Regular Expression Quick Reference v1.2

Literal Characters

`\f`	Form feed
`\n`	Newline
`\r`	Carriage return
`\t`	Tab
`\v`	Vertical tab
`\a`	Alarm (beep)
`\e`	Escape
`\xHH`	The ASCII character specified by the two digit hexadecimal code. For octal use `\OOO` except JS
`\x{HHHH}`	PHP: ASCII character represented by a four digit hexadecimal code. Javascript uses `\uHHHH`
`\cX`	The control character `^X.` For example, `\cI` is equivalent to `\t` and `\cJ` is equivalent to `\n`

Character Classes

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

Replacement

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

Repetition

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

Options

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

Extended Regular Expression

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

Grouping

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

Anchors

`^`	Match the beginning of the string, and, in multiline searches (/m), the beginning of a line. PHP: Use \A to match beginning of string in all line matching modes.
`$`	Match the end of the string, and, in multiline searches (/m), the end of a line. PHP: Use \z and \Z to match the end of a string or end of text respectively.
`\b`	Match a word boundary. That is, match the position between a \w character and a \W character. (Note, however, that [\b] matches backspace.)
`\B`	Match a position that is not a word boundary.