EDiNA Labs

(De)?mystifying Regular Expressions

Rejected titles:

~~Dem(ystify|onstrat|onis)ing Regular Expressions~~
~~How not to be scared of Regular Expressions~~
~~How not to be scared of building Regular Expressions~~
~~How to build scary Regular Expressions~~
~~How to be scared by your own Regular Expressions~~
~~Engaging with change using Regular Expressions~~

by Ben(edict)? Soares

Wednesday, 13 July, 2016

EDiNA Labs

regex, huh, what is it good for?

Searching

Filtering

Replacing

● **Capturing**

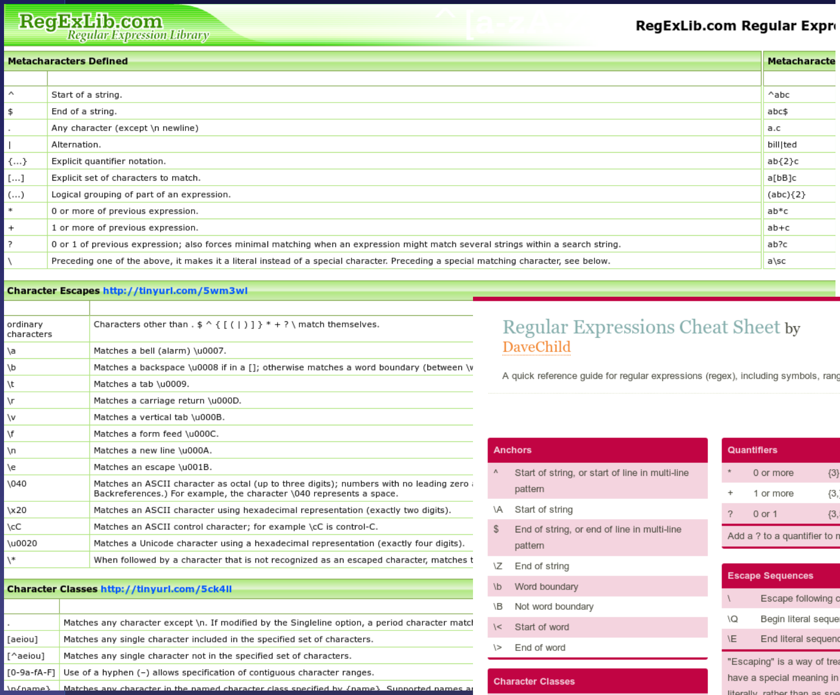
Replacing

EDINA Labs

(crib|cheat) sheets

There are many:

http://lmgtfy.com/?q=regex+crib+sheet



RegExLib.com Regular Expression Library

RegExLib.com Regular Expr

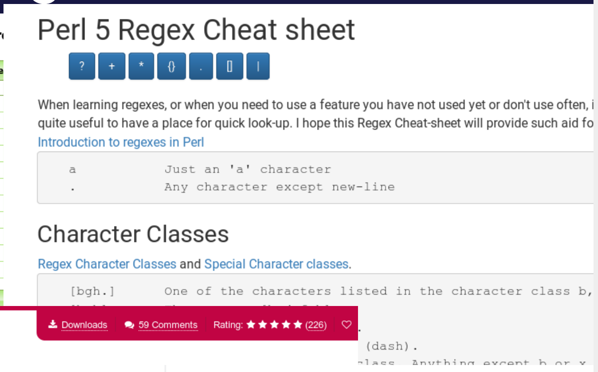
Metacharacters Defined		Metacharacte
^	Start of a string.	^abc
$	End of a string.	abc$
.	Any character (except \n newline)	a.c
|	Alternation.	bill|ted
{...}	Explicit quantifier notation.	ab{2}c
[...]	Explicit set of characters to match.	a[bB]c
(...)	Logical grouping of part of an expression.	(abc){2}
*	0 or more of previous expression.	ab*c
+	1 or more of previous expression.	ab+c
?	0 or 1 of previous expression; also forces minimal matching when an expression might match several strings within a search string.	ab?c
\	Preceding one of the above, it makes it a literal instead of a special character. Preceding a special matching character, see below.	a\sc

Character Escapes http://tinyurl.com/5wm3wl

ordinary characters	Characters other than . $ ^ { [(|)] } * + ? \ match themselves.
\a	Matches a bell (alarm) \u0007.
\b	Matches a backspace \u0008 if in a []; otherwise matches a word boundary (between \w
\t	Matches a tab \u0009.
\r	Matches a carriage return \u000D.
\v	Matches a vertical tab \u000B.
\f	Matches a form feed \u000C.
\n	Matches a new line \u000A.
\e	Matches an escape \u001B.
\040	Matches an ASCII character as octal (up to three digits); numbers with no leading zero a Backreferences.) For example, the character \040 represents a space.
\x20	Matches an ASCII character using hexadecimal representation (exactly two digits).
\cC	Matches an ASCII control character; for example \cC is control-C.
\u0020	Matches a Unicode character using a hexadecimal representation (exactly four digits).
*	When followed by a character that is not recognized as an escaped character, matches t

Character Classes http://tinyurl.com/5ck4ll

.	Matches any character except \n. If modified by the Singleline option, a period character matcl
[aeiou]	Matches any single character included in the specified set of characters.
[^aeiou]	Matches any single character not in the specified set of characters.
[0-9a-fA-F]	Use of a hyphen (–) allows specification of contiguous character ranges.
\p{name}	Matches any character in the named character class specified by {name}. Supported names ar



Perl 5 Regex Cheat sheet

? + * {} . [] |

When learning regexes, or when you need to use a feature you have not used yet or don't use often, i quite useful to have a place for quick look-up. I hope this Regex Cheat-sheet will provide such aid fo

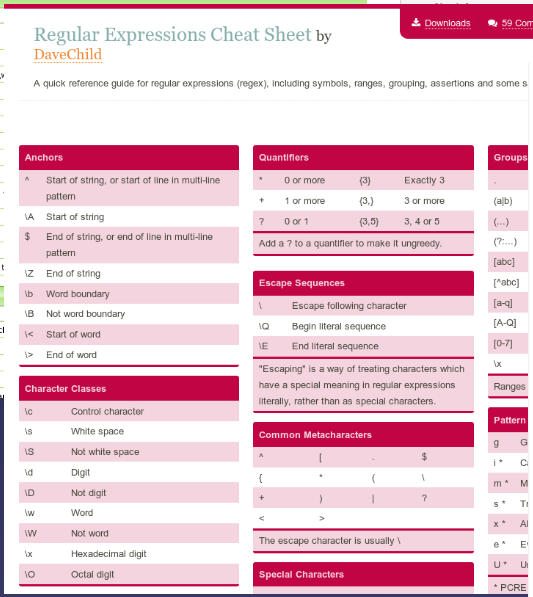
Introduction to regexes in Perl

```
    a           Just an 'a' character
    .           Any character except new-line
```

Character Classes

Regex Character Classes and Special Character classes.

```
    [bgh.]      One of the characters listed in the character class b,
                                                             (dash).
                                                  class. Anything except b or x
```



Downloads 59 Comments Rating: ★★★★★ (226)

Regular Expressions Cheat Sheet by DaveChild

A quick reference guide for regular expressions (regex), including symbols, ranges, grouping, assertions and some s

Anchors	
^	Start of string, or start of line in multi-line pattern
\A	Start of string
$	End of string, or end of line in multi-line pattern
\Z	End of string
\b	Word boundary
\B	Not word boundary
\<	Start of word
\>	End of word

Character Classes	
\c	Control character
\s	White space
\S	Not white space
\d	Digit
\D	Not digit
\w	Word
\W	Not word
\x	Hexadecimal digit
\O	Octal digit

Quantifiers			
*	0 or more	{3}	Exactly 3
+	1 or more	{3,}	3 or more
?	0 or 1	{3,5}	3, 4 or 5

Add a ? to a quantifier to make it ungreedy.

Escape Sequences	
\	Escape following character
\Q	Begin literal sequence
\E	End literal sequence

"Escaping" is a way of treating characters which have a special meaning in regular expressions literally, rather than as special characters.

Common Metacharacters			
^	[	.	$
{	*	(	\
+	)	|	?
<	>		

The escape character is usually \

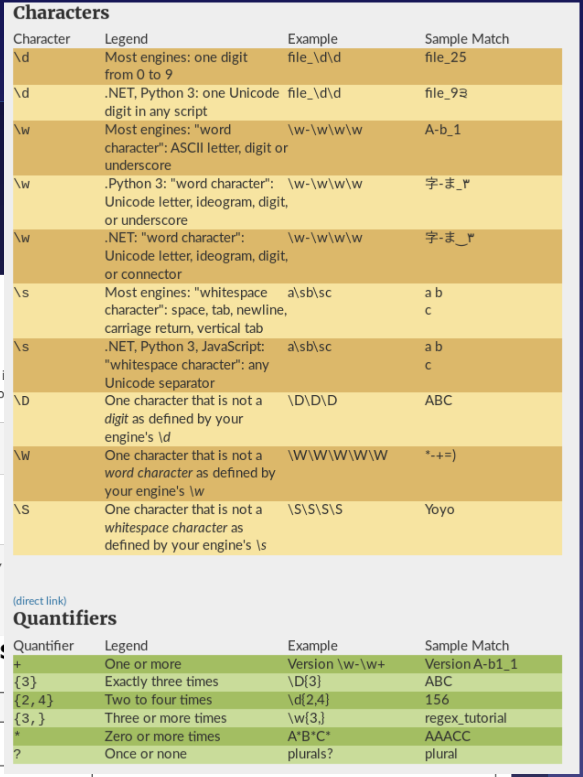
Special Characters

Groups
.
(a|b)
(...)
(?:...)
[abc]
[^abc]
[a-q]
[A-Q]
[0-7]
\x

Ranges

Pattern	
g	G
i *	C
m *	M
s *	Tr
x *	Al
e *	E
U *	U

* PCRE



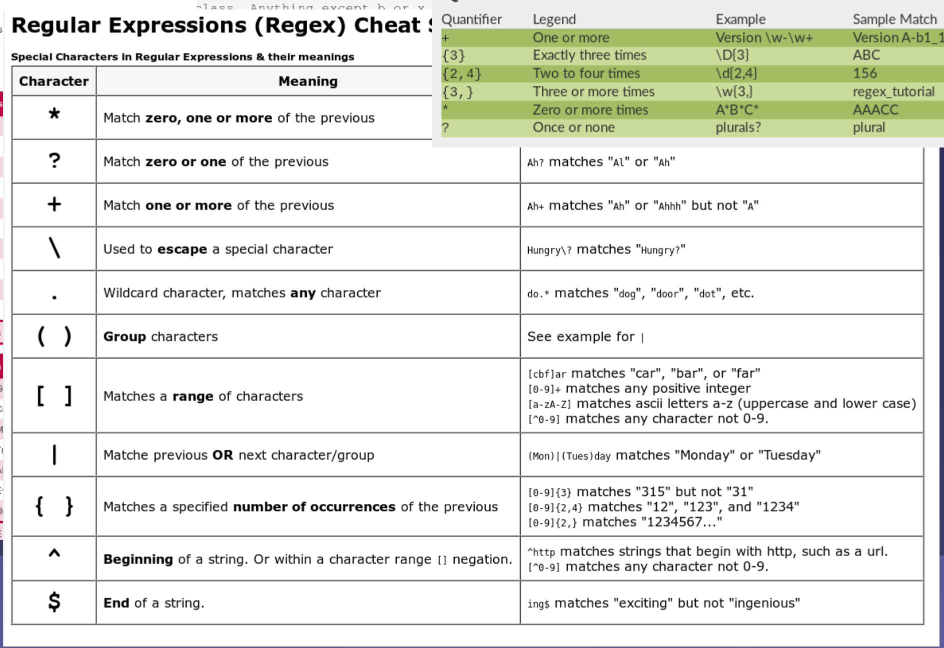
Characters

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_9३
\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*	AACC
?	Once or none	plurals?	plural



Regular Expressions (Regex) Cheat

Special Characters in Regular Expressions & their meanings

Character	Meaning	
*	Match **zero, one or more** of the previous	
?	Match **zero or one** of the previous	Ah? matches "Al" or "Ah"
+	Match **one or more** of the previous	Ah+ matches "Ah" or "Ahhh" but not "A"
\	Used to **escape** a special character	Hungry\? matches "Hungry?"
.	Wildcard character, matches **any** character	do.* matches "dog", "door", "dot", etc.
( )	**Group** characters	See example for |
[ ]	Matches a **range** of characters	[cbf]ar matches "car", "bar", or "far" [0-9]+ matches any positive integer [a-zA-Z] matches ascii letters a-z (uppercase and lower case) [^0-9] matches any character not 0-9.
|	Matche previous **OR** next character/group	(Mon)|(Tues)day matches "Monday" or "Tuesday"
{ }	Matches a specified **number of occurrences** of the previous	[0-9]{3} matches "315" but not "31" [0-9]{2,4} matches "12", "123", and "1234" [0-9]{2,} matches "1234567..."
^	**Beginning** of a string. Or within a character range [] negation.	^http matches strings that begin with http, such as a url. [^0-9] matches any character not 0-9.
$	**End** of a string.	ing$ matches "exciting" but not "ingenious"

This talk isn't one

Wednesday, 13 July, 2016

except for this basic and slightly fictitious one

.	any **character**
[a-mz]	collection of characters (case sensitive, unicode)
[^a-mz]	a negated collection of characters (i.e. *not* those characters)
\w	any letter or underscore
\d	any numeric digit (same as [0-9])
\n	newline
(match)	capture block
n?	zero or one of *n*
*n**	zero or more of *n* *(greedy)*
n+	one or more of *n* *(greedy)*
*n**?	non-greedy match
\r	Carriage return (oldline, ha ha ha)
\r\n	You're in the wrong operating system
[\r\n]+	You're not too bothered
\[	an actual (escaped) '['
\b	a word boundary (not a character!)
\🦁	an escaped lion
\🦁+	one or more greedy escaped lions

EDINA Labs

translations

English	Perl	Python	Java
map	hash	dictionary	HashMap
command line script	one-liner	how do you indent it?	no meaningful translation
recent version	last 9 years	version 2.7 but not 3 or above	5 years but IANAE

EDiNA Labs

creating regular expressions

Build it up, build it across, join it up

Pre-compile? (efficiency for e.g. sorting comparisons)

Test: http://pythex.org/

Visualise: http://debuggex.com/

EDINA Labs

usage

Perl:

$string =~ m/*my_regex*/;

$string =~ s/*my_search*/*my_replace*/;

Python:

import re

match = re.search(‘*my_regex*’, string)

newsting = re.sub(‘*my_search*’, ‘*my_replace*’, string)

EDiNA Labs

capturing

Perl:

```
my $string = ‘abcdefghijklmnop’;
$string =~ m/abc(...)ghi/;
my $letters = $1;

$string =~ m/abc(?<letters>...)ghi/;
my $letters = $+{letters};

$string =~ s/^(?<surname>([A-Z][A-Za-z]*[ -]?){1,2}),\s*(?
<initial>[A-Z])(\.|[a-z]+).*/$+{initial}. $+{surname}/;
```

EDiNA Labs

capturing

Python:

```
string = ‘abcdefghijklmnop’
match = re.search(‘abc(...)ghi’, string)
letters = match.group(1)

match = re.search(‘abc(?P<letters>...)ghi’, string)
letters = match.group(‘letters’)

newstring = re.sub(‘^(?P<surname>([A-Z][A-Za-z]*[ -]?){1,2}),\s*(?
P<initial>[A-Z])(\.|[a-z]+).*', '\g<initial>. \g<surname>', string)
```

EDiNA Labs

example: parsing a cubic polynomial

school project: input a cubic in *x* in the form

e.g. 4.3 x^3 + x^2 + 5.6 x + 0.1

(then use Cardano's formula to factorise)

```
cubic = re.sub("\s+", "", cubic)
regex = "(?P<a>-?\d*(\.\d+)?)x\^3(?P<b>(\+|-)\d*(\.\d+)?)x\^2
(?P<c>(\+|-)\d*(\.\d+)?)x(?P<d>(\+|-)\d*(\.\d+)?)"
m = re.match(regex, cubic)
a = int(m.group("a"))
b = int(m.group("b"))
c = int(m.group("c"))
d = int(m.group("d"))
```

EDiNA Labs

examples

Apache logs

XML

volunteer from the audience

EDiNA
Labs

prime number detector

^1?$|^(11+?)\1+$

e.g.

```
perl -e 'print("$ARGV[0] is ",(1 x $ARGV[0]) =~
m/^1?$|^(11+?)\1+$/
?"not ":"","prime\n");' 29
29 is prime
```