Regular Expressions cheat sheet

Basic matching

Each symbol matches a single character:

`.`	anything[1]
`\d`	digit in `0123456789`
`\D`	non-digit
`\w`	“word” (letters and digits and _)
`\W`	non-word
`␣`	space
`\t`	tab
`\r`	return
`\n`	new line[2]
`\s`	whitespace (`␣`, `\t`, `\r`, `\n`)
`\S`	non-whitespace

Character classes

Character classes [...] match any of the characters in the class. **Ex:** `[aeiou]` matches vowels. Use ˆ to specify the complement set: `[ˆaeiou]` matches non-vowels (including non-letters!). Use `-` to specify a range of letters: `[a-e]` matches `abcde` and `[0-9a-f]` matches ‘`0123456789abcdef`‘.

Boundaries

Boundary characters are helpful in “anchoring” your pattern to some edge, but do not select any characters themselves.

`\b`	word boundaries (defined as any edge between a `\w` and `\W`)
`\B`	non-word-boundaries
`ˆ`	the beginning of the line
`$`	the end of the line

Ex: `\bcat\b` finds a match in “the cat in the hat” but not in “locate”.

Disjunction

`(X|Y)`	`X` or `Y`

Ex: `\b(cat|dog)s\b` matches cats and dogs.

“Quantifiers”

`X*`	0 or more repetitions of `X`
`X+`	1 or more repetitions of `X`
`X?`	0 or 1 instances of `X`
`X{`m`}`	exactly m instances of `X`
`X{`m`,}`	at least m instances of `X`
`X{`m`,`n`}`	between m and n (inclusive) instances of `X`

By default, quantifiers just apply to the one character. Use `(...)` to specify explicit quantifier “scope.”

Ex: `ab+` matches `ab,abb,abbb,abbbb`...
`(ab)+` matches `ab,abab,ababab`...

Quantifiers are by default *greedy* in regex. Good regex engines support adding `?` to a quantifier to make it *lazy*.

Ex: *greedy:* `^.*b` $\underline{\text{aabaaba}}$
lazy: `^.*?b` $\underline{\text{aab}}\text{aaba}$

Special characters

The characters `{}[]()^$.|*+?\` (and `-` inside `[...]`) have special meaning in regex, so they must be “escaped” with `\` to match them.

Ex: `\.` matches the period `.` and `\\` matches the backslash `\` .

Backreferences

Count your open parentheses `(` from the left, starting with 1. Whatever is matched by parenthesis number n can be referenced later by `\`n.

Ex: `\b(\w+)␣\1\b` matches two identical words with a space in between

Backreferences are useful for *find/replace*s:

Ex: Finding `\b(\w+)er\b` and replacing with `more \1` will map “the taller man” ↦ “the more tall man” and “I am shorter” ↦ “I am more short”.

Advanced

Read about “non-capturing parentheses” and “look-ahead” and “look-behind” online. Also, visualize your regexes as finite-state machines at `http://www.regexper.com/`.

[1]...except line breaks, depending on your engine.

[2]Depending on where you got your file, line breaks may be `\r`, `\n`, or `\r\n`. Also, in some regex engines (e.g. TextWrangler), `\r` and `\n` match the same things.

Hadas Kotek (courtesy of Michael Y. Erlewine)