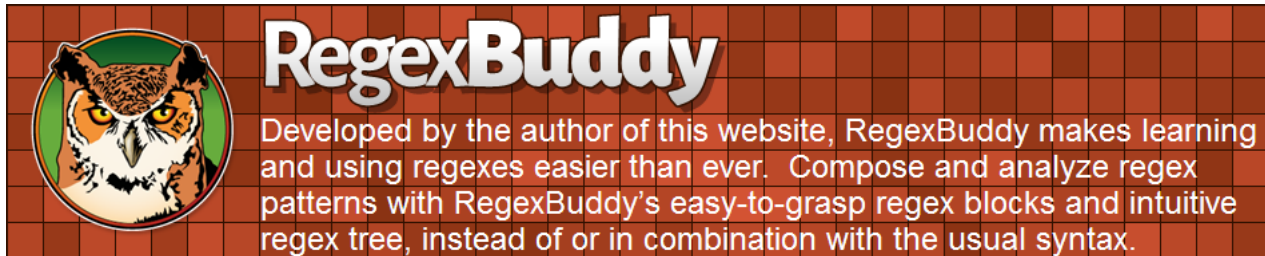




	Quick Start Examples	Tutorial Reference	Tools & Languages Book Reviews	
--	-------------------------	-----------------------	-----------------------------------	--



Regular Expressions Tutorial

Learn How to Use and Get The Most out of Regular Expressions

This tutorial teaches you all you need to know to be able to craft powerful time-saving regular expressions. It starts with the most basic concepts, so that you can follow this tutorial even if you know nothing at all about regular expressions yet.

The tutorial doesn't stop there. It also explains how a regular expression engine works on the inside and alerts you to the consequences. This helps you to quickly understand why a particular regex does not do what you initially expected. It will save you lots of guesswork and head scratching when you need to write more complex regexes.

What Regular Expressions Are Exactly - Terminology

Basically, a regular expression is a pattern describing a certain amount of text. Their name comes from the mathematical theory on which they are based. But we will not dig into that. You will usually find the name abbreviated to "regex" or "regexp". This tutorial uses "regex", because it is easy to pronounce the plural "regexes". On this website, regular expressions are shaded gray as `regex`.

This first example is actually a perfectly valid regex. It is the most basic pattern, simply matching the literal text `regex`. A "match" is the piece of text, or sequence of bytes or characters that pattern was found to correspond to by the regex processing software. Matches are highlighted in blue on this site.

`\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}\b` is a more complex pattern. It describes a series of letters, digits, dots, underscores, percentage signs and hyphens, followed by an at sign, followed by another series of letters, digits and hyphens, finally followed by a single dot and two or more letters. In other words: this pattern describes an [email address](#). This also shows the syntax highlighting applied to regular expressions on this site. Word boundaries and quantifiers are blue, character classes are orange, and escaped literals are gray. You'll see additional colors like green for grouping and purple for meta tokens later in the tutorial.

With the above regular expression pattern, you can search through a text file to find email addresses, or verify if a given string looks like an email address. This tutorial uses the term "string" to indicate the text that the regular expression is applied to. This website highlights them in green. The term "string" or "character string" is used by programmers to indicate a sequence of characters. In practice, you can use regular expressions with whatever data you can access using the application or programming language you are

working with.

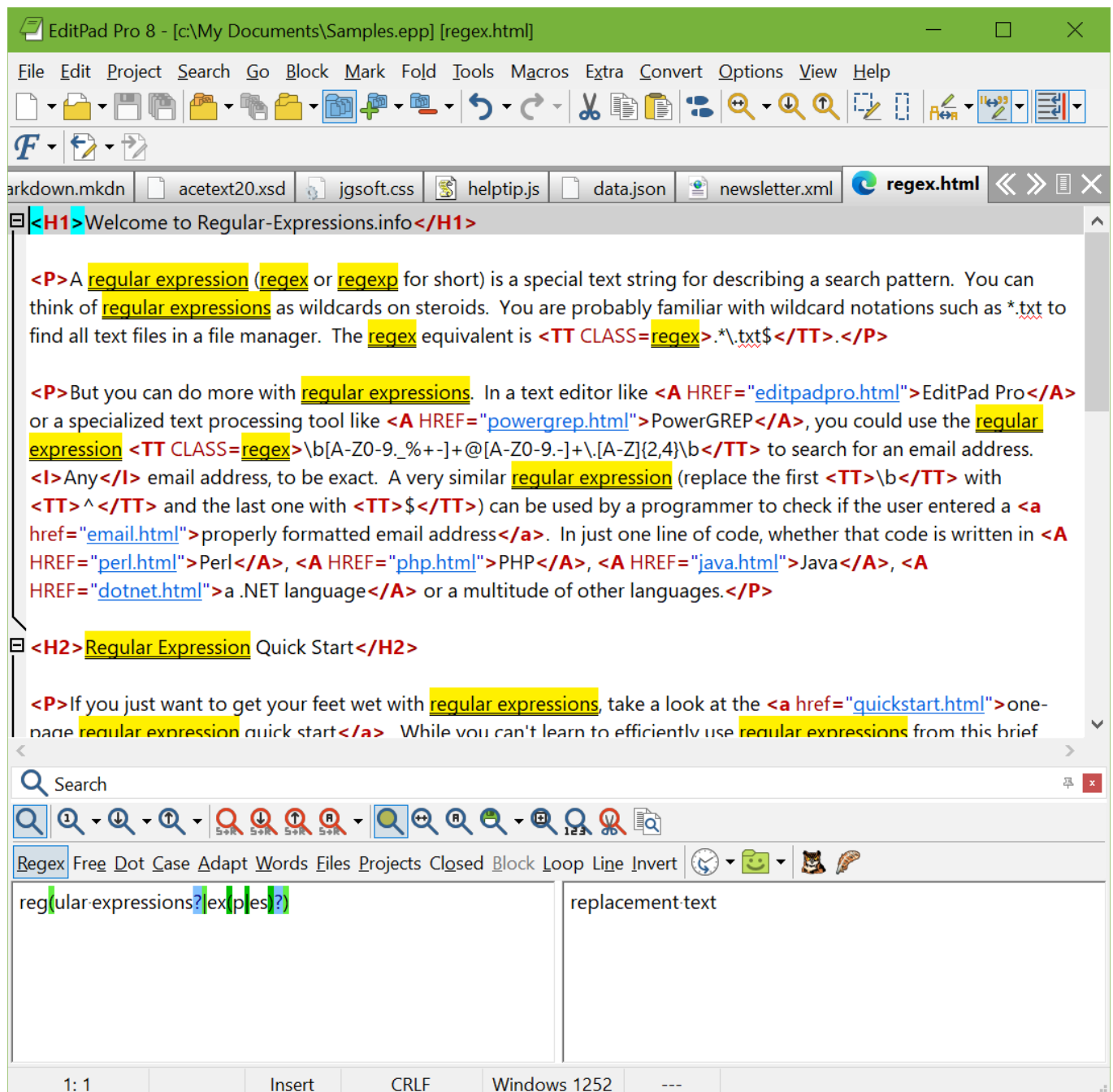
Different Regular Expression Engines

A regular expression “engine” is a piece of software that can process regular expressions, trying to match the pattern to the given string. Usually, the engine is part of a larger application and you do not access the engine directly. Rather, the application invokes it for you when needed, making sure the right regular expression is applied to the right file or data.

As usual in the software world, different regular expression engines are not fully compatible with each other. The syntax and behavior of a particular engine is called a regular expression flavor. This tutorial covers all the popular regular expression flavors, including [Perl](#), [PCRE](#), [PHP](#), [.NET](#), [Java](#), [JavaScript](#), [XRegExp](#), [VBScript](#), [Python](#), [Ruby](#), [Delphi](#), [R](#), [Tcl](#), [POSIX](#), and [many others](#). The tutorial alerts you when these flavors require different syntax or show different behavior. Even if your application is not explicitly covered by the tutorial, it likely uses a regex flavor that is covered, as most applications are developed using one of the programming environments or regex libraries just mentioned.

Give Regexes a First Try

You can easily try the following yourself in a text editor that supports regular expressions, such as [EditPad Pro](#). If you do not have such an editor, you can [download the free evaluation version of EditPad Pro](#) to try this out. EditPad Pro’s regex engine is fully functional in the demo version.



As a quick test, copy and paste the text of this page into EditPad Pro. Then select Search|Multiline Search Panel in the menu. In the search panel that appears near the bottom, type in **regex** in the box labeled "Search Text". Mark the "Regular expression" checkbox, and click the Find First button. This is the leftmost button on the search panel. See how EditPad Pro's regex engine finds the first match. Click the Find Next button, which sits next to the Find First button, to find further matches. When there are no further matches, the Find Next button's icon flashes briefly.

Now try to search using the regex **reg(ular expressions?|ex(p|es)?)**. This regex finds all names, singular and plural, I have used on this page to say "regex". If we only had plain text search, we would have needed 5 searches. With regexes, we need just one search. Regexes save you time when using a tool like EditPad Pro. Select Count Matches in the Search menu to see how many times this regular expression can match the file you have open in EditPad Pro.

If you are a programmer, your software will run faster since even a simple regex engine applying the above regex once will outperform a state of the art plain text search algorithm searching through the data five times. Regular expressions also reduce development time. With a regex engine, it takes only one line (e.g. in Perl, PHP, Python, Ruby, Java, or .NET) or a couple of lines (e.g. in C using PCRE) of code to, say, check if the user's input looks like a [valid email address](#).

[Regex Tutorial Table of Contents](#)

| [Quick Start](#) | [Tutorial](#) | [Tools & Languages](#) | [Examples](#) | [Reference](#) | [Book Reviews](#) |
| [Introduction](#) | [Table of Contents](#) | [Special Characters](#) | [Non-Printable Characters](#) |
[Regex Engine Internals](#) | [Character Classes](#) | [Character Class Subtraction](#) | [Character Class Intersection](#)
| [Shorthand Character Classes](#) | [Dot](#) | [Anchors](#) | [Word Boundaries](#) | [Alternation](#) | [Optional Items](#) |
[Repetition](#) | [Grouping & Capturing](#) | [Backreferences](#) | [Backreferences, part 2](#) | [Named Groups](#) |
[Relative Backreferences](#) | [Branch Reset Groups](#) | [Free-Spacing & Comments](#) | [Unicode](#) |
[Mode Modifiers](#) | [Atomic Grouping](#) | [Possessive Quantifiers](#) | [Lookahead & Lookbehind](#) |
[Lookaround, part 2](#) | [Keep Text out of The Match](#) | [Conditionals](#) | [Balancing Groups](#) | [Recursion](#) |
[Subroutines](#) | [Infinite Recursion](#) | [Recursion & Quantifiers](#) | [Recursion & Capturing](#) |
[Recursion & Backreferences](#) | [Recursion & Backtracking](#) | [POSIX Bracket Expressions](#) |
[Zero-Length Matches](#) | [Continuing Matches](#) |

Page URL: <https://www.regular-expressions.info/tutorial.html>

Page last updated: 19 August 2021

Site last updated: 17 March 2023

Copyright © 2003-2023 Jan Goyvaerts. All rights reserved.