### Anchors

- `^` Start of string, or start of line in multi-line pattern
- `\A` Start of string
- `$` End of string, or end of line in multi-line pattern
- `\Z` End of string
- `\b` Word boundary
- `\B` Not word boundary
- `<` Start of word
- `>` End of word

### Assertions

- `?=` Lookahead assertion
- `?!` Negative lookahead
- `?<=` Lookbehind assertion
- `?!=` or `?!<` Negative lookbehind
- `?>` Once-only Subexpression
- `?()` Condition [if then]
- `?()|` Condition [if then else]
- `#` Comment

### Quantifiers

- `*` 0 or more
- `{3}` Exactly 3
- `+` 1 or more
- `{3,}` 3 or more
- `?` 0 or 1
- `{3,5}` 3, 4 or 5

The escape character is usually `/`. Add a ? to a quantifier to make it ungreedy.

### Escape Sequences

- `\` Escape following character
- `\Q` Begin literal sequence
- `\E` End literal sequence

“Escaping” is a way of treating characters which have a special meaning in regular expressions literally, rather than as special characters.

### Pattern Modifiers

- `g` Global match
- `i` Case-insensitive
- `m` Multiple lines
- `s` Treat string as single line
- `x` Allow comments and whitespace in pattern
- `e` Evaluate replacement
- `U` Ungreedy pattern
- `*` PCRE modifier

### String Replacement

- `$n` nth non-passive group
- `$2 “xyz” in /&(abc)(xyz)$/
- `$1 “xyz” in /&(?:abc)(xyz)$/
- `$` Before matched string
- `$+` After matched string
- `$+` Last matched string
- `$+` Entire matched string

Some regex implementations use \ instead of $.

### Groups and Ranges

- `.` Any character except new line (`\n`)
- `(a|b)` a or b
- `(...)` Group
- `(?:...)` Passive (non-capturing) group
- `[abc]` Range (a or b or c)
- `[^abc]` Not (a or b or c)
- `[a-q]` Lower case letter from a to q
- `[A-Q]` Upper case letter from A to Q
- `[0-7]` Digit from 0 to 7
- `\x` Group/subpattern number "x"

Ranges are inclusive.

### Character Classes

- `[^c]` Control character
- `[^s]` White space
- `[^S]` Not white space
- `[^d]` Digit
- `[^D]` Not digit
- `[^w]` Word
- `[^W]` Not word
- `[^x]` Hexadecimal digit
- `[^O]` Octal digit

### POSIX

- `[upper:]` Upper case letters
- `[lower:]` Lower case letters
- `[alpha:]` All letters
- `[alnum:]` Digits and letters
- `[digit:]` Digits
- `[xdigit:]` Hexadecimal digits
- `[punct:]` Punctuation
- `[blank:]` Space and tab
- `[word:]` Digits, letters and underscore

### Common Metacharacters

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>^</code></td>
<td>Start of string</td>
</tr>
<tr>
<td><code>{</code></td>
<td>Group</td>
</tr>
<tr>
<td><code>+</code></td>
<td>1 or more</td>
</tr>
<tr>
<td><code>?</code></td>
<td>0 or 1</td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>Start of word</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>End of word</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\</code></td>
<td>Escape following character</td>
</tr>
<tr>
<td><code>\Q</code></td>
<td>Begin literal sequence</td>
</tr>
<tr>
<td><code>\E</code></td>
<td>End literal sequence</td>
</tr>
</tbody>
</table>

The escape character is usually `/`. Add a ? to a quantifier to make it ungreedy.

### Special Characters

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\n</code></td>
<td>New line</td>
</tr>
<tr>
<td><code>\r</code></td>
<td>Carriage return</td>
</tr>
<tr>
<td><code>\t</code></td>
<td>Tab</td>
</tr>
<tr>
<td><code>\v</code></td>
<td>Vertical tab</td>
</tr>
<tr>
<td><code>\f</code></td>
<td>Form feed</td>
</tr>
<tr>
<td><code>\x</code></td>
<td>Octal character xxx</td>
</tr>
<tr>
<td><code>\xhh</code></td>
<td>Hex character hh</td>
</tr>
</tbody>
</table>

Published 19th October, 2011.

Everyone has a novel in them. Finish Yours!
https://apollopad.com