Quick-Start: Regex Cheat Sheet

The tables below are a reference to basic regex. While reading the rest of the site, when in doubt, you can always come back and look here. (It you want a bookmark, here's a direct link to the regex reference tables). I encourage you to print the tables so you have a cheat sheet on your desk for quick reference.

The tables are not exhaustive, for two reasons. First, every regex flavor is different, and I didn't want to crowd the page with overly exotic syntax. For a full reference to the particular regex flavors you'll be using, it's always best to go straight to the source. In fact, for some regex engines (such as Perl, PCRE, Java and .NET) you may want to check once a year, as their creators often introduce new features.

The other reason the tables are not exhaustive is that I wanted them to serve as a quick introduction to regex. If you are a complete beginner, you should get a firm grasp of basic regex syntax just by reading the examples in the tables. I tried to introduce features in a logical order and to keep out oddities that I've never seen in actual use, such as the "bell character". With these tables as a jumping board, you will be able to advance to mastery by exploring the other pages on the site.

How to use the tables

The tables are meant to serve as an accelerated regex course, and they are meant to be read slowly, one line at a time. On each line, in the leftmost column, you will find a new element of regex syntax. The next column, "Legend", explains what the element means (or encodes) in the regex syntax. The next two columns work hand in hand: the "Example" column gives a valid regular expression that uses the element, and the "Sample Match" column presents a text string that could be matched by the regular expression.

You can read the tables online, of course, but if you suffer from even the mildest case of online-ADD (attention deficit disorder), like most of us… Well then, I highly recommend you print them out. You'll be able to study them slowly, and to use them as a cheat sheet later, when you are reading the rest of the site or experimenting with your own regular expressions.

Enjoy!

If you overdose, make sure not to miss the next page, which comes back down to Earth and talks about some really cool stuff: **The 1001 ways to use Regex**.

Regex Accelerated Course and Cheat Sheet

For easy navigation, here are some jumping points to various sections of the page:

(direct link)

Characters

Character	Legend	Example	Sample Match
\d	Most engines: one digit from 0 to 9	file_\d\d	file_25
\d	.NET, Python 3: one Unicode digit in any script	file_\d\d	file_9৩
\w	Most engines: "word character": ASCII letter, digit or underscore	\w-\w\w\w	A-b_1
\w	.Python 3: "word character": Unicode letter, ideogram, digit, or underscore	\w-\w\w\w	字-ま_۳
\w	.NET: "word character": Unicode letter, ideogram, digit, or connector	\w-\w\w\w	字-ま‿۳
\s	Most engines: "whitespace character": space, tab, newline, carriage return, vertical tab	a\sb\sc	a b c
\s	.NET, Python 3, JavaScript: "whitespace character": any Unicode separator	a\sb\sc	a b c
\D	One character that is not a *digit* as defined by your engine's *\d*	\D\D\D	ABC
\W	One character that is not a *word character* as defined by your engine's *\w*	\W\W\W\W\W	*-+=)
\S	One character that is not a *whitespace character* as defined by your engine's *\s*	\S\S\S\S	Yoyo

(direct link)

Quantifiers

Quantifier	Legend	Example	Sample Match
+	One or more	Version \w-\w+	Version A-b1_1
{3}	Exactly three times	\D{3}	ABC
{2,4}	Two to four times	\d{2,4}	156
{3,}	Three or more times	\w{3,}	regex_tutorial
*	Zero or more times	A*B*C*	AAACC
?	Once or none	plurals?	plural

(direct link)

More Characters

Character	Legend	Example	Sample Match
.	Any character except line break	a.c	abc
.	Any character except line break	.*	whatever, man.
\.	A period (special character: needs to be escaped by a \)	a\.c	a.c
\	Escapes a special character	\.*\+\? \$\^\/\\	.*+? $^/\
\	Escapes a special character	\[\{\(\)\}\]	[{()}]

(direct link)

Logic

Logic	Legend	Example	Sample Match
|	Alternation / OR operand	22|33	33
(…)	Capturing group	A(nt|pple)	Apple (captures "pple")
\1	Contents of Group 1	r(\w)g\1x	regex
\2	Contents of Group 2	(\d\d)\+(\d\d)=\2\+\1	12+65=65+12
(?: …)	Non-capturing group	A(?:nt|pple)	Apple

(direct link)

More White-Space

Character	Legend	Example	Sample Match
\t	Tab	T\t\w{2}	T ab
\r	Carriage return character	see below	
\n	Line feed character	see below	
\r\n	Line separator on Windows	AB\r\nCD	AB CD
\N	Perl, PCRE (C, PHP, R…): one character that is not a line break	\N+	ABC
\h	Perl, PCRE (C, PHP, R…), Java: one horizontal whitespace character: tab or Unicode space separator		
\H	One character that is not a horizontal whitespace		
\v	.NET, JavaScript, Python, Ruby: vertical tab		
\v	Perl, PCRE (C, PHP, R…), Java: one vertical whitespace character: line feed, carriage return, vertical tab, form feed, paragraph or line separator		
\V	Perl, PCRE (C, PHP, R…), Java: any character that is not a vertical whitespace		
\R	Perl, PCRE (C, PHP, R…), Java: one line break (carriage return + line feed pair, and all the characters matched by \v)		

(direct link)

More Quantifiers

Quantifier	Legend	Example	Sample Match
+	The + (one or more) is "greedy"	\d+	12345
?	Makes quantifiers "lazy"	\d+?	1 in **1**2345
*	The * (zero or more) is "greedy"	A*	AAA
?	Makes quantifiers "lazy"	A*?	empty in AAA
{2,4}	Two to four times, "greedy"	\w{2,4}	abcd
?	Makes quantifiers "lazy"	\w{2,4}?	ab in **ab**cd

(direct link)

Character Classes

Character	Legend	Example	Sample Match
[…]	One of the characters in the brackets	[AEIOU]	One uppercase vowel
[…]	One of the characters in the brackets	T[ao]p	*Tap* or *Top*
-	Range indicator	[a-z]	One lowercase letter
[x-y]	One of the characters in the range from x to y	[A-Z]+	GREAT
[…]	One of the characters in the brackets	[AB1-5w-z]	One of either: A,B,1,2,3,4,5,w,x,y,z
[x-y]	One of the characters in the range from x to y	[-~]+	Characters in the printable section of the ASCII table.
[^x]	One character that is not x	[^a-z]{3}	A1!
[^x-y]	One of the characters **not** in the range from x to y	[^ -~]+	Characters that are **not** in the printable section of the ASCII table.
[\d\D]	One character that is a digit or a non-digit	[\d\D]+	Any characters, including new lines, which the regular dot doesn't match
[\x41]	Matches the character at hexadecimal position 41 in the ASCII table, i.e. A	[\x41-\x45]{3}	ABE

(direct link)

Anchors and Boundaries

Anchor	Legend	Example	Sample Match
^	Start of string or start of line depending on multiline mode. (But when [^inside brackets], it means "not")	^abc .*	abc (line start)
$	End of string or end of line depending on multiline mode. Many engine-dependent subtleties.	.*? the end$	this is the end
\A	Beginning of string (all major engines except JS)	\Aabc[\d\D]*	abc (string... ...start)
\z	Very end of the string Not available in Python and JS	the end\z	this is...\n...**the end**
\Z	End of string or (except Python) before final line break Not available in JS	the end\Z	this is...\n...**the end**\n
\G	Beginning of String or End of Previous Match .NET, Java, PCRE (C, PHP, R…), Perl, Ruby		
\b	Word boundary Most engines: position where one side only is an ASCII letter, digit or underscore	Bob.*\bcat\b	Bob ate the cat
\b	Word boundary .NET, Java, Python 3, Ruby: position where one side only is a Unicode letter, digit or underscore	Bob.*\b\кошка\b	Bob ate the кошка
\B	Not a word boundary	c.*\Bcat\B.*	copycats

(direct link)

POSIX Classes

Character	Legend	Example	Sample Match
[:alpha:]	PCRE (C, PHP, R…): ASCII letters A-Z and a-z	[8[:alpha:]]+	WellDone88
[:alpha:]	Ruby 2: Unicode letter or ideogram	[[:alpha:]\d]+	кошка99
[:alnum:]	PCRE (C, PHP, R…): ASCII digits and letters A-Z and a-z	[[:alnum:]]{10}	ABCDE12345
[:alnum:]	Ruby 2: Unicode digit, letter or ideogram	[[:alnum:]]{10}	кошка90210
[:punct:]	PCRE (C, PHP, R…): ASCII punctuation mark	[[:punct:]]+	?!.,:;
[:punct:]	Ruby: Unicode punctuation mark	[[:punct:]]+	¿.: ~\¡

(direct link)

Inline Modifiers

None of these are supported in JavaScript. In Ruby, beware of (?s) and (?m).

Modifier	Legend	Example	Sample Match
(?i)	Case-insensitive mode (except JavaScript)	(?i)Monday	monDAY
(?s)	DOTALL mode (except JS and Ruby). The dot (.) matches new line characters (\r\n). Also known as "single-line mode" because the dot treats the entire input as a single line	(?s)From A.*to Z	From A to Z
(?m)	Multiline mode (except Ruby and JS) ^ and $ match at the beginning and end of every line	(?m)1\r\n^2$\r\n^3$	1 2 3
(?m)	In Ruby: the same as (?s) in other engines, i.e. DOTALL mode, i.e. dot matches line breaks	(?m)From A.*to Z	From A to Z
(?x)	Free-Spacing Mode mode (except JavaScript). Also known as comment mode or whitespace mode	(?x) # this is a # comment abc # write on multiple # lines [ ]d # spaces must be # in brackets	abc d

(?n)	.NET, PCRE 10.30+: named capture only	Turns all (parentheses) into non-capture groups. To capture, use named groups.
(?d)	Java: Unix linebreaks only	The dot and the ^ and $ anchors are only affected by \n
(?^)	PCRE 10.32+: unset modifiers	Unsets ismnx modifiers

(direct link)

Lookarounds

Lookaround	Legend	Example	Sample Match
(?=…)	Positive lookahead	(?=\d{10})\d{5}	01234 in **01234**56789
(?<=…)	Positive lookbehind	(?<=\d)cat	cat in 1**cat**
(?!…)	Negative lookahead	(?!theatre)the\w+	theme
(?<!…)	Negative lookbehind	\w{3}(?<!mon)ster	Munster

(direct link)

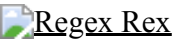
Character Class Operations

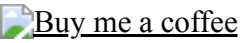
Class Operation	Legend	Example	Sample Match
[…-[…]]	.NET: character class subtraction. One character that is in those on the left, but not in the subtracted class.	[a-z-[aeiou]]	Any lowercase consonant
[…-[…]]	.NET: character class subtraction.	[\p{IsArabic}-[\D]]	An Arabic character that is not a non-digit, i.e., an Arabic digit
[…&&[…]]	Java, Ruby 2+: character class intersection. One character that is both in those on the left and in the && class.	[\S&&[\D]]	An non-whitespace character that is a non-digit.
[…&&[…]]	Java, Ruby 2+: character class intersection.	[\S&&[\D]&&[^a-zA-Z]]	An non-whitespace character that a non-digit and not a letter.
[…&&[^…]]	Java, Ruby 2+: character class subtraction is obtained by intersecting a class with a negated class	[a-z&&[^aeiou]]	An English lowercase letter that is not a vowel.
[…&&[^…]]	Java, Ruby 2+: character class subtraction	[\p{InArabic}&&[^\p{L}\p{N}]]	An Arabic character that is not a letter or a number

(direct link)

Other Syntax

Syntax	Legend	Example	Sample Match
\K	Keep Out Perl, PCRE (C, PHP, R…), Python's alternate *regex* engine, Ruby 2+: drop everything that was matched so far from the overall match to be returned	prefix\K\d+	12
\Q…\E	Perl, PCRE (C, PHP, R…), Java: treat anything between the delimiters as a literal string. Useful to escape metacharacters.	\Q(C++ ?)\E	(C++ ?)


Ask Rex



Leave a Comment
1-10 of 19 Threads
Appu – Japan
March 07, 2022 - 19:05
Subject: You are God of regex !! Thank you so much :)

This site is absolute gold mine. I once stumbled upon and missed it, now found again… So happy :D Thank you so much for all your efforts!!
maureen – san francisco
June 18, 2021 - 16:25
Subject: absolutely the BEST website for regex

This is the go-to website for everything on regex. Thank you!
Pythia – New Zealand
July 15, 2020 - 03:54
Subject: Very thoughtful and useful cheat sheet

Unlike lots of other cheat sheets or regex web sites, I was able (without much persistent regex knowledge) to apply the rules and to solve my problem. THANK YOU :)
Mark
July 04, 2020 - 10:14
Subject: Thanks a lot

Thanks a lot for the quick guide. It's really helpful.
Purusharth Amrut
June 10, 2020 - 14:41
Subject: Very useful site

Thank you soooooo much for this site. I'm using python regex for natural language processing in sentiment analysis and this helped me a lot.
Alessandro Maiorana – Italy, Milan
April 15, 2020 - 12:43
Subject: Thank you! Excellent resource for any student

Thank you so much for this incredible cheatsheet! It is facilitating a lot my regex learning! God bless you and your passion!
michael – Bulgaria
April 10, 2020 - 12:43
Subject: Thank you for doing such a geat work.

I am now learning regex and for finding such a well organized site is a blessing! You are a good soul! Thank you for everything and stay inspired!
Yuri – California
November 13, 2019 - 17:39
Subject: Simple = perfect

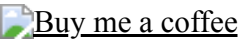
Thanks a lot, saved me tons of time!!!!
Tom – Europe, Poland
September 30, 2019 - 18:43
Subject: Congratulations

Well done, very useful page. Thank you for your effort. T
Najam
March 25, 2019 - 03:44
Subject: Thank you very much

Hi Rex,

Thankyou very much for compiling these. I am new to text analytics and is struggling a lot with regex. This is helping me a lot pick up. Great work


Next


Buy me a coffee