A Regular Expressions Summary

Creating:

Standard regex literal	`/ … /<options>`
Extended regex literal	`%r{ … }<options>`
Formal constructor	`Regex.new(' … ', 'options')`

Regex Options:

`i`	Case insensitive pattern matching. Default is case sensitive.
`o`	Substitute once only.
`m`	Multi-line mode: The special key "." now also matches newlines.
`x`	Extended mode: Spaces/newlines allowed to increase readability.
`n e u s`	Character encoding: One of None, EUC, UTF-8, SJIS. The default encoding is the same as the source encoding.

Special Keys:

`.`	Any character except a newline (unless in mode m).
`^`	The beginning of the line or string
`$`	The ending of the line or string
`\/.^$()[]<>|?*+`	These characters require a \ prefix to appear as themselves in a regular expression.
`\A`	The beginning of the string.
`\b`	A word boundary (outside of [] only)
`\B`	Not a word boundary
`\d`	A digit (0 through 9).
`\D`	Not a digit
`\h`	A hex digit (0 through 9, a through f, and A through F).
`\H`	Not a hex digit
`\s`	A white space character (including spaces, tabs, newlines, carriage returns, and form feeds).
`\S`	Not a white space character.
`\w`	A word ("C" identifier) character.
`\W`	Not a word character.
`\xHH`	An encoded hexadecimal character value.
`\z`	The end of the string.
`\Z`	The end of the string or line.

Grouping:

`ab…z`	Sequence: Expressions a through z in sequence.
`a|b|…z`	Alternation: One and only one of a or b or etc...
`(e)`	An unnamed group. Also acts as an operator precedence modifier.
`(?<name> … )`	Define a named sub-group. Typically tagged with {0}, see below.
`\g<name>`	Invoke an named sub-group.
`[ … ]`	Any character from the set of characters in the brackets.
`[^ … ]`	Any character not in the set of characters in the brackets.

Set Special Keys:

`\] \\ \/ \-`	A “]”, “\”, “/”, or “-” character.
`a-z`	A character range.
`\b`	A backspace (0x08) character (inside a [] only).

Repetition:

`r*`	Matches r zero or more times.
`r*?`	Matches r zero or more times (non-greedy).
`r+`	Matches r one or more times.
`r+?`	Matches r one or more times (non-greedy).
`r?`	Matches r zero or one times.
`r{M,N}`	Matches r M through N times.
`r{M,N}?`	Matches r M through N times (non-greedy).
`r{M,}`	Matches r M or more times.
`r{,N}`	Matches r zero through N times.
`r{N}`	Matches r exactly N times.

Peeking Outward:

`(?= … )`	A positive look ahead, not part of the match.
`(?! … )`	A negative look ahead, not part of the match.
`(?<= … )`	A positive look behind, not part of the match.
`(?<! … )`	A negative look behind, not part of the match.

Usage:

```
pos = regex =~ string       #$PREMATCH      $MATCH      $POSTMATCH
md = regex.match(string)    #md.pre_match   md.to_s     md.post_match
                            #md[number] fetches the value of an unnamed sub-group.
                            #md[name] fetches the values of any named sub-groups.
```

Never forget: irb, http://rubular.com/, and http://teuthida-technologies.com/