JavaScript Regex Cheat Sheet

rjl.io/regex-tester/

Character classes

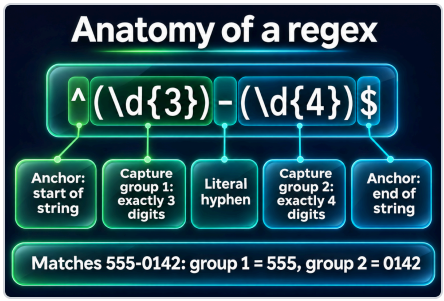
Token	Matches
`.`	Any character except a line break (`s` flag: including)
`\d \D`	Digit `[0-9]` / non-digit
`\w \W`	Word char `[A-Za-z0-9_]` / other
`\s \S`	Whitespace / non-whitespace
`[abc]`	One of a, b, c
`[^abc]`	Anything except a, b, c
`[a-z0-9]`	One char from a range
`\p{L}`	Unicode letter (needs `u` flag)
`\. \* \\`	Escaped literal characters

Anchors

Token	Matches
`^ $`	Start / end of string (each line with `m`)
`\b \B`	Word boundary / not a boundary

Quantifiers

Token	Matches
`*`	0 or more (greedy)
`+`	1 or more (greedy)
`?`	0 or 1
`{3}`	Exactly 3
`{3,}`	3 or more
`{3,5}`	3 to 5
`*? +? ?? {3,5}?`	Lazy: as few as possible



Groups and lookarounds

Token	Meaning
`(abc)`	Capture group (numbered from 1)
`(?:abc)`	Non-capturing group
`(?<year>\d{4})`	Named group
`\1 \k<year>`	Backreference
`a|b`	a or b
`(?=x) (?!x)`	Followed by / not followed by x
`(?<=x) (?<!x)`	Preceded by / not preceded by x

Flags

Flag	Effect
`g`	Global: all matches
`i`	Ignore case
`m`	Multiline `^` and `$`
`s`	Dot matches line breaks
`u`	Unicode mode, `\p{...}`

Replacement tokens

Token	Inserts
`$1 $2`	Capture group 1, 2, ...
`$<name>`	Named group
`$&`	Whole match
`$$`	A literal dollar sign

Common patterns

Use	Pattern
Integer	`^-?\d+$`
Date YYYY-MM-DD	`\d{4}-(?:0[1-9]|1[0-2])-(?:0[1-9]|[12]\d|3[01])`
Time HH:MM (24h)	`(?:[01]?\d|2[0-3]):[0-5]\d`
Hex color	`#(?:[0-9a-fA-F]{3}){1,2}\b`
IPv4 address	`\b(?:(?:25[0-5]|2[0-4]\d|[01]?\d\d?)\.){3}(?:25[0-5]|2[0-4]\d|[01]?\d\d?)\b`
URL slug	`^[a-z0-9]+(?:-[a-z0-9]+)*$`
HTML tag	`<[^>]+>`
Quoted text	`"([^"]*)"`
Extra spaces	`\s{2,}`

Greedy vs lazy, and common mistakes

Topic	Detail
Greedy vs lazy	On `<b>one</b><b>two</b>`, `<.+>` matches the whole string; `<.+?>` matches `<b>` first.
Unescaped dot	`example.com` also matches `exampleXcom`. Write `example\.com`.
No anchors	`\d{3}` matches inside `a1234b`. Use `^\d{3}$` to validate.
Nested quantifiers	`(a+)+$` can backtrack catastrophically on long failing input.
`g` with `.test()`	The regex keeps `lastIndex`, so repeated calls can alternate. Drop `g` or reset it to 0.