**Regular Expressions**

A **Regular Expression** is a string that describes a search pattern.

There are many implementations of regular expressions:

|  |  |
| --- | --- |
| **BRE** – Basic Regular Expressions  **ERE** – Extended Regular Expressions  **PCRE** – PERL Compatible Regular Expressions | EMACS regular expressions  vi regular expressions  etc. |

This "cheat sheet" covers **BRE** and **ERE** syntax.

*IMPORTANT: Symbols change their meaning based on their position in a regular expression. E.g., ? has multiple meanings. To understand a symbol, you must examine its position in the regular expression.*

Character to match

[now many times?]

**Basic syntax:**

**How to match a character:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Text** | **Meaning** | **Text** | **Meaning** | **Equivalent to:** |
| type the character | matches the specified character | \w | matches any word character | [A-Za-z0-9\_] |
| . | any character | \W | not a "word" character | [^A-Za-z0-9\_] |
| [abc] | matches a, b, or c | \d | matches any digit | [0-9] |
| [a-e] | matches a, b, c, d, or e | \D | not a digit character | [^0-9] |
| [a-ex-z] | matches a, b, c, d, e, x, y or z | \s | matches whitespace | [ \n\r\t] |
| [^abc] | matches if **not** a, b, or c | \S | not whitespace | [^ \n\r\t] |
| [^a-e] | matches if **not** a, b, c, d, or e | [-a-e] | matches -, a, b, c, d, or e (dash must come first) | |

**How to specify the number of characters to match:** (the default is to be **greedy** – match as many as possible)

|  |  |
| --- | --- |
| **Text** | **Meaning** |
| + | matches 1 or more characters |
| \* | matches 0 or more characters |
| {n,m} | matches at least *n* characters, but not more than *m* characters. |
| {n,} | matches at least *n* characters, but there is no limit on the number of matches above n |
| {,m} | matches 0 up to m characters |
| ? | Matches 0 or 1 character, effectively making the character optional |

**How to specify the number of characters to match:** (**lazy** matching – match as few as possible)

|  |  |
| --- | --- |
| **Text** | **Meaning** |
| +? | matches 1 or more characters (as few as possible) |
| \*? | matches 0 or more characters (as few as possible) |
| {n,m}? | matches at least *n* characters, but not more than *m* characters. (as few as possible) |
| {n,}? | matches at least *n* characters, but as few as possible above *n* |
| {,m}? | matches 0 up to m characters (as few as possible) |

**How to specify where in a string to start a match (anchors):** (does not match a character, matches a position)

|  |  |  |  |
| --- | --- | --- | --- |
| **Text** | **Meaning** | **Text** | **Meaning** |
| ^ | at the beginning on the string | \b | at a word boundary |
| $ | at the end of the string | \B | not at a word boundary |

**Escape Characters:** (non-printable characters and using operator characters for a literal match)

|  |  |  |  |
| --- | --- | --- | --- |
| **Text** | **Meaning** | **Text** | **Meaning** |
| \000 | character code in octal | All of the symbols that have a special meaning to a regular expression must be "escaped" if you want to match that character: \. \+ \\* \? (repeaters) \^ \$ (anchors) \[ \] \{ \} \( \) (groups) \\ \| \/ | |
| \x00 | character code in hexadecimal |
| \xu0000 | Unicode (16 bit hexadecimal code) |
| \t | tab |
| \n | line feed |
| \r | carriage return |
| \f | form feed |

**Capture Groups:** (combine a sequence of tokens to operate on them together; capture the match in a variable)

|  |  |
| --- | --- |
| **Text** | **Meaning** |
| (ab)+ | one or more sequences of "ab" |
| (ab|cd) | match either "ab" or "cd" |
| \1, \2, … | back reference: the same string as a previous match, numbered left to right by ( |
| (?:abc) | Match the sequence "abc", but don’t create a capture group |
| (?=abc) | Match the sequence "abc", but don't include the characters in the final results |

**Flags that modify how matching works:**

|  |  |
| --- | --- |
| **Text** | **Meaning** |
| i | Default: character matching is case sensitive. ignore case: character matching ignores case. |
| g | Default: Find the first match in a string. globally: find all matches in the string. |
| m | Default: ^ and $ match at the start and end of the entire string. multiline: ^ and $ match at the start and end of each line. |

**Using regular expressions in a "search and replace" operation:** (substitution)

in vi substitution, the "replacement\_string" can use the results of the regular expression match:

:n,m s/regular\_expression/replacement\_string/g

|  |  |
| --- | --- |
| **Text** | **Meaning** |
| $& | the entire text of the match |
| $1, $2, … | the text of a specific capture group, numbered left to right by ( |
| $` | the text in the original string before the match |
| $' | the text in the original string after the match |
| $$ | include one dollar sign in the replacement string |

**Some differences between** vim **and ERE**

Start a pattern with \v to make it use ERE syntax.

|  |  |  |
| --- | --- | --- |
| **ERE** | **vi** | **Meaning** |
|  | \b | matches a backspace character |
| \b | \< \> | word boundary |
| \B |  | non-word boundary |
|  | \s | whitespace, only space and tab, [ \t] |
| \s | \\_s | whitespace characters, [ \t\r\n\v\f] |
| . | \. | Any character, except a new-line character |
| (?s). | \\_. | Any character, including a new-line character |

**VIM – default mode**

**Regular Expression Special Characters Not Requiring Escaping**

Regular expression operators ***without*** escaping (escaping them makes them literals):

|  |  |
| --- | --- |
| \ | Escape next character (use "\\" for literal backslash). |
| ^ | Start-of-line (at start of pattern). |
| $ | End-of-line. |
| . | Matches any character. |
| \* | Matches 0 or more occurrences of the previous atom. |
| ~ | Matches last given substitute string. |
| [...] | Matches any of the characters given within the brackets. |
| [^...] | Matches any character *not* given within the brackets. |
| & | In replacement pattern: insert the whole matched search pattern. |

**Regular Expression Special Characters Requiring Escaping**

Regular expression operators ***only when escaped*** (otherwise they are literals):

|  |  |
| --- | --- |
| \< | Matches beginning of a word (left word break/boundary). |
| \> | Matches end of a word (right word break/boundary). |
| \(...\) | Grouping into an atom. |
| \| | Separating alternatives. |
| \\_. | Matches any single character or end-of-line. |
| \+ | 1 or more of the previous atom (greedy). |
| \= | 0 or one of the previous atom (greedy). |
| \? | 0 or one of the previous atom (greedy). |
| \{ | Multi-item count match specification (greedy).   |  |  | | --- | --- | | \{n,m} | n to m occurrences of the preceding atom (as many as possible). | | \{n} | Exactly n occurrences of the preceding atom. | | \{n,} | At least n occurrences of the preceding atom (as many as possible). | | \{,m} | 0 to n occurrences of the preceding atom (as many as possible). | | \{} | 0 or more occurrences of the preceding atom (as many as possible). | |
| \{- | Multi-item count match specification (non-greedy).   |  |  | | --- | --- | | \{-n,m} | n to m occurrences of the preceding atom (as few as possible). | | \{-n} | Exactly n occurrences of the preceding atom. | | \{-n,} | At least n occurrences of the preceding atom (as few as possible). | | \{-,m} | 0 to n occurrences of the preceding atom (as few as possible). | | \{-} | 0 or more occurrences of the preceding atom (as few as possible). | |

Source: <http://jeetworks.org/vim-regular-expression-special-characters-to-escape-or-not-to-escape/>