Regular expressions

.	any one character
gr.y	gray, grey, tray, grby, gr1y, gr y, gr!y, greay
(|)	Match any of the literal strings between the brackets
(Sun|Mon|Tue)day	Monday, Tuesday, Wednesday, Sunday, sunday
[ ]	any character between brackets
gr[ea]y	grey, gray, groy, gruy, griy
[-]	includes ranges: e.g. [b-f] [5-8] [a-z] [A-Za-z] [0-9]
H[2-4]0	H0, H10, H20, H30, H40, H50
[^]	any character *except* those listed
c[^u]t	cut, cat, cbt, cct, cdt, cet, c t, c-t, cát, c?t, caat

Regex: Quantifiers

`{num}`	match previous element (*num*) times
`b[ao]{2}t`	boaat, boot, baat, boat, baot, bot, bat
`{min,max}`	match previous element between (*min*) and (*max*) times
`[A-Z]o{,4}h`	Boooh! Booooooh!
`*`	match previous element zero or more times
`la*`	Oh la la la. Oh la la laaa! C'est magnifique! All at once!
`?`	match previous element zero or one time
`beholde?`	behold, beholde

Regex: Replacements

`(...)`	replacement group (in search expression)
`\\1`	replacement group 1 (in replace expression)
`(t?here) it is (cold|warm)`	And he remarked: there it is cold. She added: here it is warm.
`it was \\2 out \\1.`	And he remarked: it was cold out there. She added: it was warm out there.

Regex: Greedy – lazy

`.*`	‘stuff’
`anti.*ism`	The anti-disestablishmentarianism really got her down…
`*`	greedy
`str.*re`	A structure is a reality which is immaterial, but manifests itself materially.
`*?`	lazy
`str.*?re`	A structure is a reality which is immaterial, but manifests itself materially.

.* (greedy stuff) | .*? (lazy stuff)

Regex: Other options

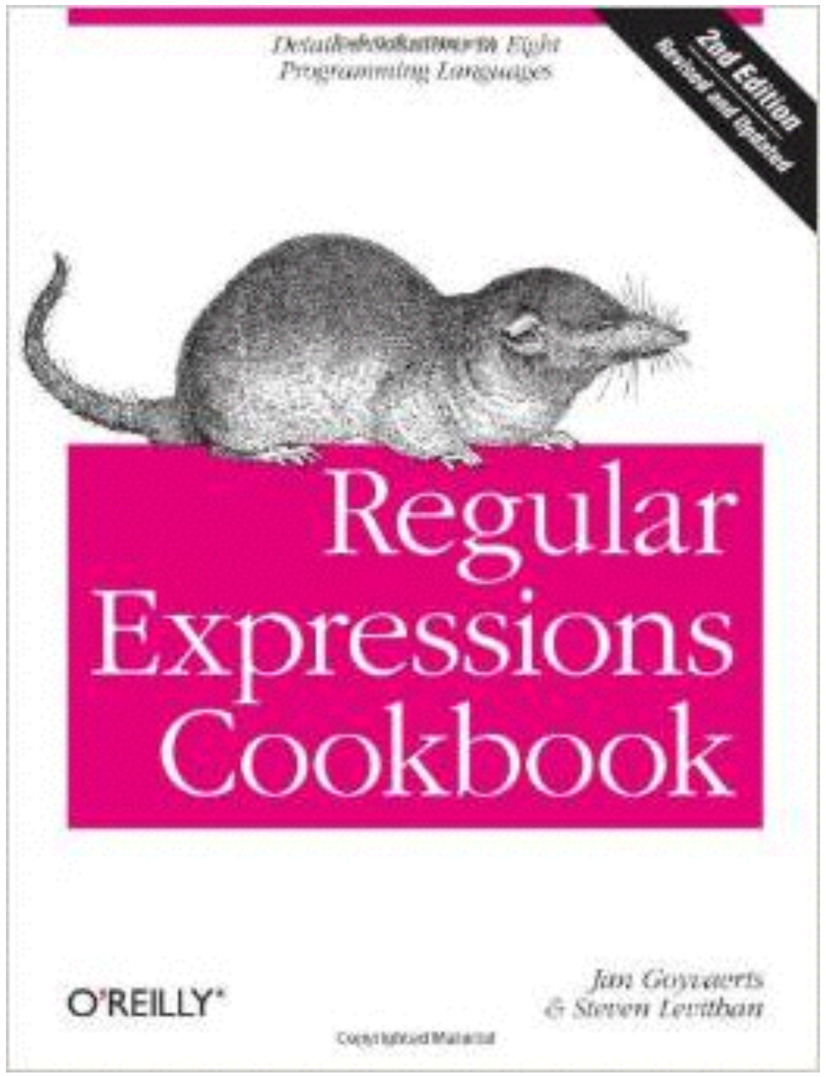
(?s)	makes following regex ignore line breaks
(?s)he (is|was) here	yesterday morning, he was here for a minute or two
\	escape special characters: \\. \\\\ \\? \\*
math.*?\\.	A horse that can count to ten is a remarkable horse, not a remarkable mathematician.
\\\\n \\\\t	new line \\n - tab \\t
\\\\w \\\\d	any word character - any digit

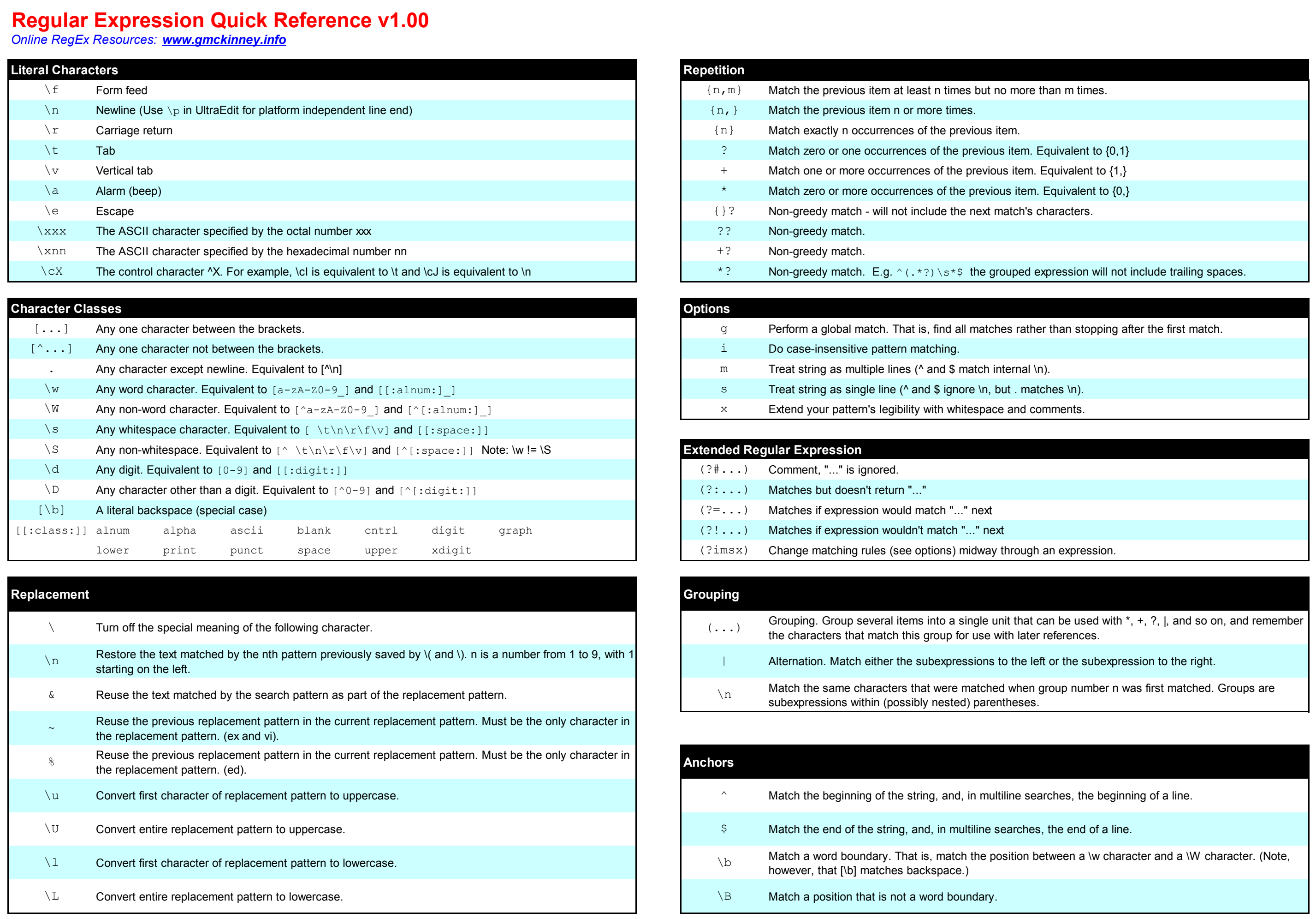
Regex: Anchors and boundaries

^	Matches at start of line or element
^John	John S. is coming, but John. D. isn't.
$	Matches at end of line or element
enough$	Enough is enough
\\w - \\W	any word / non-word character (consuming)
\\WJohn\\W	This is John Johnson.
\\b	word boundary (non-consuming!) (>< \\B)
\\bJohn\\b	This is John Johnson.

See also: lookaround (non-consuming)

Further reference





Regular Expression Quick Reference v1.00
Online RegEx Resources: www.gmckinney.info

Literal Characters

\f	Form feed
\n	Newline (Use \p in UltraEdit for platform independent line end)
\r	Carriage return
\t	Tab
\v	Vertical tab
\a	Alarm (beep)
\e	Escape
\xxx	The ASCII character specified by the octal number xxx
\xnn	The ASCII character specified by the hexadecimal number nn
\cX	The control character ^X. For example, \cI is equivalent to \t and \cJ is equivalent to \n

Character Classes

[...]	Any one character between the brackets.
[^...]	Any one character not between the brackets.
.	Any character except newline. Equivalent to [^\n]
\w	Any word character. Equivalent to [a-zA-Z0-9_] and [[:alnum:]_]
\W	Any non-word character. Equivalent to [^a-zA-Z0-9_] and [^[:alnum:]_]
\s	Any whitespace character. Equivalent to [\t\n\r\f\v] and [[:space:]]
\S	Any non-whitespace. Equivalent to [^ \t\n\r\f\v] and [^[:space:]] Note: \w != \S
\d	Any digit. Equivalent to [0-9] and [[:digit:]]
\D	Any character other than a digit. Equivalent to [^0-9] and [^[:digit:]]
[\b]	A literal backspace (special case)
[[:class:]]	alnum alpha ascii blank cntrl digit graph
	lower print punct space upper xdigit

Replacement

\	Turn off the special meaning of the following character.
\n	Restore the text matched by the nth pattern previously saved by \(and \). n is a number from 1 to 9, with 1 starting on the left.
&	Reuse the text matched by the search pattern as part of the replacement pattern.
~	Reuse the previous replacement pattern in the current replacement pattern. Must be the only character in the replacement pattern. (ex and vi).
%	Reuse the previous replacement pattern in the current replacement pattern. Must be the only character in the replacement pattern. (ed).
\u	Convert first character of replacement pattern to uppercase.
\U	Convert entire replacement pattern to uppercase.
\l	Convert first character of replacement pattern to lowercase.
\L	Convert entire replacement pattern to lowercase.

Repetition

{n,m}	Match the previous item at least n times but no more than m times.
{n,}	Match the previous item n or more times.
{n}	Match exactly n occurrences of the previous item.
?	Match zero or one occurrences of the previous item. Equivalent to {0,1}
+	Match one or more occurrences of the previous item. Equivalent to {1,}
*	Match zero or more occurrences of the previous item. Equivalent to {0,}
{}?	Non-greedy match - will not include the next match's characters.
??	Non-greedy match.
+?	Non-greedy match.
*?	Non-greedy match. E.g. ^(.*?)\s*$ the grouped expression will not include trailing spaces.

Options

g	Perform a global match. That is, find all matches rather than stopping after the first match.
i	Do case-insensitive pattern matching.
m	Treat string as multiple lines (^ and $ match internal \n).
s	Treat string as single line (^ and $ ignore \n, but . matches \n).
x	Extend your pattern's legibility with whitespace and comments.

Extended Regular Expression

(?#...)	Comment, "..." is ignored.
(?:...)	Matches but doesn't return "..."
(?=...)	Matches if expression would match "..." next
(?!...)	Matches if expression wouldn't match "..." next
(?imsx)	Change matching rules (see options) midway through an expression.

Grouping

(...)	Grouping. Group several items into a single unit that can be used with *, +, ?, |, and so on, and remember the characters that match this group for use with later references.
|	Alternation. Match either the subexpressions to the left or the subexpression to the right.
\n	Match the same characters that were matched when group number n was first matched. Groups are subexpressions within (possibly nested) parentheses.

Anchors

^	Match the beginning of the string, and, in multiline searches, the beginning of a line.
$	Match the end of the string, and, in multiline searches, the end of a line.
\b	Match a word boundary. That is, match the position between a \w character and a \W character. (Note, however, that [\b] matches backspace.)
\B	Match a position that is not a word boundary.