Quantifiers

Quantifiers indicate numbers of characters or expressions to match.

onst ghostSpeak = 'booh boooooooh';

const regexpSpooky = /bo{3,}h/;

console.log(ghostSpeak.match(regexpSpooky));

// Expected output: Array ["boooooooh"]

const modifiedQuote = '[He] ha[s] to go read this novel [Alice in Wonderland].';

const regexpModifications = /\[.\*?\]/g;

console.log(modifiedQuote.match(regexpModifications));

// Expected output: Array ["[He]", "[s]", "[Alice in Wonderland]"]

const regexpTooGreedy = /\[.\*\]/g;

console.log(modifiedQuote.match(regexpTooGreedy));

// Expected output: Array ["[He] ha[s] to go read this novel [Alice in Wonderland]"]

[**Types**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_expressions/Quantifiers#types)

**Note:** In the following, *item* refers not only to singular characters, but also includes [character classes](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_expressions/Character_classes) and [groups and backreferences](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_expressions/Groups_and_backreferences).

| **Characters** | **Meaning** |
| --- | --- |
| *x*\* | Matches the preceding item "x" 0 or more times. For example, /bo\*/ matches "boooo" in "A ghost booooed" and "b" in "A bird warbled", but nothing in "A goat grunted". |
| *x*+ | Matches the preceding item "x" 1 or more times. Equivalent to {1,}. For example, /a+/ matches the "a" in "candy" and all the "a"'s in "caaaaaaandy". |
| *x*? | Matches the preceding item "x" 0 or 1 times. For example, /e?le?/ matches the "el" in "angel" and the "le" in "angle."  If used immediately after any of the quantifiers \*, +, ?, or {}, makes the quantifier non-greedy (matching the minimum number of times), as opposed to the default, which is greedy (matching the maximum number of times). |
| *x*{*n*} | Where "n" is a positive integer, matches exactly "n" occurrences of the preceding item "x". For example, /a{2}/ doesn't match the "a" in "candy", but it matches all of the "a"'s in "caandy", and the first two "a"'s in "caaandy". |
| *x*{*n*,} | Where "n" is a positive integer, matches at least "n" occurrences of the preceding item "x". For example, /a{2,}/ doesn't match the "a" in "candy", but matches all of the a's in "caandy" and in "caaaaaaandy". |
| *x*{*n*,*m*} | Where "n" is 0 or a positive integer, "m" is a positive integer, and *m* > *n*, matches at least "n" and at most "m" occurrences of the preceding item "x". For example, /a{1,3}/ matches nothing in "cndy", the "a" in "candy", the two "a"'s in "caandy", and the first three "a"'s in "caaaaaaandy". Notice that when matching "caaaaaaandy", the match is "aaa", even though the original string had more "a"s in it. |
| *x*\*? *x*+? *x*?? *x*{n}? *x*{n,}? *x*{n,m}? | By default quantifiers like \* and + are "greedy", meaning that they try to match as much of the string as possible. The ? character after the quantifier makes the quantifier "non-greedy": meaning that it will stop as soon as it finds a match. For example, given a string like "some <foo> <bar> new </bar> </foo> thing":   * /<.\*>/ will match "<foo> <bar> new </bar> </foo>" * /<.\*?>/ will match "<foo>" |

[**Examples**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_expressions/Quantifiers#examples)

[**Repeated pattern**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_expressions/Quantifiers#repeated_pattern)

JS

const wordEndingWithAs = /\w+a+\b/;

const delicateMessage = "This is Spartaaaaaaa";

console.table(delicateMessage.match(wordEndingWithAs)); // [ "Spartaaaaaaa" ]

[**Counting characters**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_expressions/Quantifiers#counting_characters)

JS

const singleLetterWord = /\b\w\b/g;

const notSoLongWord = /\b\w{2,6}\b/g;

const longWord = /\b\w{13,}\b/g;

const sentence = "Why do I have to learn multiplication table?";

console.table(sentence.match(singleLetterWord)); // ["I"]

console.table(sentence.match(notSoLongWord)); // [ "Why", "do", "have", "to", "learn", "table" ]

console.table(sentence.match(longWord)); // ["multiplication"]

[**Optional character**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_expressions/Quantifiers#optional_character)

JS

const britishText = "He asked his neighbour a favour.";

const americanText = "He asked his neighbor a favor.";

const regexpEnding = /\w+ou?r/g;

// \w+ One or several letters

// o followed by an "o",

// u? optionally followed by a "u"

// r followed by an "r"

console.table(britishText.match(regexpEnding));

// ["neighbour", "favour"]

console.table(americanText.match(regexpEnding));

// ["neighbor", "favor"]

[**Greedy versus non-greedy**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_expressions/Quantifiers#greedy_versus_non-greedy)

JS

const text = "I must be getting somewhere near the center of the earth.";

const greedyRegexp = /[\w ]+/;

// [\w ] a letter of the latin alphabet or a whitespace

// + one or several times

console.log(text.match(greedyRegexp)[0]);

// "I must be getting somewhere near the center of the earth"

// almost all of the text matches (leaves out the dot character)

const nonGreedyRegexp = /[\w ]+?/; // Notice the question mark

console.log(text.match(nonGreedyRegexp));

// "I"

// The match is the smallest one possible