Why do people defend the regex syntax? [closed]

Asked 16 years, 1 month ago Modified 12 years, 6 months ago

Viewed 3k times

11

Closed. This question is opinion-based. It is not currently accepting answers.

Want to improve this question? Update the question so it can be answered with facts and citations by editing this post.

Closed 11 years ago.

Improve this question

There is a similar question going around, but it just got the same old answers that people always give about Regex syntax, but that's not the point here, so please try to not knee jerk the same old answers about regex syntax. Try to be a little more original and personal about it this time.

Regex syntax is very VERY compact, almost too compact to be good. Its like code-golfing, and everyone agrees code-golfing isn't a good thing in production code. However most people accept regex syntax, which seems... contradictory to say the least.

So now some common defenses one is likely to hear include :

- *Answer*: It's compact
- *Counter*: Haven't we all agreed in this day and age that code should be literate and a variable like "client" is better than "c"?
- *Answer*: It's a "domain specific language"
- *Counter*: how about all the very easy to understand, non compact, non cryptic and dare I say pretty domain languages out there like SQL or LINQ?
- *Answer*: Its easy to understand once you know it.
- *Counter*: Most great languages are easy to understand even if you've never used them before. For example anyone could jump into Python very easily even if they had never seen it before. And why do people defend Regex when its such a hard language to look at, but then go on and complain about Lisps parenthesis?

Ok now everyone try to be original and honest here, don't just pull out the same old rote answers programmers used 20 years ago to design regex. Unless you really believe they are valid propositions in this day and age.

Edit: For the record, I know Regex from years ago, use them frequently even today and might even grok them. However I suddenly had a feeling it was perhaps time to reconsider things I had taken as "truths" about regex, and look at them from a modern standpoint. Mostly because questioning principles is necessary for further development, and because so many newcomers complain vehemently about them, they can't just be flat out right, so I decided to try to step into the shoes of a newcomer and consider what are some good points against regex.

As for being **subjective**, I don't think this is less subjective OR less programmer related than **Programmer Jokes** of the days stuff. On the contrary it is very programmer related.

As for **argumentative**, thats the point of the question. To get good arguments pro and con regex outdated syntax, that can serve newcomers to actually understand more about why regex are what they are, and even better hopefully get some newcomer to come up with a better solution US old minded can't see because we are blinded by the "coolness" of regex.

Quote:

> The Perl 5.10 documentation for regexes has melted down into a heap of unreadable drivel because so many zany features have creeped into the syntax that no-one can write sensible documentation for it any more.

You're trying to say regex have become unmaintainable? Well then as good programmers should we consider refactoring them? Maybe cleaning up and trying over as we've done with some many other technologies?

regex syntax

Share

Improve this question

Follow

edited Nov 7, 2008 at 23:06

brian d foy
132k ● 31 ● 211 ● 604

asked Nov 5, 2008 at 3:56

Robert Gould
69.7k ● 61 ● 191 ● 275

1 An indication if you *know* regex would also be nice. By *know*, I mean something on the level of being able to distinguish an NFA from a DFA. – Tomalak Nov 5, 2008 at 6:18

please close this question, not related to programming. – zamfir Nov 5, 2008 at 7:02

3 SQL a pretty domain language? – Cheery Nov 5, 2008 at 7:16

2 Zamfir this is not, not related with programming. It is totally about programming and in many senses. There is no need to declare anything that doesn't solve an immediate problem to not be acceptable, because SO is full of similar questions, and the community has in general accepted things like this – Robert Gould Nov 5, 2008 at 9:09

@Cheery, well I work with SQL a lot and just and I've come to find it pretty. But you are probably right its not really pretty when I think about it, that just so you see how sick we programmers are :) – Robert Gould Nov 5, 2008 at 9:11

19 Answers

Sorted by: Highest score (default)

36

Most of what I have to say were addressed by Adam and DGM, but I don't think they cover your second point very well.

"how about all the very easy to understand, non compact, non cryptic and dare I say pretty domain languages out there like SQL or LINQ?"

I think a good way to express an answer to this is to ask, how would you use English to explain a regular expression?

```
<TAG\b[^>]*>(.*?)</TAG>
```

Look for "<TAG" a word boundary zero or more of something that is not '>' followed by a '>' remember zero or more of something, stopping at the first "</TAG>"

This is a fairly simple regex. Is the English form really easier to understand? Could you do better?

Regular expressions are hard to read, but what you want from them can be just as hard to explain.

Share Improve this answer answered Nov 5, 2008 at 5:04

Follow

he_the_great
6,764 ● 2 ● 33 ● 28

6 +1: Good points. "If it was hard to write, it should be hard to understand. Why d'ya think they call it *code*!?" – Adam Liss Nov 5, 2008 at 5:26

+1 from here too. Thanks for making a good point there! – Robert Gould Nov 5, 2008 at 6:06

Now, if you hadn't explained that it would have made no sense at all! – RCIX Sep 5, 2009 at 12:37

2 @RCIX That is only if you don't know regex. – he_the_great Sep 6, 2009 at 22:50

To play devil's advocate here; how would you replace this code with a elegant non-regex alternative? – Rebecca Nov 30, 2011 at 9:53

28

Look at the other side of the question: how would you design a new syntax that embodies all the features, consistency, conciseness, and robustness as regex, but is more programmer-friendly?

Share Improve this answer

Follow

answered Nov 5, 2008 at 4:00

Adam Liss
48.3k ● 13 ● 113 ● 152

1 I'm not smart enough to think of a better solution, but I sure hope someone smart did. – Robert Gould Nov 5, 2008 at 6:00

Brad Gilbert, I was going to say that. – Axeman Nov 5, 2008 at 15:01

1 I don't think conciseness is a virtue when it hinders understanding. Here's an attempt to do BNF in C++: boost.org/doc/libs/1_35_0/libs/spirit/index.html – Mark Ransom Nov 5, 2008 at 16:34

Unless I've missed something Perl6 grammars won't do much of anything to improve the syntax of regexes. They will however do a great deal to improve the *use* of them for programmers. – Michael Carman Nov 6, 2008 at 1:11

19

Your counter-arguments are specious. Do you know regex syntax, or are you arguing from a point of ignorance? It's an important point to establish your bias.

- It's not at all like code golfing. I'm not sure of your connection there. Why not complain about pointers or something else using the same argument?
- Regular expression's compactness has nothing to do with poor variable names. A variable named c could be anything. The regex syntax is neither ambiguous nor vague. It exactly describes its pattern.
- It's a DSL. So what if it is? Have you ever tried to do complex things in SQL? It's a big mess too. Making the same thing require more typing and more syntax doesn't improve the situation. Most people I teach have problems with regexes because they are not used to thinking in and designing patterns, not because the syntax is exotic.

- It's easy to understand once you know it. Well, it is. Power tools are not optimized for newbies or for people unwilling to learn. I don't complain about Lisp parenthesis, but I do not mind the regex syntax.

If you don't want to use regular expressions, then don't. Use the string manipulation functions or parsers. Use some other tool. While you're busy with that, I'll be ten problems ahead of you because I'm not swimming against the tide or blaming the tools for the work I can't get done.

It's up to you how much work you want to get done. Find the tool that gets you there the fastest and learn it. If you don't like that, invent something better. Until then, stop complaining.

Share Improve this answer Follow

edited Apr 22, 2010 at 13:48

answered Nov 5, 2008 at 4:29

brian d foy
132k ● 31 ● 211 ● 604

1 Sorry for not being clear, Yes I know them, use them and even dare say like them. However I'm trying to get good arguments against my counter arguments, and yours are true to a degree. – Robert Gould Nov 5, 2008 at 6:09

It is a *little* like golfing. e.g. "\d" is golfed compared to the POSIX "[[:digit:]]" notation. That said, I'd hate the see a verbose form for the "*?" quantifier. – Michael Carman Nov 6, 2008 at 0:36

1 I don't think it's like golfing at all. `\d` isn't there to obfuscate or use a feature in a clever, unintended, or surprising fashion. The POSIX character classes aren't synonyms for Perl's character classes, and in some cases they are different. effectiveperlprogramming.com/blog/991 – brian d foy Mar 14, 2012 at 4:02

The second bullet point, about the variable named `c`, is quite powerful. – Ray Toal Jul 11, 2016 at 21:46

17

It's actually a conspiracy perpetuated by the American Association of Retired Programmers against today's young whippersnappers who cut their teeth on Python and Java. We need to maintain a sense of awe and respect for the mystics whose cleverness surmounted the challenges of tiny core memories and arcane languages with 3-character mnemonics ... and *liked* it. Uphill ... both ways ... in the snow. :-)

Share Improve this answer Follow

answered Nov 5, 2008 at 4:28

community wiki
Adam Liss

Ok, that is a VERY GOOD answer lol :) – Robert Gould Nov 5, 2008 at 6:10

I'd defend regex syntax because it (roughly) matches the notation that I learned when I took my Algorithms &

11

Machines course. It's an easy way to generate a machine to ingest the specified regular language.

The regex syntax is the way it is because it's really all that you need to fully describe the behaviour you're looking for.

Share Improve this answer Follow

answered Nov 5, 2008 at 4:14

Tony Arkles
1,946 ● 13 ● 14

+1 That is a good point I had forgotten about, and does give a valid precedence for the strange syntax that goes beyond computational limits of decades ago. – Robert Gould Nov 5, 2008 at 6:07

9

it works!!
if there is a easy-to-read language that has extensions in every major programming language and is well documented and tested and not as compressed as regex, but neither too verbose (verbose = irritating), i would love to know more about it

Share Improve this answer Follow

answered Nov 5, 2008 at 4:14

Aditya Mukherji
9,246 ● 5 ● 46 ● 50

Yes this is a very good point in favor of Regex! I totally agree with you – Robert Gould Nov 5, 2008 at 5:59

8

Some of the problem with regexes isn't the language itself but what people try to use them for. They will write lines and lines of regexes when what they really want is a fairly simple parser.

Regexes are great for simple to moderately complicated substring matching and data extraction. But at some point of complexity you have just got to whip out the compiler-compiler and write a real parser. I think a lot of people don't realize that regexes are primarily for matching, not for doing parsing.

Share Improve this answer Follow

answered Nov 5, 2008 at 5:26
user21714
5,921 ● 1 ● 22 ● 26

Hear, hear! Regexes are one of my favorite tools, but they are not the appropriate tool for even 10% of what I do.
– Dave Sherohman Nov 5, 2008 at 15:37

8

Others have hinted at this but it bears stating explicitly:

Regular languages are not like programming languages. They're closer to mathematical notation.

The compactness and quirkiness is more a result of attempting to coerce a precise notation out of ASCII characters than a deliberate attempt at brevity or obfuscation.

Share Improve this answer

Follow

answered Nov 6, 2008 at 1:03

Michael Carman

30.8k ● 10 ● 79 ● 124

7

You should be looking at regular expressions as high-end power tools (and I mean power tools in the construction industry sense).

If you're building a little work table for your shed, you don't pull out the nail gun, chainsaw and industrial router. You use a saw, some nails and a hammer.

Similarly, you don't build a 30-floor building without a crane in there somewhere.

The idea is to use the right tool for the job **AND** the right tool for your skill level.

If you have to cut down a tree, make sure you know all about kickback before you start up the chainsaw. If you don't, then use a hand saw instead and save yourself a trip to the hospital to re-attach your severed limb.

I use regular expressions the way I use my chainsaw - very carefully. If you're uncomfortable with the tool, don't use it. Once you learn how to use it properly, you'll find it a lot easier to get things done faster.

Share Improve this answer

Follow

answered Nov 5, 2008 at 4:18

paxdiablo

880k ● 241 ● 1.6k ● 2k

1 +1 they are power-tools in a sense. But maybe we should make safer tools? – Robert Gould Nov 5, 2008 at 6:10

7

I think an SQL-like regular expression language would be a fascinating project. I'd love to see someone create that.

Why *not* have a language where you can write

`LOOK FOR "<TAG"`

`THEN WORDBOUNDARY THEN ZERO-OR-MORE NOT('>') FOLLOWED-BY '>'`

`THEN ZERO-OR-MORE SOMETHING REMEMBERED`

`THEN NEAREST "</TAG>"`

I'm not sure who's the target audience though -- I don't think I would use it, because I've spent all this time learning regexes.

Surely the set of 'people who need to use an expression as complex as that' pretty much maps to the set of programmers who have to deal with stuff just as complex and more on a daily basis?

Share Improve this answer Follow

answered Nov 5, 2008 at 5:43

AmbroseChapel
12.1k ● 7 ● 49 ● 69

1 +1 Agreed I'd be really happy to have something like this. After all even a new syntax if good can catch really fast. A great example of that would be LINQ, and Ruby. Both caught on very fast (less than a year) thanks to being pretty – Robert Gould Nov 5, 2008 at 6:12

2 Urrgh - sorry but that looks horrible. I would say use the whitespace/comments syntax so you can split it up logically, and comment anything likely to be found confusing by anyone maintaining the code (including yourself a year down the line ;) – Hamish Downer Nov 5, 2008 at 16:33

Ooh now that would be nice. a form of SQL (String Query Language)? :D – RCIX Sep 5, 2009 at 12:39

1 @JimGrisham, such tool for explaining regexes already exists, even though it doesn't translate back and forth: regexr.com – Expurple Jun 14, 2022 at 11:33

2 This natural-language parser for english-to-regex translation now exists: autoregex.xyz/home – AmbroseChapel Jul 15, 2022 at 21:59

7

Another problem with regex is that there are many flavours of it. .Net regex vs php regex vs other regex, all look alike but don't give the same result (sometimes no result at all).

Share Improve this answer Follow

edited Nov 5, 2008 at 15:53

answered Nov 5, 2008 at 4:00

Frederic Morin
8,913 ● 5 ● 29 ● 27

2 Which is why I tend to think of regex as "irregular expressions"... – Ilsi 50MHz Dec 7, 2010 at 1:29

7

Pyparsing (http://pyparsing.wikispaces.com/Examples) is a Python library that makes it easy to write regex-like expressions that are highly readable, like these lines that will parse "Hello, World!":

```
from pyparsing import Word, alphas
greet = Word( alphas ) + "," + Word( alphas ) +
"!"
greet.parseString("Hello, World!")
```

It looks like the library is very close to being able to match the power of regexes (see the examples page mentioned above).

Share Improve this answer Follow

answered Nov 5, 2008 at 16:23

Brandon
3,764 ● 1 ● 21 ● 26

1 Pyparsing is no longer hosted on wikispaces.com. Go to github.com/pyparsing/pyparsing – PaulMcG Aug 27, 2018 at 13:00

6

The are similar to BNF notation and the like, the basis of many a good language specification. So it makes sense that notation like this would be used in lexers to indicate classes of expected characters. The base symbols really aren't that cryptic.

Then I think there was the UNIX-do-all-you-can-in-a-single-line idea that took over. After improvements in sed and grep scripts, regexes acquired new powers but shorthand denotations for them. Larry Wall incorporated them into Perl as a general tool for parsing text. I'd guess it retained compactness for the oneliners that still were important with perl. And there were shorthand names for common character classes, and even more power was asked and given to regexes. Of course, since Perl was also a language of modules, the regex syntax also worked in blocks of statements and it leveraged syntax that was more widely known.

Their incorporation into Java is really what made people look at them, IMO. Java did nothing to accommodate them. And as a result, expressions with backslashes of moderate difficulty turned into dense jungles of backslashes. Java gave regular expressions a new market if you will, but it their most atrocious form. If you've seen those, and paid no more attention, you might think that the regex is a Thing That Should Not Be.

It's interesting that given a clear enough syntax, for Ambrose's verbose version, somebody could come up with a module for Perl that could take the verbose regex and "compile" it into a compact regex that Perl understands, using simpler regexes through `overload::constants` or possibly a `Parse::RecDescent` grammar.

Share Improve this answer Follow

answered Nov 5, 2008 at 15:49
Axeman
29.8k ● 2 ● 49 ● 103

Thanks for the rather well rounded answer. To be honest I skipped Java for many years and it interesting point that Java might have brought regex to the masses :) – Robert Gould Nov 5, 2008 at 16:08

Well, I'm not sure about that, it's just that if I've seen a moderately complex RE that takes a couple of minutes to parse, it's usually been a result of the backslash explosion necessary for Java. The other links seem plausible, if not actual. – Axeman Nov 6, 2008 at 3:55

4

It's the way it is... mostly for traditional reasons as you pointed out correctly. Now

- **Retraining and retooling**: it has gained a huge following and the roots have grown too deep for an overhaul even if someone wants to. People have learned the arcane rules and have developed their bag of tricks, communities and tools (I plug Expresso).
- **Extensive Support**: The current syntax is widely supported across platforms.. rewriting that level of support is a huge task *even if you discount the*

monumental task of writing your own engine and handling all the fringe cases

- **Reg Exp are less likely to change** Finally and most importantly RegExp cannot be equated to code w.r.t. readability. Personally I use a regexp sparingly and for quick incisions, where its advantages outweigh the benefits. (e.g. an internal tool to scrape the customer's Excel spreadsheet into an XML format devised by-for the developer cleanly.) Regexps should not need to be maintained and modified.. if it is very complex.. mask the odor with a comment (and it should only be once). If you find regexp being modified regularly (or if no one else at your workplace knows regexps), it probably was a bad choice and you should switch to regular code.

Personally I find regexps (atleast the section required for routine tasks) easy to pick up.. a day or 2. The advanced stuff is hard (second half of the MasteringRegExp book) but then you don't need it that often either.

Share Improve this answer Follow

answered Nov 5, 2008 at 4:36

Gishu
137k ● 47 ● 226 ● 311

Very good arguments! Thanks, however when thinking about it that didn't save MFC, or COBOL... But nevertheless we let regex be king... odd isn't it? – Robert Gould Nov 5, 2008 at 6:05

---

1 But you must remember the successors to those were 'easier to use' and did much 'more with less effort'.. I think it'll be

difficult to beat regexps in the latter dept even if you managed the former. – Gishu Nov 6, 2008 at 5:21

4

A regular expression (at least originally) describes a regular language. Regular languages have very nice theoretical properties in that they both can be described by and describe a deterministic finite automata. DFAs for non-trivial regular expressions are painful to code by hand.

More importantly, regular expression compilers used in the back end of perl, etc. are very good at what they do. It would be very difficult to hand tune competitively performing code.

Finally, their existence is largely a historical artifact. They've been around a long time, picked up mindshare, a lot of people know them. A tool you have and that other people can support is a lot better than a theoretical tool that doesn't yet exist.

If it is just the syntax that turns you off, maybe you should consider looking at parser combinators in Haskell. They can express a superset of the same ideas and have a more explicit syntax.

Share Improve this answer Follow

answered Nov 5, 2008 at 16:09

Edward Kmett
30k ● 7 ● 87 ● 107

But all languages began by not existing, and with the purpose of improving and filling in needs. As mentioned who imagined LINQ? – Robert Gould Nov 5, 2008 at 16:25

1 Well, facetiously, Eric Meijer. ;) LINQ is a descendant of monad comprehensions extended to handle sorting and ordering and those go back to Wadler and derive from list comprehensions, which you can chase back to SETL from the 60s. We tend to build on what came before rather than create whole cloth. – Edward Kmett Nov 5, 2008 at 18:08

3

Looking over the similar question which you mentioned and its answers, I saw a few attempts at creating "friendlier" alternative syntaxes, from both supporters and detractors of regexes as we know them today.

I found them to be uniformly less readable than equivalent regexes.

Now, granted, I am a regular user of regexes, so I'm sure my comfort with them is a significant part of this. But my primary problem with them was not unfamiliarity, but rather that they quickly grew too large to take in all at once. When your 20-character regex becomes a 10-line-by-30-column pseudo-English expression, it becomes much more difficult to see how the parts of it relate to each other.

Perhaps someone will come up with an alternative syntax to regexes which is universally more readable, even in complex cases, but I submit that such a syntax would inherently require some equivalent to subroutine calls. We don't write 200-line blocks of application code with 15 layers of nested logic because it would be a monumental task just to track the logic of it, never mind figuring out what it actually does. If we're going to explode regexes into a more English-like form, then that same problem will occur and we'll need the same tools to manage it.

Share Improve this answer Follow

answered Nov 5, 2008 at 15:57

Dave Sherohman
46.1k ● 14 ● 66 ● 103

the need for subroutines is a very good point and possible stumbling stone for a succesor. But on the other hand long regexs themselves are bad manners, maybe – Robert Gould Nov 5, 2008 at 16:22

Regexs already tend to have subroutines, but the routines and arguments are just named such that J. Random Newbie will think they look like gibberish. Initially, they reminded me of what would happen when someone dumped random data to one of the old campus printers, causing it spewed anything from a few characters to a few dozen sheets of random characters. Working in regexp is like working in some of the esoteric challenge languages; all's exceedingly terse and perfectly logical, reading them is initially beastly, but writing them gives a tremendous sense of Hack Value. – Ilsi 50MHz Dec 7, 2010 at 1:48

▲
3

From the perl module Regexp::English:

> Regexp::English provides an alternate regular expression syntax, one that is slightly more verbose than the standard mechanisms. In addition, it adds a few convenient features, like incremental expression building and bound captures.

```
use Regexp::English;

my $re = Regexp::English
        -> start_of_line
        -> literal('Flippers')
        -> literal(':')
        -> optional
                -> whitespace_char
        -> end
        -> remember
                -> multiple
                        -> digit;

while (<INPUT>) {
        if (my $match = $re->match($_)) {
                print "$match\n";
        }
}
```

Share Improve this answer Follow

answered May 31, 2012 at 4:14

Matthew Lock
13.5k ● 14 ● 96 ● 132

2

Like Adam said, is there anything better? I shudder to think of trying to do a bunch of strcmp operations instead of a good regex. Like any expressive language it is possible to abuse the regex and make very unreadable constructs, but often even a seemingly unreadable regex makes much more sense than the equivalent procedural code to achieve the same thing.

Without the regex, you'd have to write your own routine to parse, backtrack, compare, store indexes, make substitutions, and manage all the results. Regex gives you all that in a very expressive form. I don't like to repeat code, but if I had to code my own parser every time I needed to look for a moderately complex pattern... I don't know what I'd do.

Yes there are different flavors of regex, but most the most popular ones are very similar, and whatever language you are using them in usually has documentation to help you read it.

Share Improve this answer Follow

answered Nov 5, 2008 at 4:15

DGM
27k ● 7 ● 64 ● 82

0

I bet all participants in this discussion will agree that for certain **small** regex codes one would have to write a long paragraph in English describing what it does. Any kind of language that may be used to do what even the simplest regexp describes will probably do it with N lines of code, where N (probably) grows exponentially in comparison to the length of the regexp itself.

answered Nov 29, 2011 at 15:23

Share Improve this answer

Follow

DejanLekic

19.8k ● 4 ● 51 ● 81