sequences

<code>\</code>	Escape following character
<code>\Q</code>	Begin literal sequence
<code>\E</code>	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

<code>^</code>	<code>[</code>	<code>.</code>	<code>\$</code>
<code>{</code>	<code>*</code>	<code>(</code>	<code>\</code>
<code>+</code>	<code>)</code>	<code> </code>	<code>?</code>
<code><</code>	<code>></code>		

The escape character is usually `\`

Special Characters

<code>\n</code>	New line
<code>\r</code>	Carriage return
<code>\t</code>	Tab
<code>\v</code>	Vertical tab
<code>\f</code>	Form feed
<code>\xxx</code>	Octal character xxx
<code>\xhh</code>	Hex character hh

Groups and Ranges

<code>.</code>	Any character except new line (<code>\n</code>)
<code>(a b)</code>	a or b
<code>(...)</code>	Group
<code>(?:...)</code>	Passive (non-capturing) group
<code>[abc]</code>	Range (a or b or c)
<code>[^abc]</code>	Not (a or b or c)
<code>[a-q]</code>	Lower case letter from a to q
<code>[A-Q]</code>	Upper case letter from A to Q
<code>[0-7]</code>	Digit from 0 to 7
<code>\x</code>	Group/subpattern number "x"

Ranges are inclusive.

Pattern Modifiers

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

String Replacement

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

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

