REGEX CHEAT SHEET
regexguide.com/patterns/camelcase.html

camelCase identifier

Lowercase first word followed by capitalized words.

```
^[a-z]+([A-Z][a-z0-9]*)*$
```

Token-by-token breakdown

Token	Meaning
`^`	start of string (or line in multiline mode)
`[a-z]+`	one or more: any of: lowercase letters
`(`	start of a capturing group
`[A-Z]`	any of: uppercase letters
`[a-z0-9]*`	zero or more: any of: lowercase letters, digits
`)*`	end of group, zero or more
`$`	end of string (or line in multiline mode)

Quick usage in 7 languages

JavaScript	`const re = /^[a-z]+([A-Z][a-z0-9]*)*$/;` `re.test(value);`
Python	`import re` `re.match(r"^[a-z]+([A-Z][a-z0-9]*)*$", value)`
Java	`Pattern.compile("^[a-z]+([A-Z][a-z0-9]*)*$").matcher(value).matches();`
C# / .NET	`Regex.IsMatch(value, @"^[a-z]+([A-Z][a-z0-9]*)*$");`
Go	``regexp.MustCompile(`^[a-z]+([A-Z][a-z0-9]*)*$`).MatchString(value)``
Ruby	`/^[a-z]+([A-Z][a-z0-9]*)*$/.match?(value)`
PHP	`preg_match('~^[a-z]+([A-Z][a-z0-9]*)*$~', $value);`

Common pitfalls

- **Anchored to the whole string.** This pattern uses ^ and $, so it requires the entire input to match. To find it inside a longer text, drop the anchors and use the global (g) flag.
- **Case sensitivity.** Letter ranges are case-sensitive — use the i flag if the input case can vary.
- **Watch for backtracking.** This pattern nests quantifiers; on adversarial input that can cause exponential backtracking (ReDoS). Test with long non-matching strings, or use an RE2-based engine.
- **Validate beyond format.** Matching the format doesn't guarantee the value is real. Confirm the camelcase identifier against a source of truth (database, API, or checksum) where it matters.