

Regular Expression Quick Reference v1.00

Online RegEx Resources: www.gmckinney.info

Literal Characters	
<code>\f</code>	Form feed
<code>\n</code>	Newline (Use <code>\p</code> in UltraEdit for platform independent line end)
<code>\r</code>	Carriage return
<code>\t</code>	Tab
<code>\v</code>	Vertical tab
<code>\a</code>	Alarm (beep)
<code>\e</code>	Escape
<code>\xxx</code>	The ASCII character specified by the octal number xxx
<code>\xnn</code>	The ASCII character specified by the hexadecimal number nn
<code>\cX</code>	The control character ^X. For example, <code>\cI</code> is equivalent to <code>\t</code> and <code>\cJ</code> is equivalent to <code>\n</code>

Character Classes	
<code>[...]</code>	Any one character between the brackets.
<code>[^...]</code>	Any one character not between the brackets.
<code>.</code>	Any character except newline. Equivalent to <code>[^\n]</code>
<code>\w</code>	Any word character. Equivalent to <code>[a-zA-Z0-9_]</code> and <code>[[:alnum:]]</code>
<code>\W</code>	Any non-word character. Equivalent to <code>[^a-zA-Z0-9_]</code> and <code>[^[:alnum:]]</code>
<code>\s</code>	Any whitespace character. Equivalent to <code>[\t\n\r\f\v]</code> and <code>[[:space:]]</code>
<code>\S</code>	Any non-whitespace. Equivalent to <code>[^\t\n\r\f\v]</code> and <code>[^[:space:]]</code> Note: <code>lw != \S</code>
<code>\d</code>	Any digit. Equivalent to <code>[0-9]</code> and <code>[[:digit:]]</code>
<code>\D</code>	Any character other than a digit. Equivalent to <code>[^0-9]</code> and <code>[^[:digit:]]</code>
<code>[\b]</code>	A literal backspace (special case)
<code>[[:class:]]</code>	
	alnum alpha ascii blank cntrl digit graph
	lower print punct space upper xdigit

Replacement	
<code>\</code>	Turn off the special meaning of the following character.
<code>\n</code>	Restore the text matched by the nth pattern previously saved by <code>\(</code> and <code>\)</code> . n is a number from 1 to 9, with 1 starting on the left.
<code>&</code>	Reuse the text matched by the search pattern as part of the replacement pattern.
<code>~</code>	Reuse the previous replacement pattern in the current replacement pattern. Must be the only character in the replacement pattern. (ex and vi).
<code>%</code>	Reuse the previous replacement pattern in the current replacement pattern. Must be the only character in the replacement pattern. (ed).
<code>\u</code>	Convert first character of replacement pattern to uppercase.
<code>\U</code>	Convert entire replacement pattern to uppercase.
<code>\l</code>	Convert first character of replacement pattern to lowercase.
<code>\L</code>	Convert entire replacement pattern to lowercase.

Repetition	
<code>{n,m}</code>	Match the previous item at least n times but no more than m times.
<code>{n,}</code>	Match the previous item n or more times.
<code>{n}</code>	Match exactly n occurrences of the previous item.
<code>?</code>	Match zero or one occurrences of the previous item. Equivalent to <code>{0,1}</code>
<code>+</code>	Match one or more occurrences of the previous item. Equivalent to <code>{1,}</code>
<code>*</code>	Match zero or more occurrences of the previous item. Equivalent to <code>{0,}</code>
<code>{ }?</code>	Non-greedy match - will not include the next match's characters.
<code>??</code>	Non-greedy match.
<code>+?</code>	Non-greedy match.
<code>*?</code>	Non-greedy match. E.g. <code>^(.*?)\s*\$</code> the grouped expression will not include trailing spaces.

Options	
<code>g</code>	Perform a global match. That is, find all matches rather than stopping after the first match.
<code>i</code>	Do case-insensitive pattern matching.
<code>m</code>	Treat string as multiple lines (^ and \$ match internal \n).
<code>s</code>	Treat string as single line (^ and \$ ignore \n, but <code>.</code> matches \n).
<code>x</code>	Extend your pattern's legibility with whitespace and comments.

Extended Regular Expression	
<code>(?#...)</code>	Comment, "... is ignored.
<code>(?:...)</code>	Matches but doesn't return "..."
<code>(?=...)</code>	Matches if expression would match "... next
<code>(?!...)</code>	Matches if expression wouldn't match "... next
<code>(?imsx)</code>	Change matching rules (see options) midway through an expression.

Grouping	
<code>(...)</code>	Grouping. Group several items into a single unit that can be used with <code>*</code> , <code>+</code> , <code>?</code> , <code> </code> , and so on, and remember the characters that match this group for use with later references.
<code> </code>	Alternation. Match either the subexpressions to the left or the subexpression to the right.
<code>\n</code>	Match the same characters that were matched when group number n was first matched. Groups are subexpressions within (possibly nested) parentheses.

Anchors	
<code>^</code>	Match the beginning of the string, and, in multiline searches, the beginning of a line.
<code>\$</code>	Match the end of the string, and, in multiline searches, the end of a line.
<code>\b</code>	Match a word boundary. That is, match the position between a <code>\w</code> character and a <code>\W</code> character. (Note, however, that <code>[b]</code> matches backspace.)
<code>\B</code>	Match a position that is not a word boundary.