# Perl Quick Reference Card

version 0.02 – editor: John Bokma – freelance programmer
**DRAFT VERSION**, check: http://johnbokma.com/perl/

## Backslashed Character Escapes 61

| | | | |
|---|---|---|---|
| `\n` | Newline (usually LF) | `\0` | Null character (NUL) |
| `\r` | Carriage return (usually CR) | `\033` | ESC in octal |
| `\t` | Horizontal tab (HT) | `\x7f` | DEL in hexadecimal |
| `\f` | Form feed (FF) | `\cC` | Control-C |
| `\b` | Backspace (BS) | `\x{263a}` | Unicode, ☺(smiley) |
| `\a` | Alert (BEL) | `\N{NAME}` | Named character |
| `\e` | Escape (ESC) | | |

## Translation Escapes 61

`\u` Force next character to uppercase ("titlecase" in Unicode).
`\l` Force next character to lowercase.
`\U` Force all following characters to uppercase
`\L` Force all following characters to lowercase
`\Q` Backslash all following non-"word" characters (quotemeta)
`\E` End `\U`, `\L`, or `\Q`.

## Quote Constructs 63

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

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

## Operator Precedence 87

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

## Operator Precedence (continued) 87

| Associativiy | Arity | Precedence Class |
|---|---|---|
| Right | 3 | `?:` |
| Right | 2 | `= += -= *=` and so on |
| Left | 2 | `, =>` |
| Right | 0+ | List operators (rightward) |
| Right | 1 | `not` |
| Left | 2 | `and` |
| Left | 2 | `or xor` |

## File Test Operators 98

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

## Pattern Modifiers 147

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

## Additional m// Modifiers 150

`/g` Globally find all matches.
`/cg` Allow continued search after failed `/g` match.

## Additional s/// Modifiers 153

`/g` Replace globally, that is, all occurences.
`/e` Evaluate the right side as an expression.

## tr/// Modifiers 156

`/c` Complement SEARCHLIST.
`/d` Delete found but unreplaced characters.
`/s` Squash duplicate replaced characters.

## General Regex Metacharacters 159

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

## Regex Quantifiers 159-160

| Quantifier | Atomic | Meaning |
|---|---|---|
| `*` | No | Match 0 or more times (maximal). |
| `+` | No | Match 1 or more times (maximal). |
| `?` | No | Match 0 or 1 time (maximal). |
| `{COUNT}` | No | Match exactly *`COUNT`* times. |
| `{MIN,}` | No | Match at least *`MIN`* times (maximal). |
| `{MIN,MAX}` | No | Match at least *`MIN`* but not more than *`MAX`* times (maximal). |
| `*?` | No | Match 0 or more times (minimal). |
| `+?` | No | Match 1 or more times (minimal). |
| `??` | No | Match 0 or 1 time (minimal). |
| `{MIN,}?` | No | Match at least *`MIN`* times (minimal). |
| `{MIN,MAX}?` | No | Match at least *`MIN`* but not more than *`MAX`* times (minimal). |

## Extended Regex Sequences 160

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

## Alphanumeric Regex Metasymbols 161-162

| Symbol | Atomic | Meaning |
|---|---|---|
| \0 | Yes | Match the null character (ASCII NUL). |
| \NNN | Yes | Match the character given in octal, up to \377. |
| \n | Yes | Match *n*th previously captured string (decimal). |
| \a | Yes | Match the alarm character (BEL). |
| \A | No | True at the beginning of a string. |
| \b | Yes | Match the backspace character (BS). |
| \b | No | True at a word boundary. |
| \B | No | True when not at a word boundary. |
| \cX | Yes | Match the control character Ctrl-X (\cZ). |
| \C | Yes | Match one byte (C char) even in utf8 (dangerous). |
| \d | Yes | Match any digit character. |
| \D | Yes | Match any non-digit character. |
| \e | Yes | Match the escape character (ASCII ESC, not \ ). |
| \E | — | End case (\L, \U) or quotemeta (\Q) translation. |
| \f | Yes | Match the form feed character (FF). |
| \G | No | True at end-of-match position of prior m//g. |
| \l | — | Lowercase the next character only. |
| \L | — | Lowercase till \E. |
| \n | Yes | Match the newline character (usually NL, but CR on Macs). |
| \N{*NAME*} | Yes | Match the named char (\N{greek:Sigma}). |
| \p{*PROP*} | Yes | Match any character with named property. |
| \P{*PROP*} | Yes | Match any character without the named property. |
| \Q | — | Quote (de-meta) metacharacters till \E. |
| \r | Yes | Match the return character (usually CR, but NL on Macs). |
| \s | Yes | Match any whitespace character. |
| \S | Yes | Match any nonwhitespace character. |
| \t | Yes | Match the tab character (HT). |
| \u | — | Titlecase next character only. |
| \U | — | Uppercase (not titlecase) till \E. |
| \w | Yes | Match any "word" character (alphanum plus "_"). |
| \W | Yes | Match any nonword character. |
| \x*HEX* | Yes | Match the character given one or two hex digits. |
| \x{*abcd*} | Yes | Match the character given in hexadecimal. |
| \X | Yes | Match Unicode "combining character sequence" string. |
| \z | No | True at end of string only. |
| \Z | No | True at end of string or before optional newline. |

## Classic Character Classes 167

| Symbol | Meaning | As Bytes | As utf8 |
|---|---|---|---|
| \d | Digit | [0-9] | \p{IsDigit} |
| \D | Nondigit | [^0-9] | \P{IsDigit} |
| \s | White | [ \t\n\r\f] | \p{IsSpace} |
| \S | Nonwhitespace | [^ \t\n\r\f] | \P{IsSpace} |
| \w | Word character | [a-zA-Z0-9_] | \p{IsWord} |
| \W | Non-(word character) | [^a-zA-Z0-9_] | \P{IsWord} |

## Composite Unicode Properties 168-169

| Property | Equivalent |
|---|---|
| IsASCII | [\x00-\x7f] |
| IsAlnum | [\p{IsLl}\p{IsLu}\p{IsLt}\p{IsLo}\p{IsNd} |
| IsAlpha | [\p{IsLl}\p{IsLu}\p{IsLt}\p{IsLo} |
| IsCntrl | \p{IsC} |
| IsDigit | \p{IsNd} |
| IsGraph | [^\pC\p{IsSpace}] |
| IsLower | \p{IsLl} |
| IsPrint | \P{IsC} |
| IsPunct | \p{IsP} |
| IsSpace | [\t\n\f\r\p{IsZ}] |
| IsUpper | [\p{IsLu}\p{IsLt}] |
| IsWord | [_\p{IsLl}\p{IsLu}\p{IsLt}\p{IsLo}\p{IsNd}] |
| IsXDigit | [0-9a-fA-F] |

Perl also provides the following composites:

| Property | Meaning | Normative |
|---|---|---|
| IsC | Crazy control characters and such | Yes |
| IsL | Letters | Partly |
| IsM | Marks | Yes |
| IsN | Numbers | Yes |
| IsP | Punctuation | No |
| IsS | Symbols | No |
| IsZ | Separators (Zeparators?) | Yes |

## POSIX-Style Character Classes 174-175

| Class | Meaning |
|---|---|
| alnum | Any alphanumeric, that is an alpha or a digit. |
| alpha | Any letter. (That's a lot more letters than you think, unless you're thinking Unicode, in which case it's still a lot.) |
| ascii | Any character with an ordinal value between 0 and 127. |
| cntrl | Any control character. Usually characters that don't produce output as such, but instead control the terminal somehow; for example, newline, form feed, and backspace. |
| digit | A character representing a decimal digit, such as 0 to 9. (Includes other characters under Unicode.) Equivalent to \d. |
| graph | Any alphanumeric or punctuation character. |
| lower | A lowercase letter. |
| print | Any alphanumeric or punctuation character or space. |
| punct | Any punctuation character. |
| space | Any space character. Includes tab, newline, form feed, and carriage return (and a lot more under Unicode.) Equivalent to \s. |
| upper | Any uppercase (or titlecase) letter. |
| word | Any identifier character, either an alnum or underline. |
| xdigit | Any hexadecimal digit. Equivalent to [0-9a-fA-F]. |

You can negate the POSIX character classes by prefixing the class name with a ^ following the [:. (This is a Perl extension.)