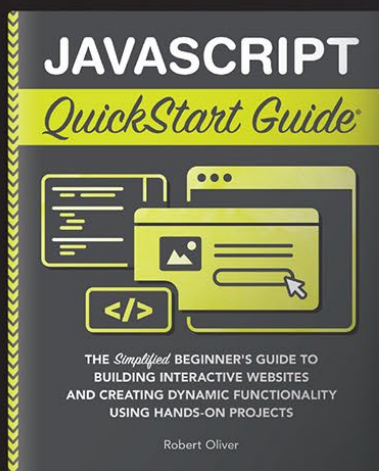


---

# JavaScript Regular Expressions Cheat Sheet

Digital Asset for:



QUICKSTARTGUIDES.COM

## SPECIAL CHARACTERS

<code>^</code>	Matches the expression to its right at the start of a string. It matches every such instance before each <code>\n</code> in the string.
<code>\$</code>	Matches the expression to its left at the end of a string. It matches every such instance before each <code>\n</code> in the string.
<code>.</code>	Matches any character except line terminators like <code>\n</code> .
<code>\</code>	Escapes special characters or denotes character classes.
<code>A B</code>	Matches expression <code>A</code> or <code>B</code> . If <code>A</code> is matched, <code>B</code> is not tried.
<code>+</code>	Greeditly matches the expression to its left 1 or more times.
<code>*</code>	Greeditly matches the expression to its left 0 or more times.
<code>?</code>	Greeditly matches the expression to its left 0 or 1 times. But if <code>?</code> is added to qualifiers ( <code>+</code> , <code>*</code> , and <code>?</code> itself) it will perform matches in a non-greedy manner.
<code>{m}</code>	Matches the expression to its left <code>m</code> times, and not less.
<code>{m,n}</code>	Matches the expression to its left <code>m</code> to <code>n</code> times, and not less.
<code>{m,n}?</code>	Matches the expression to its left <code>m</code> times, and ignores <code>n</code> . See <code>?</code> above.

## CHARACTER CLASSES

<code>\A</code>	Matches the expression to its right at the absolute start of a string whether in single or multi-line mode
<code>\Z</code>	Matches the expression to its left at the absolute end of a string whether in single or multi-line mode
<code>\b</code>	Matches the boundary (or empty string) at the start and end of a word, that is, between <code>\w</code> and <code>\W</code>
<code>\B</code>	Matches where <code>\b</code> does not, that is, the boundary of <code>\w</code> characters
<code>[\b]</code>	Matches backspaces
<code>\d</code>	Matches digits, which means <code>0-9</code>
<code>\D</code>	Matches any non-digits
<code>\f</code>	Matches any formfeeds (i.e., page breaks)
<code>\n</code>	Matches linefeeds (i.e., the end of one line and the beginning of a new line)

<code>\r</code>	Matches carriage returns
<code>\s</code>	Matches whitespace characters, which include the <code>\t</code> , <code>\n</code> , <code>\r</code> , and space characters
<code>\S</code>	Matches non-whitespace characters
<code>\t</code>	Matches horizontal tabs
<code>\v</code>	Matches vertical tabs
<code>\w</code>	Matches alphanumeric characters, which means <code>a-z</code> , <code>A-Z</code> , and <code>0-9</code> . It also matches the underscore, <code>_</code>
<code>\W</code>	Matches non-alphanumeric characters such as <code>%</code> or <code>ó</code>

## SETS

<code>[ ]</code>	Contains a set of characters to match.
<code>[amk]</code>	Matches either <code>a</code> , <code>m</code> , or <code>k</code> . It does not match <code>amk</code> .
<code>[a-z]</code>	Matches any alphabet <code>a</code> to <code>z</code> .
<code>[a\ -z]</code>	Matches <code>a</code> , <code>-</code> , or <code>z</code> . It matches <code>-</code> because <code>\</code> escapes it.
<code>[a-]</code>	Matches <code>a</code> or <code>-</code> , because <code>-</code> is not being used to indicate a series of characters.
<code>[-a]</code>	As above, matches <code>a</code> or <code>-</code> .
<code>[a-z0-9]</code>	Matches characters from <code>a</code> to <code>z</code> and also from <code>0</code> to <code>9</code> .
<code>[(+*)]</code>	Special characters become literal inside a set, so this matches <code>(</code> , <code>+</code> , <code>*</code> , and <code>)</code> .
<code>[^ab5]</code>	Adding <code>^</code> excludes any character in the set. Here, it matches characters that are not <code>a</code> , <code>b</code> , or <code>5</code> .

## GROUPS

<code>( )</code>	Matches the expressions inside the parentheses and groups it.
<code>(?)</code>	Inside parentheses like this, <code>?</code> acts as an extension notation. Its meaning depends on the character immediately to its right.
<code>(?PAB)</code>	Matches the expression <code>AB</code> , and it can be accessed with the group name.

<code>(?aiLmsux)</code>	<p>Here, <code>a</code>, <code>i</code>, <code>L</code>, <code>m</code>, <code>s</code>, <code>u</code>, and <code>x</code> are flags:</p> <ul style="list-style-type: none"> <li><code>a</code> – Matches ASCII only</li> <li><code>i</code> – Ignore case</li> <li><code>L</code> – Locale dependent</li> <li><code>m</code> – Multi-line</li> <li><code>s</code> – Matches all</li> <li><code>u</code> – Matches unicode</li> <li><code>x</code> – Verbose</li> </ul>
<code>(?:A)</code>	Matches the expression as represented by <code>A</code> , but unlike <code>(?PAB)</code> , it cannot be retrieved afterwards
<code>(?#...)</code>	A comment. Contents are for us to read, not for matching.
<code>A(?=B)</code>	Lookahead assertion. This matches the expression <code>A</code> only if it is followed by <code>B</code> .
<code>A(?!B)</code>	Negative lookahead assertion. This matches the expression <code>A</code> only if it is not followed by <code>B</code> .
<code>(?&lt;=B)A</code>	Positive lookbehind assertion. This matches the expression <code>A</code> only if <code>B</code> is immediately to its left. This can only be matched fixed length expressions.
<code>(?&lt;!B)A</code>	Negative lookbehind assertion. This matches the expression <code>A</code> only if <code>B</code> is not immediately to its left. This can only be matched fixed length expressions.
<code>(?P=name)</code>	Matches the expression matched by an earlier group named "name".
<code>(...)\1</code>	The number <code>1</code> corresponds to the first group to be matched. If we want to match more instances of the same expression, simply use its number instead of writing out the whole expression again. We can use from <code>1</code> up to <code>99</code> such groups and their corresponding numbers.