Matching single characters

Action	**BRE**	**ERE** *(plus features only in Tcl)*	**EMACS**	**VIM**	**PCRE** *(plus features only in Perl 5)*	**PSIX**
Match the specified char once	*Any single character* ***except:*** . \ ^ $ [*	*Any single character* ***except:*** . \ ^ $ [* ( ) | + ? {	*Any single character* ***except:*** . \ ^ $ [* ( ) | + ? {	*Any single character* ***except:*** . \ ^ $ * ~	*Any single alphanumeric or any of:* ~ ! % & _ ` - = @ : ; " ' < > , /	*Any single unescaped alphanumeric or any single escaped non-alphanumeric*
Match any one char	.	.	▨	_.	\p{Any} (?s:.)	<:Any> .
Match any one char except an end-of-line marker	▨	▨	.	.	. \N	\N
Match any one of the enclosed characters	[*XXX*]	[*XXX*]	[*XXX*]	[*XXX*]	[*XXX*] *(?[XXX])*	<[*XXX*]>
Match any one character not specified within	[^*XXX*]	[^*XXX*]	[^*XXX*]	[^*XXX*]	[^*XXX*] *(?[!{XXX}])*	<-[*XXX*]>
Match one alarm (beep, bell) char	▨	\a	\a	▨	\a	\a
Match one escape char	▨			▨	\e	\e
Match one formfeed char	▨	\f	\f	▨	\f	\f
Match one linefeed/newline char	▨	\n	\n	▨	\n	\c[NL]
Match one carriage return char	▨	\r	\r	▨	\r	\r
Match one tab char	▨	\t	\t	▨	\t	\t
Match one char (specified as octal)	▨	*\o77* *\o777*		▨	\777 \o{*7777*}	\o*77* \o[*7777*]
Match one char (specified as hex)	▨	\x*FF* \x{*FFFF*}	\x*FF* \x{*FFFF*}	▨	\x*FF* \x{*FFFF*} \u*FFFF*	\x*FF* \x[*FFFF*]

Match one named Unicode char	▨	*\uFFFF*
		\UFFFFFFFF
				▨	*\N{NAME}*
					\N{U+FFFF}
						\c[*NAME*]

Match one control char (specified by single-letter name)	▨	▨	▨	▨	\c*A*	\c*A*
Match one digit	▨	\d	\d	\d	\d	\d

Matching single characters

Action	**BRE**	**ERE** *(plus features only in Tcl)*	**EMACS**	**VIM**	**PCRE** *(plus features only in Perl 5)*	**PSIX**
Match one non-digit		`\D`	`\D`	`\D`	`\D`	`\D`

Match one identifier character		`\w`	`\w`
			`\sw`
			`\s_`
				`\w`	`\w`	`\w`

Match one character that is not an identifier character		`\W`	`\W`
			`\Sw`
			`\S_`
				`\W`	`\W`	`\W`

Match one whitespace character		`\s`	`\s-`	`\s`	`\s`	`\s`
Match one character that is not whitespace		`\S`	`\S-`	`\S`	`\S`	`\S`
Match one horizontal whitespace					`\h`	`\h`
Match one character that is not horizontal whitespace					`\H`	`\H`
Match one vertical whitespace character					`\v`	`\v`
Match one character that is not vertical whitespace					`\V`	`\V`
Match one Unicode newline character (or sequence)					`\R`	`\n`
Match one native char					`\C`	`[:bytes .]`
Match one Unicode extended grapheme					`\X`	`<.>`
Match one character with the specified Unicode property					`\p`*X* `\p{`*NAME*`}`	`<:`*NAME*`>`
Match one character without the specified Unicode property					`\P`*X* `\P{`*NAME*`}`	`<-:`*NAME*`>`
Escape the following metacharacter	`\`	`\`	`\`	`\`	`\`	`\`

Escape the enclosed metacharacters						`'`*METAS*`'`
						`"`*METAS*`"`

Matching multiple characters						
Action	**BRE**	**ERE** *(plus features only in Tcl)*	**EMACS**	**VIM**	**PCRE** *(plus features only in Perl 5)*	**PSIX**

Match preceding atom zero or more times (maximally)	`A*`	`A*`	`A*`	`A*`
				`A\{}`
					`A*`	`A*`

Match preceding atom zero or more times (minimally)		`A*?`	`A*?`	`A\{-}`	`A*?`	`A*?`

Match preceding atom zero or more times (possessively)					`A*+`	`A*+`
						`A*:`

Match preceding atom 1 or more times (maximally)		`A+`	`A+`	`A\+`	`A+`	`A+`
Match preceding atom 1 or more times (minimally)		`A+?`	`A+?`	`A\{-1,}`	`A+?`	`A+?`

Match preceding atom 1 or more times (possessively)					`A++`	`A++`
						`A+:`

Match preceding atom zero or 1 times (maximally)		`A?`	`A?`	`A\?`
				`A\=`
					`A?`	`A?`

Match preceding atom zero or 1 times (minimally)		`A??`	`A??`	`A\{-0,1}`	`A??`	`A??`

Match preceding atom zero or 1 times (possessively)					`A?+`	`A?+`
						`A?:`

Match preceding atom *M* times (maximally)	`A\{M\}`	`A{M}`	`A\{M\}`	`A\{M}`	`A{M}`	`A ** M`
Match preceding atom *M* or more (maximally)	`A\{M,\}`	`A{M,}`	`A\{M,\}`	`A\{M,}`	`A{M,}`	`A ** M..*`
Match preceding atom *M* to *N* times (maximally)	`A\{M,N\}`	`A{M,N}`	`A\{M,N\}`	`A\{M,N}`	`A{M,N}`	`A ** M..N`
Match preceding atom *M* times (minimally)		`A{M}?`		`A\{-M}`	`A{M}?`	`A **? M`

Matching multiple characters

Action	**BRE**	**ERE** *(plus features only in Tcl)*	**EMACS**	**VIM**	**PCRE** *(plus features only in Perl 5)*	**PSIX**
Match preceding atom *M* or more (minimally)		`A{M,}?`		`A\{-M,}`	`A{M,}?`	`A **? M..*`
Match preceding atom *M* to *N* times (minimally)		`A{M,N}?`		`A\{-M,N}`	`A{M,N}?`	`A **? M..N`
Match preceding atom *M* times (possessively)					`A{M}+`	`A ** M :`
Match preceding atom *M* or more (possessively)					`A{M,}+`	`A ** M..* :`
Match preceding atom *M* to *N* times (possessively)					`A{M,N}+`	`A ** M..N :`

Match specified separator between preceding repetitions						`REP % SEP`
						`REP %% SEP`

Match as much of the enclosed sequence as possible				`\%[SEQUENCE]`		`<*SEQUENCE>`

Rematch exact substring previously matched by *N*th group	`\N`	`\N`	`\N`	`\N`	`\N`
					`\gN`
					`\g{N}`
						`$N`

Rematch exact substring previously matched by *N*th group before current position					`\g-N`
					`\g{-N}`	

Rematch exact substring previously matched by *named*					`\g{NAME}`
					`\k<NAME>`
					`\k'NAME'`
					`\k{NAME}`
					`(?P=NAME)`
						`$<NAME>`

Escape all subsequent metacharacters until \E		`\Q`			`\Q`	
End of escaped sequence		`\E`			`\E`	

Grouping and capture

Action	BRE	ERE	EMACS	VIM	PCRE	PSIX
Group and capture *(group is numbered by order of opening bracket)*	`\( SUBPAT \)`	`( SUBPAT )`	`\( SUBPAT \)`	`\( SUBPAT \)`	`( SUBPAT )`	`( SUBPAT )`
Group and capture (*group is numbered explicitly*)			`\(?N: SUBPAT \)`		`(?<N> SUBPAT )`	`$N=( SUBPAT )`

Group and capture (*group is named explicitly*)					`(?<NAME> SUBPAT )`
					`(?'NAME' SUBPAT )`
					`(?P<NAME> SUBPAT )`
						`$<NAME>=(SUBPAT)`

Group but don't capture		`(?: SUBPAT )`	`\(?: SUBPAT \)`	`\%( SUBPAT \)`	`(?: SUBPAT )`	`[ SUBPAT ]`

Zero-width assertions

Action	BRE	ERE *(plus features only in Tcl)*	EMACS	VIM	PCRE *(plus features only in Perl 5)*	PSIX
Delimited comment		`(?# COMMENT )`			`(?# COMMENT )`	``#`( COMMENT )``
Single-line comment		`# COMMENT`			`# COMMENT`	`# COMMENT`
Match position is at start of string/file	`^`	`^` `\A`	``\` ``	`\%^`	`^` `\A`	`^`
Match position is at start of any line within the string/file			`^`	`^` `\_^`	`(?m:^)`	`^^`
Match position is at end of string/file (with optional final newline)		`\Z`			`\Z` `$`	
Match position is at end of string/file	`$`	`$`	`\'`	`\%$`	`\z`	`$`
Match position is at end of any line within the string/file (with optional newline)			`$`	`$` `\_$`	`(?m:$)`	`$$`
Match position is at index *N* in string						`<?at:N>`
Match position is at line *N* of file				`\%NL`		

Zero-width assertions

Action	BRE	ERE *(plus features only in Tcl)*	EMACS	VIM	PCRE *(plus features only in Perl 5)*	PSIX
Match position is at column *N* of file				`\%`*N*`c`		
Match position is at virtual column *N* of file				`\%`*N*`v`		

Match position is at an identifier boundary		`\b`
		`\y`
			`\b`		`\b`	`<wb>`
						`<|w>`

Match position is not at an identifier boundary		`\B`
		`\Y`
			`\B`		`\B`	`<!wb>`
						`<!|w>`

Match position is at the start of an identifier		`\<`
		`\m`
			`\<`
			`\_<`
				`\<`		`<<`
						`«`

Match position is at the end of an identifier		`\>`
		`\M`
			`\>`
			`\_>`
				`\>`		`>>`
						`»`

Match position is at the "start-of-match" position *(usually where previous match finished, or at explicit starting offset)*					`\G`	
Substring following current position would match the enclosed subpattern *("positive lookahead")*		*`(?=`* *SUBPAT* *`)`*		`\%(` *SUBPAT* `\)\@=`	`(?=` *SUBPAT* `)`	`<?before` *SUBPAT*`>`
Substring following current position would *not* match the enclosed subpattern *("negative lookahead")*		*`(?!`* *SUBPAT* *`)`*		`\%(` *SUBPAT* `\)\@!`	`(?!` *SUBPAT* `)`	`<!before` *SUBPAT*`>`
Substring preceding current position would match the enclosed subpattern *("positive lookbehind")*				`\%(` *SUBPAT* `\)\@<=`	`(?<=` *SUBPAT* `)`	`<?after` *SUBPAT*`>`
Substring preceding current position would *not* match the enclosed subpattern *("negative lookbehind")*				`\%(` *SUBPAT* `\)\@<!`	`(?<!` *SUBPAT* `)`	`<!after` *SUBPAT*`>`

Flow of control						
Action	**BRE**	**ERE**	**EMACS**	**VIM**	**PCRE**	**PSIX**
If left subpattern fails to match, try to match right subpattern from same starting position (i.e. lhs *OR* rhs)		`|`	`\\|`	`\\|`	`|`	`||`
Match both left and right subpatterns and accept the longer match of the two						`|`
If the left subpattern matches, try to match the right from the same starting position (i.e. lhs *AND* rhs)				`\&`		`&&` `&`
Call independent subpattern					`(?&NAME)`	`<NAME>`

Match the enclosed subpattern if the specified capture group has already successfully matched					`(?(N)        SUBPAT )`
					`(?(-N)       SUBPAT )`
					`(?(+N)       SUBPAT )`
					`(?(<NAME>)   SUBPAT )`
					`(?('NAME')   SUBPAT )`
					`(?(NAME)     SUBPAT )`
						`[ <?{ ?$N      }> SUBPAT ]`
						
						`[ <?{ ?$<NAME> }> SUBPAT ]`

Match one of two subpatterns depending on whether the specified capture group has successfully matched or not					`(?(N)        YES | NO )`
					`(?(-N)       YES | NO )`
					`(?(+N)       YES | NO )`
					`(?(<NAME>)   YES | NO )`
					`(?('NAME')   YES | NO )`
					`(?(NAME)     YES | NO )`
						`[ <?{ ?$N      }> YES || NO ]`
						
						`[ <?{ ?$<NAME> }> YES || NO ]`

Match the enclosed subpattern if currently recursively matching within the specified group					`(?(N)        SUBPAT )`
					`(?(-N)       SUBPAT )`
					`(?(+N)       SUBPAT )`
					`(?(<NAME>)   SUBPAT )`
					`(?('NAME')   SUBPAT )`
					`(?(NAME)     SUBPAT )`	

Flow of control						
Action	**BRE**	**ERE**	**EMACS**	**VIM**	**PCRE**	**PSIX**

Match one of two subpatterns depending on whether currently recursively matching within the specified group					`(?(R)` *YES* `|` *NO* `)`
					`(?(R`*N*`)` *YES* `|` *NO* `)`
					`(?(R&`*NAME*`)` *YES* `|` *NO* `)`	

Recursively match subpattern number *N*					`(?`*N*`)`
					`\g<`*N*`>`
					`\g'`*N*`'`
						`<~~`*N*`>`

Recursively match *N*th previous subpattern					`(?-`*N*`)`
					`\g<-`*N*`>`
					`\g'-`*N*`'`	

Recursively match *N*th next subpattern					`(?+`*N*`)`
					`\g<+`*N*`>`
					`\g'+`*N*`'`	

Recursively match complete surrounding pattern					`(?R)`
					`(?0)`
						`<~~>`

Recursively match named subpattern					`(?&`*NAME*`)`
					`(?P>`*NAME*`)`
					`\g<`*NAME*`>`
						`<`*NAME*`>`
						`<~~`*NAME*`>`

Code execution within matches						
Action	**BRE**	**ERE**	**EMACS**	**VIM**	**PCRE** *(plus features only in Perl 5)*	**PSIX**
Execute embedded code					*`(?{ CODE })`*	`{ `*CODE*` }`
Execute embedded code then match the result as a regex					*`(??{ CODE })`*	`<{ `*CODE*` }>`
Execute embedded code and fail if result is false						`<?{ `*CODE*` }>`

Execute external "callout" code, passing the identifier *N* (or zero by default)					`(?C)`
					`(?C`*N*`)`	

Configuration of matching behaviour						
Action	**BRE**	**ERE** *(feature only in Tcl)*	**EMACS**	**VIM**	**PCRE** *(plus features only in Perl 5)*	**PSIX**
Exclude anything already matched before this point from final reported match substring				`\zs`	`\K`	`<(`
Keep matching but exclude anything matched after this point from final reported match substring				`\ze`		`)>`
Restart numbering of capture groups in each separate | branch within the enclosed subpattern					`(?|` *SUBPAT* `)`	*default behaviour*
Specify a special region in which named subpatterns may be declared, but which never matches directly (only via subpattern calls)					`(?(DEFINE)` *DEFNS* `)`	`[ <!>` *DEFNS* `]`

Cumulatively enable or disable pattern modifiers for remainder of current subpattern *(specify any one or more of the modifier letters)*		*(?bceimnpqstwx)*			`(?imsx-imsx)`
					(?adlupimsx-imsx)
						`:i` `:!i`
						`:s` `:!s`
						`:P5` `:!P5`
						`:r` `:!r`
						`:bytes` `:!bytes`
						`:chars` `:!chars`

Cumulatively enable or disable pattern modifiers for enclosed subpattern *(specify any one or more of the modifier letters)*					`(?imsx-imsx:` *SUBPAT* `)` *(?adluimsx-imsx: SUBPAT)*	`[:i :!s` *SUBPAT* `]`

Backtracking control						
Action	**BRE**	**ERE**	**EMACS**	**VIM**	**PCRE**	**PSIX**
Prohibit backtracking within the enclosed subpattern				`\%(` *SUBPAT* `\)\@>`	`(?>` *SUBPAT* `)`	`[:r` *SUBPAT* `]`
Immediately cease matching current pattern (or recursive subpattern) and signal a successful match					`(*ACCEPT)`	`<?>`
Immediately backtrack					`(*FAIL)`	`<!>`
If backtracking ever reaches this directive, cause the entire surrounding pattern match to fail completely.					`(*COMMIT)`	`<commit>`
Skip immediately to the next available alternation (i.e. to after the next | operator) if backtracking ever reaches this directive. If the entire pattern ultimately matches, report back the associated name, if one is specified.					`(*THEN)`	`::`
Report back specified name if match succeeds					`(*MARK:`*NAME*`)` `(*:`*NAME*`)`	

Backtracking control						
Action	**BRE**	**ERE**	**EMACS**	**VIM**	**PCRE**	**PSIX**

If backtracking ever reaches this directive, cause the entire surrounding pattern match to fail at the current starting point (and move on to the next possible starting point). . If the entire pattern ultimately matches, report back the associated name, if one is specified.					`(*PRUNE)`
					`(*PRUNE:`*`NAME`*`)`	

If backtracking ever reaches this directive, cause the entire surrounding pattern match to fail at the current starting point. Move the starting point past the place in the string where this directive was encountered.					`(*SKIP)`	
If backtracking ever reaches this directive, cause the entire surrounding pattern match to fail at the current starting point. Move the starting point past the place in the string where the last `(*MARK:`*`NAME`*`)` was matched.					`(*SKIP:`*`NAME`*`)`	

Regular expression basic metasyntax summary

	BRE *("Just the basics")*	**ERE** *("The standard core")*	**EMACS** *("Extra backslashes")*	**VIM** *("Mostly backslashes")*	**PCRE** *("One with the lot")*	**PSIX** *("Total resyntax")*
Any char	`.`	`.`		`\_.`	`\p{Any}   (?s:.)`	`<:Any>   .`
…except newline			`.`	`.`	`\N   .`	`\N`
Charset	`[XXX]`	`[XXX]`	`[XXX]`	`[XXX]`	`[XXX]`	`<[XXX]>`
…anything except	`[^XXX]`	`[^XXX]`	`[^XXX]`	`[^XXX]`	`[^XXX]`	`<-[XXX]>`
Digit/ident/space	▨	`\d  \w  \s`	`\d  \w  \s-`	`\d  \w  \s`	`\d \w \s \h \v \R`	`\d \w \s \h \v \n`
…anything except	▨	`\D  \W  \S`	`\D  \W  \S-`	`\D  \W  \S`	`\D \W \S \H \V`	`\D \W \S \H \V`
Start/end of str	`^   $`	`^   $`	`` \`   \' ``	`\%^   \%$`	`\A   \z  \Z`	`^   $`
Start/end of line			`^   $`	`\_^   \_$`	`(?m:^)   (?m:$)`	`^^   $$`
Alternation	▨	`|`	`\\|`	`\\|`	`|`	`|   ||`
Loops:						
Maximal	`A*`	`A*   A+   A?`	`A*   A+   A?`	`A*   A\+   A\?`	`A*   A+   A?`	`A*   A+   A?`
Minimal		`A*?   A+?   A??`	`A*?   A+?   A??`		`A*?   A+?   A??`	`A*?   A+?   A??`
Possessive					`A*+   A++   A?+`	`A*:   A+:   A?:`
Maximal range		`A{M,N}`	`A\{M,N\}`	`A\{M,N}`	`A{M,N}`	`A ** M..N`
Minimal range				`A\{-M,N}`	`A{M,N}?`	`A **? M..N`
Possessive range					`A{M,N}+`	`A ** M..N:`
Capture	`\( SUBPAT \)`	`( SUBPAT )`	`\( SUBPAT \)`	`\( SUBPAT \)`	`( SUBPAT )`	`( SUBPAT )`
Numbered			`\(?N: SUBPAT \)`		`(?<N> SUBPAT )`	`$N=( SUBPAT )`
Named					`(?<NAME> SUBPAT )`	`$<NAME>=( SUBPAT )`
Non-capture	▨	`(?: SUBPAT )`	`\(?: SUBPAT \)`	`\%( SUBPAT \)`	`(?: SUBPAT )`	`[ SUBPAT ]`
Non-backtracking	▨	▨	▨	`\%( SUBPAT \)\@>`	`(?> SUBPAT )`	`[:r SUBPAT ]`
Call to subpattern	▨	▨	▨	▨	`(?&NAME)`	`<NAME>`
Ident boundaries	▨	`\<  \b  \>`	`\<  \b  \>`	`\<  \>`	`\b`	`<<  <wb>  >>`
…not at boundary	▨	`\B`	`\B`		`\B`	`<!wb>`
+'ve lookahead	▨	▨	▨	`\%( SUBPAT \)\@=`	`(?= SUBPAT )`	`<?before SUBPAT>`
-'ve lookahead	▨	▨	▨	`\%( SUBPAT \)\@!`	`(?! SUBPAT )`	`<!before SUBPAT>`
+'ve lookbehind	▨	▨	▨	`\%( SUBPAT \)\@<=`	`(?<= SUBPAT )`	`<?after SUBPAT>`
-'ve lookbehind	▨	▨	▨	`\%( SUBPAT \)\@<!`	`(?<! SUBPAT )`	`<!after SUBPAT>`
Escape a meta	`\`	`\`	`\`	`\`	`\`	`\   '…'   "…"`