

# Perl Quick Reference Card

version 0.01 – editor: John Bokma – freelance programmer  
**DRAFT VERSION**, check: <http://johnbokma.com/perl/>

## Backslashed Character Escapes 61

<code>\n</code>	Newline (usually LF)	<code>\e</code>	ESC character
<code>\r</code>	Carriage return (usually CR)	<code>\033</code>	ESC in octal
<code>\t</code>	Horizontal tab	<code>\x7f</code>	DEL in hexadecimal
<code>\f</code>	Form feed	<code>\cC</code>	Control-C
<code>\b</code>	Backspace	<code>\x{263a}</code>	Unicode (smiley)
<code>\a</code>	Alert (bell)	<code>\N{NAME}</code>	Named character

## Translation Escapes 61

<code>\u</code>	Force next character to uppercase (“titlecase” in Unicode).
<code>\l</code>	Force next character to lowercase.
<code>\U</code>	Force all following characters to uppercase
<code>\L</code>	Force all following characters to lowercase
<code>\Q</code>	Backslash all following nonalphanumeric characters.
<code>\E</code>	End <code>\U</code> , <code>\L</code> , or <code>\Q</code> .

## Quote Constructs 63

Customary	Generic	Meaning	Interpolates
<code>' '</code>	<code>q//</code>	Literal string	No
<code>""</code>	<code>qq//</code>	Literal string	Yes
<code>` `</code>	<code>qx//</code>	Command execution	Yes
<code>()</code>	<code>qw//</code>	Word list	No
<code>//</code>	<code>m//</code>	Pattern match	Yes
<code>s///</code>	<code>s///</code>	Pattern substitution	Yes
<code>y///</code>	<code>tr///</code>	Character translation	No
<code>""</code>	<code>qr//</code>	Regular expression	Yes

**Note:** no interpolation is done if you use single quotes for delimiters.

## Operator Precedence 87

Associativity	Arity	Precedence Class
None	0	Terms, and list operators (leftward)
Left	2	<code>-&gt;</code>
None	1	<code>++ --</code>
Right	2	<code>**</code>
Right	1	<code>! ~ &gt;</code> and unary <code>+</code> and unary <code>-</code>
Left	2	<code>== !~</code>
Left	2	<code>* / % x</code>
Left	2	<code>+ - .</code>
Left	2	<code>&lt;&lt; &gt;&gt;</code>
Right	0,1	Named unary operators
None	2	<code>&lt; &gt; &lt;= &gt;= lt gt le ge</code>
None	2	<code>== != &lt;=&gt; eq ne cmp</code>
Left	2	<code>&amp;</code>
Left	2	<code>  ^</code>
Left	2	<code>&amp;&amp;</code>
Left	2	<code>  </code>
None	2	<code>.. ...</code>

## Operator Precedence (continued) 87

Associativity	Arity	Precedence Class
Right	3	<code>?:</code>
Right	2	<code>= += -= *=</code> and so on
Left	2	<code>, =&gt;</code>
Right	0+	List operators (rightward)
Right	1	<code>not</code>
Left	2	<code>and</code>
Left	2	<code>or xor</code>

## File Test Operators 98

<code>-r</code>	File is readable by effective UID/GID.
<code>-w</code>	File is writable by effective UID/GID.
<code>-x</code>	File is executable by effective UID/GID.
<code>-o</code>	File is owned by effective UID/GID.
<code>-R</code>	File is readable by real UID/GID.
<code>-W</code>	File is writable by real UID/GID.
<code>-X</code>	File is executable by real UID/GID.
<code>-O</code>	File is owned by real UID/GID.
<code>-e</code>	File exists.
<code>-z</code>	File has zero size
<code>-s</code>	File has nonzero size (returns size).
<code>-f</code>	File is a plain file.
<code>-d</code>	File is a directory.
<code>-l</code>	File is a symbolic link.
<code>-p</code>	File is a named pipe (FIFO).
<code>-S</code>	File is a socket.
<code>-b</code>	File is a block special file.
<code>-c</code>	File is a character special file.
<code>-t</code>	Filehandle is open to a tty.
<code>-u</code>	File has setuid bit set.
<code>-g</code>	File has setgid bit set.
<code>-k</code>	File has sticky bit set.
<code>-T</code>	File is a text file.
<code>-B</code>	File is a binary file (opposite of <code>-T</code> ).
<code>-M</code>	Age of file (at startup) in (fractional) days since modification.
<code>-A</code>	Age of file (at startup) in (fractional) days since last access.
<code>-C</code>	Age of file (at startup) in (fractional) days since inode change.

## Pattern Modifiers 147

<code>/i</code>	Ignore alphabetic case distinctions (case insensitive).
<code>/s</code>	Let <code>.</code> match newline and ignore deprecated <code>\$*</code> variable.
<code>/m</code>	Let <code>^</code> and <code>\$</code> match next embedded <code>\n</code> .
<code>/x</code>	Ignore (most) whitespace and permit comments in pattern.
<code>/o</code>	Compile pattern only once.

## Additional m// Modifiers 150

<code>/g</code>	Globally find all matches.
<code>/cg</code>	Allow continued search after failed <code>/g</code> match.

## Additional s/// Modifiers 153

<code>/g</code>	Replace globally, that is, all occurrences.
<code>/e</code>	Evaluate the right side as an expression.

## tr/// Modifiers 156

<code>/c</code>	Complement SEARCHLIST.
<code>/d</code>	Delete found but unreplaced characters.
<code>/s</code>	Squash duplicate replaced characters.

## General Regex Metacharacters 159

Symbol	Atomic	Meaning
<code>\...</code>	Varies	De-meta next nonalphanumeric character, meta next alphanumeric character (maybe).
<code>... ...</code>	No	Alternation (match one or the other).
<code>(...)</code>	Yes	Grouping (treat as a unit).
<code>[...]</code>	Yes	Character class (match one character from a set).
<code>^</code>	No	True at beginning of string (or after a newline, maybe).
<code>.</code>	Yes	Match one character (except newline, normally).
<code>\$</code>	No	True at end of string (or before any newline, maybe).

## Regex Quantifiers 159-160

Quantifier	Atomic	Meaning
<code>*</code>	No	Match 0 or more times (maximal).
<code>+</code>	No	Match 1 or more times (maximal).
<code>?</code>	No	Match 0 or 1 time (maximal).
<code>{COUNT}</code>	No	Match exactly <i>COUNT</i> times.
<code>{MIN, }</code>	No	Match at least <i>MIN</i> times (maximal).
<code>{MIN, MAX}</code>	No	Match at least <i>MIN</i> but not more than <i>MAX</i> times (maximal).
<code>*?</code>	No	Match 0 or more times (minimal).
<code>+?</code>	No	Match 1 or more times (minimal).
<code>??</code>	No	Match 0 or 1 time (minimal).
<code>{MIN, }?</code>	No	Match at least <i>MIN</i> times (minimal).
<code>{MIN, MAX}?</code>	No	Match at least <i>MIN</i> but not more than <i>MAX</i> times (minimal).

## Extended Regex Sequences 160

Extension	Atomic	Meaning
<code>(?#...)</code>	No	Comment, discard.
<code>(?:...)</code>	Yes	Cluster-only parentheses, no capturing.
<code>(?imsx-imsx)</code>	No	Enable/disable pattern modifiers.
<code>(?imsx-imsx:...)</code>	Yes	Cluster-only parentheses plus modifiers.
<code>(?=...)</code>	No	True if lookahead assertion succeeds.
<code>(?!...)</code>	No	True if lookahead assertion fails.
<code>(?&lt;=...)</code>	No	True if lookbehind assertion succeeds.
<code>(?&lt;!...)</code>	No	True if lookbehind assertion fails.
<code>(?&gt;...)</code>	Yes	Match nonbacktracking subpattern.
<code>(?{...})</code>	No	Execute embedded Perl code.
<code>(??{...})</code>	Yes	Match regex from embedded Perl code.
<code>(?(...) ... ...)</code>	Yes	Match with if-then-else pattern.
<code>(?(...) ...)</code>	Yes	Match with if-then pattern.