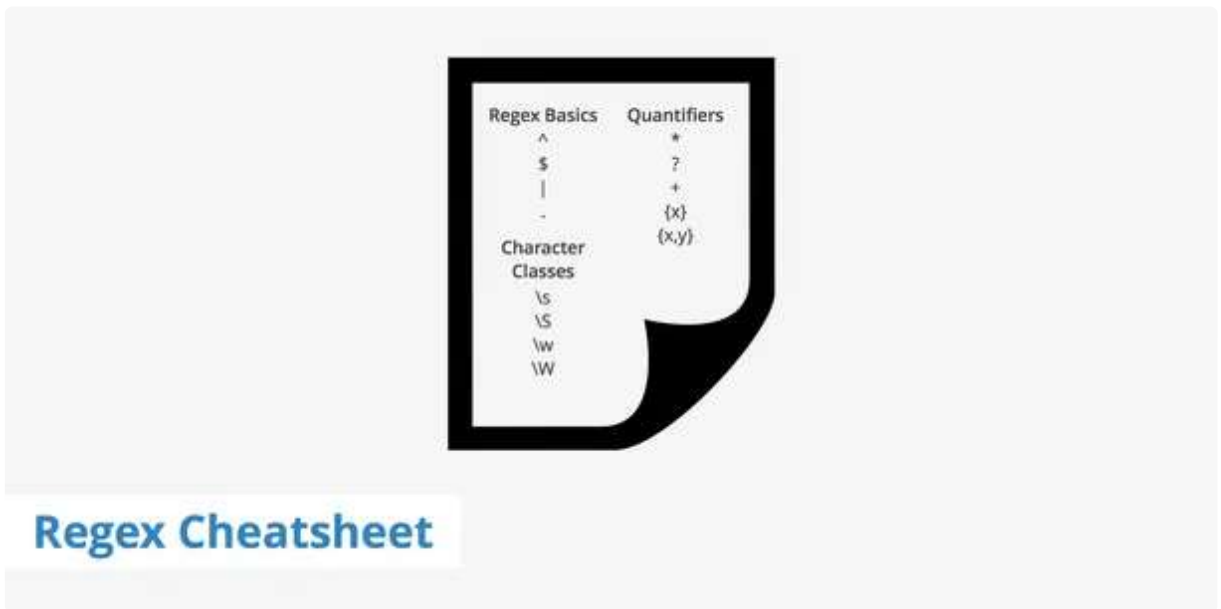
Ultimate Regex Cheat Sheet

Updated on May 7, 2023



Are you tired of spending hours searching for the correct regex pattern for your project? Regex, or regular expressions, is a powerful tool for text manipulation, but it can also be overwhelming and confusing. That's why we've compiled the ultimate regex cheat sheet to help you simplify the process and improve your productivity.

In this article, we'll cover everything from basic syntax to advanced techniques, providing you with a comprehensive guide to regex. So whether you're a beginner or an experienced programmer, keep reading to level up your regex skills.

What is regex?

Regex, also commonly called regular expression, is a combination of characters that **define a particular search pattern**. These expressions can be used for matching a string of text, find and replace operations, data validation, etc. For example, with regex you can easily check a user's input for common misspellings of a particular word.

This guide provides a regex cheat sheet that you can use as a reference when creating regex expressions. We will also go over a couple of popular regex examples and mention a few tools you can use to validate/create your regex expressions.

Regex cheat sheet

Consult the following regex cheat sheet to get a quick overview of what each regex token does within an expression.

REGEX BASICS	DESCRIPTION
^	The start of a string
$	The end of a string
.	Wildcard which matches any character, except newline (\n).
|	Matches a specific character or group of characters on either side (e.g. a|b corresponds to a or b)
\	Used to escape a special character
a	The character "a"
ab	The string "ab"

QUANTIFIERS	DESCRIPTION
*	Used to match 0 or more of the previous (e.g. xy*z could correspond to "xz", "xyz", "xyyz", etc.)
?	Matches 0 or 1 of the previous

QUANTIFIERS	DESCRIPTION
+	Matches 1 or more of the previous
{5}	Matches exactly 5
{5,}	Matches 5 or more occurrences of the preceding character or group
{5, 10}	Matches everything between 5-10

CHARACTER CLASSES	DESCRIPTION
\s	Matches a whitespace character
\S	Matches a non-whitespace character
\w	Matches a word character
\W	Matches a non-word character
\d	Matches one digit
\D	Matches one non-digit
[\b]	A backspace character
\c	A control character

SPECIAL CHARACTERS	DESCRIPTION
\n	Matches a newline
\t	Matches a tab
\r	Matches a carriage return
\ZZZ	Matches octal character ZZZ
\xZZ	Matches hex character ZZ
\0	A null character
\v	A vertical tab

GROUPS	DESCRIPTION
(xyz)	Grouping of characters
(?:xyz)	Non-capturing group of characters
[xyz]	Matches a range of characters (e.g. x or y or z)
[^xyz]	Matches a character other than x or y or z
[a-q]	Matches a character from within a specified range

GROUPS	DESCRIPTION
[0-7]	Matches a digit from within a specified range

STRING REPLACEMENTS	DESCRIPTION
$`	Insert before matched string
$'	Insert after matched string
$+	Insert last matched
$&	Insert entire match
$n	Insert nth captured group

ASSERTIONS	DESCRIPTION
(?=xyz)	Positive lookahead
(?!xyz)	Negative lookahead
?!= or ?<!	Negative lookbehind
\b	Word Boundary (usually a position between /w and /W)
?#	Comment

Regex examples

With the regex cheat sheet above, you can dissect and verify **what each token within a regex expression actually does**. However, you may still be a little confused as to how to put these tokens together to create an expression for a particular purpose. The following section contains a couple of examples that show how you can use regex to match a given string.

1. Matching an email address

To match a particular email address with regex we need to utilize various tokens. The following regex snippet will match a commonly formatted email address.

```
/^([a-z0-9_\.-]+)@([\da-z\.-]+)\.([a-z\.]{2,5})$/
```

The first part of the above regex expression uses an `^` to start the string. Then the expression is broken into three separate groups.

- **Group 1** `([a-z0-9_\.-]+)` - In this section of the expression, we match one or more lowercase letters between a-z, numbers between 0-9, underscores, periods, and hyphens. The expression is then followed by an `@` sign.
- **Group 2** `([\da-z\.-]+)` - Next, the domain name must be matched which can use one or more digits, letters between a-z, periods, and hyphens. The domain name is then followed by a period `\.` .
- **Group 3** `([a-z\.]{2,5})` - Lastly, the third group matches the top level domain. This section looks for any group of letters or dots that are 2-5 characters long. This can also account for region-specific top level domains.

Therefore, with the regex expression above you can match many of the commonly used emails such as `firstname.lastname@domain.com` for example.

2. Matching a phone number

To use regex in order to search for a particular phone number we can use the following expression.

```
/^\b\d{3}[-.]?\d{3}[-.]?\d{4}\b$/
```

This expression is somewhat similar to the email example above as it is broken into 3 separate sections. Once again, to start off the expression, we begin with `^.`

- **Section 1** `\b\d{3}` - This section begins with a word boundary to tell regex to match the alpha-numeric characters. It then matches 3 of any digit between 0-9 followed by either a hyphen, a period, or nothing `[-.]?`.
- **Section 2** `\d{3}` - The second section is quite similar to the first section, it matches 3 digits between 0-9 followed by another hyphen, period, or nothing `[-.]?`.
- **Section 3** `\d{4}\b` - Lastly, this section is slightly different in that it matches 4 digits instead of three. The word boundary assertion is also used at the end of the expression. Finally, the end of the string is defined by the `$`.

Therefore, with the above regex expression for finding phone numbers, it would identify a number in the format of 123-123-1234, 123.123.1234, or 1231231234.

Regex tools