HOW SOFTWARE WORKS · BUILDER · 14 MIN READ

Regular expressions without fear: read, test and trust them

AI writes regular expressions in seconds, and they look like line noise. Learn to read one piece at a time, know what the flags do, test a pattern with examples before you trust it, and know when not to use one.

Pass the check (4 of 5) online to earn the “Pattern Reader” badge:
https://digitalbuilders757.com/university/study/regular-expressions-without-fear

What a regex is

A regular expression (regex) is a tiny pattern language for finding text: every email address in a document, the date in a filename, the digits in a phone number. AI writes them in seconds. Your job is to read them and test them before you trust them.

In JavaScript a regex sits between slashes, like /cat/i, and you use it with test(), match(), replace() and split(). Python, most code editors and many databases use nearly the same syntax.

Read it one piece at a time

- Letters and digits match themselves: /cat/ finds 'cat'.
- . matches any one character. \d is a digit, \w a letter, digit or underscore, and \s a space, tab or new line.
- [abc] matches one of a, b or c; [^abc] anything else; [a-z] any letter from a to z.
- Quantifiers say how many of the thing before them: ? zero or one, * zero or more, + one or more, {3} exactly three, {2,4} two to four.
- ^ and $ pin the match to the start and end, so /^\d{5}$/ means exactly five digits and nothing else.
- Brackets group a part, and | means or: /(cat|dog)s?/ finds cat, cats, dog or dogs.
- To match a special character itself, put a backslash in front: \. is a real full stop.

Flags change the rules

- i ignores case: /hello/i matches 'Hello'.
- g means every match, not just the first: '2026-11-26'.replace(/-/g, '/') gives '2026/11/26'.

- Without g, replace() changes only the first match, a classic bug in AI-written code.
- m makes ^ and $ work on each line, and u handles full Unicode, such as emoji.

Test it before you trust it

- Write examples first: a few strings it must match and a few it must not, including awkward ones such as an empty string, spaces and capitals.
- Run the pattern against every example in the playground or an online regex tester, and look at each result.
- Ask your AI helper to explain the regex piece by piece, and to list strings it would get wrong.
- Keep your examples as tests, so a later change can't quietly break the pattern.

When not to use a regex

- Emails: a simple check for something@something.something, plus a confirmation email, beats a giant pattern.
- HTML, JSON, URLs and dates have proper parsers: DOMParser, JSON.parse, new URL() and date libraries.
- Nested repeats such as (a+)+ can take a very long time on some text, which can freeze a server (ReDoS). Keep patterns simple and limit input length.

Regex bugs to spot in AI-written code

- No ^ and $, so 'abc12345xyz' passes a five-digit check.
- An unescaped dot: /example.com/ also matches 'exampleXcom'.
- replace() without g, so only the first match changes.
- Case: /^[a-z]+$/ rejects 'Ana' unless you add the i flag or A-Z.
- A pattern built from user input with new RegExp(text): special characters such as . and * change what it means.
- Checks only in the browser. The server must check again.

Key terms

Regular expression (regex)
A small pattern language for finding or checking text, like /\d{5}/ for five digits in a row.

Regex flags
Letters after a regex that change its rules: g finds every match, i ignores case, m works line by line.

Quantifier (regex)
How many times the part before it may repeat: ? zero or one, * zero or more, + one or more, {3} exactly three.

Anchor (regex)
^ and $ pin a pattern to the start and end of the text, so the whole text must match, not just part of it.

Capture group
Brackets in a regex that group a part and remember what it matched, so you can use it afterwards.

ReDoS
Regular expression denial of service: a pattern that takes so long on certain text that it freezes the program running it.

Form validation
Checking what someone typed before using it: in the browser for quick feedback, and always again on the server.

Try it: Fix two regex bugs

1. Open the 'Test a regex' starter in the code playground and run it. Which check passes when it shouldn't?
2. Add ^ and $ to the five-digit pattern so only exactly five digits pass, and run it again.
3. Make the replace() change every dash, not just the first.
4. Ask your AI helper for a regex that checks something you need, such as an order number. Write three strings that should pass and three that shouldn't, and test them all.
5. Ask it to explain each piece in plain words, then check one piece yourself against this guide.

KNOWLEDGE CHECK

Regular expressions without fear: read, test and trust them

5 questions. Circle one answer for each; 4 or more correct passes. Answers are at the back.

1. /\d{5}/ is meant to accept exactly five digits. Why does 'abc12345xyz' pass?

A. It has no ^ and $, so five digits anywhere in the text are enough
B. \d also matches letters whenever the text is longer than five characters
C. {5} means at least five of anything, digits or not
D. test() ignores everything that isn't a number

2. What does '2026-11-26'.replace(/-/, '/') give?

A. 2026/11/26
B. 2026/11-26
C. 2026-11-26
D. /2026-11-26

3. Which pattern matches a real full stop, as in 'example.com'?

A. /example.com/
B. /example*com/
C. /example\.com/
D. /example?com/

4. AI suggests a 400-character regex to check email addresses. What's the better plan?

A. Use it exactly as it is, because longer patterns always catch more mistakes
B. Check the email in the browser only, so the server stays fast
C. Ask for an even longer pattern that covers every address
D. A simple something@something.something check, then a confirmation email

5. What does the i flag do in /hello/i?

A. Ignores upper and lower case
B. Finds every match instead of the first
C. Makes ^ and $ work on each line
D. Treats the pattern as plain text

Official sites

MDN: Regular expressions guide:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_expressions
MDN: Regular expression syntax cheat sheet:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_expressions/Cheatsheet

OWASP: Regular expression denial of service (ReDoS): https://owasp.org/www-community/attacks/Regular_expression_Denial_of_Service_-_ReDoS

Last reviewed 2026-11-25. AI tools change fast; check each tool's own site for current features, plans and prices.

ANSWER KEY

Regular expressions without fear: read, test and trust them

1. A: It has no ^ and $, so five digits anywhere in the text are enough
Without anchors, a regex looks for a match anywhere. /^\d{5}$/ means the whole text is five digits.

2. B: 2026/11-26
Without the g flag, replace() changes only the first match. /-/g would change both dashes.

3. C: /example\.com/
A bare . matches any character, so /example.com/ also matches 'exampleXcom'. \. matches only a full stop.

4. D: A simple something@something.something check, then a confirmation email
The only real test of an email address is sending to it. Huge patterns are hard to read, reject real addresses and can be slow.

5. A: Ignores upper and lower case
i ignores case, so /hello/i matches 'Hello' and 'HELLO'. g finds every match, and m works line by line.