# Regular expressions Pattern matching syntax

**START HERE**

**abc**
Literal text

**^...$**
Whole line

**.**
Any one character

**a|b**
Either or

**\d+**
One or more digits

**(...)**
Group and capture

**[a-z]**
One of a set

**colou?r**
Optional part

**Anchor** ^ \b \A → **then** **Atom** a . \d [abc] (...) → **repeated by** **Quantifier** ? * + {n,m} → **joined by** **Alternation** a|b

## CONCEPTS

A regex is matched left to right: at each position the engine tries the whole pattern and moves on if it fails.

Flavours differ. This sheet follows PCRE2 (Perl-style), also used by PHP and grep -P; JavaScript, Python and POSIX tools lack some features.

Prefer a specific class over .* : greedy wildcards cause surprising matches and slow backtracking.

## CHARACTERS

**.**
Any char except newline
*With the s flag it matches newlines too.*

**\d \w \s**
Digit, word char, space
*Uppercase \D \W \S is the negation. ASCII only unless Unicode mode is on.*

**[abc]**
Any one of a, b, c

**[^abc]**
Any char except a, b, c

**[a-z0-9]**
Ranges inside a set

**\.**
Escape a metacharacter
*Metacharacters: . ^ $ * + ? ( ) [ ] { } | *

**\n \t \r**
Newline, tab, carriage return

**\x41 \x{263A}**
Character by hex code

**[[:alpha:]]**
POSIX class inside a set
*Also digit, alnum, upper, lower, space, punct, xdigit.*

**\h \v \R**
Horizontal space, vertical space, any line break

**\Q...\E**
Treat text literally

## CHARACTERS (cont.)

**\p{L} \p{Lu}**
Unicode letter, uppercase letter
*Needs Unicode mode (UTF) to see non-ASCII text.*

**\p{Greek}**
Unicode script

## QUANTIFIERS

*****
Zero or more

**+**
One or more

**?**
Zero or one (optional)

**{n}**
Exactly n times

**{n,m}**
Between n and m times

**{n,}**
At least n times

***? +? ??**
Lazy: as few as possible
*Classic use: <.+?> matches one tag, <.+> swallows the line.*

***+ ++ ?+**
Possessive: never give back
*Faster failure; cannot backtrack into the match.*

## ANCHORS

**^ $**
Start, end of line
*Without the m flag: start of text, end of text (or before a final newline).*

**\b**
Word boundary

**\B**
Not a word boundary

**\A \z**
Start, end of text

**\Z**
End, before a final newline

**\G**
Where the previous match ended

## GROUPS

**(...)**
Capture group

## GROUPS (cont.)

**a|b**
Alternation
*Lowest precedence: gr(a|e)y, not gra|ey.*

**(?:...)**
Group without capturing

**(?<name>...)**
Named capture group

**\1 \2**
Backreference to group n
*(\w)\1 finds doubled letters.*

**\k<name>**
Backreference by name

**(?>...)**
Atomic group
*Once matched, the engine will not backtrack into it.*

**(?|...)**
Branch reset: reuse group numbers

**(?#...)**
Inline comment

## LOOKAROUND

**(?=...)**
Followed by (lookahead)

**(?!...)**
Not followed by

**(?<=...)**
Preceded by (lookbehind)
*Each lookbehind branch must have a bounded length.*

**(?<!...)**
Not preceded by

**\K**
Reset start of reported match
*Keeps the text before it out of the match; handy in replacements.*

### Zero-width checks

| Syntax | Asserts |
|---|---|
| (?=x) | x comes next |
| (?!x) | x does not come next |
| (?<=x) | x comes before |
| (?<!x) | x does not come before |

## FLAGS

**(?i)**
Ignore case
*Applies to the rest of the pattern or group.*

## FLAGS (cont.)

**(?m)**
^ and $ match at each line

**(?s)**
Dot matches newline

**(?x)**
Free-spacing: ignore blanks, allow #comments

**(?i:...)**
Flag for one group only

**(?-i)**
Turn a flag off

**(?U)**
Quantifiers lazy by default

**(*UCP)**
Make \d \w \s Unicode-aware
*Place at the very start of the pattern.*

## ADVANCED

**(?R)**
Recurse the whole pattern
*Matches nested brackets, e.g. \((?:[^()]|(?R))*\).*

**(?1) (?&name)**
Call a group as a subroutine

**(?(1)yes|no)**
Conditional on a group

**(?(DEFINE)...)**
Define named groups for reuse

**(*SKIP)(*FAIL)**
Match then discard (skip this)

**(*ACCEPT)**
End the match successfully

## RECIPES

### Integer or decimal

```
^-?\d+(?:\.\d+)?$
```

### ISO date with named parts

```
(?<y>\d{4})-(?<m>\d{2})-(?
<d>\d{2})
```

### Trim trailing whitespace

```
# match, replace with nothing
[ \t]+$   (with (?m))
```

### Value after a key

```
# match only the number
price=\K\d+
# or with lookbehind
(?<=price=)\d+
```

### Simple email (loose)

```
^[\w.+-]+@[\w-]+(?:\.[\w-]+)+$
```