Regular Expressions for Triage

A PRIMER ON REGEX

FOR INCIDENT RESPONDERS

Class Objectives

By the end of today's class, you will be able to:

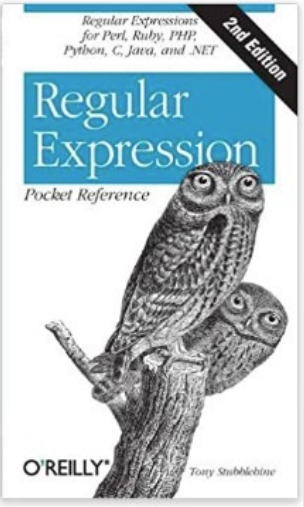
- Parse, Interpret, and analyze data with fundamental Regex syntax
- Leverage the power of Regular Expressions (Regex) to navigate logs using the grep command
- Conduct field extraction and transformation of log files within Splunk Enterprise using Regex
- Identify references that will aid you in exploring additional Regular Expression Engines

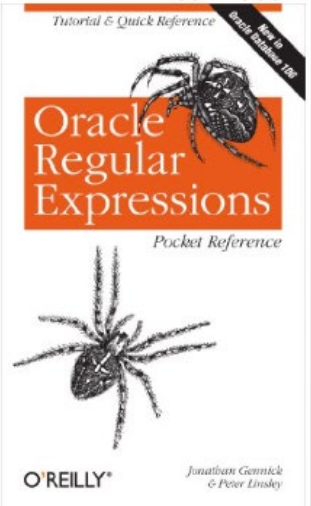
Agenda

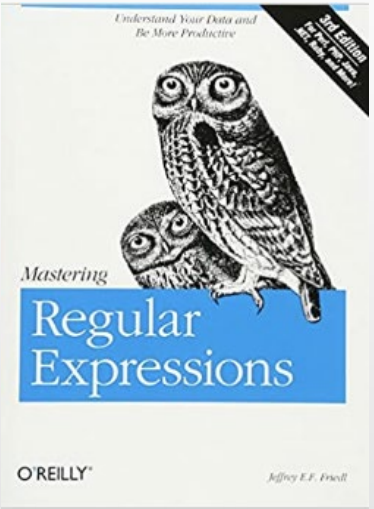
Regular Expressions Background

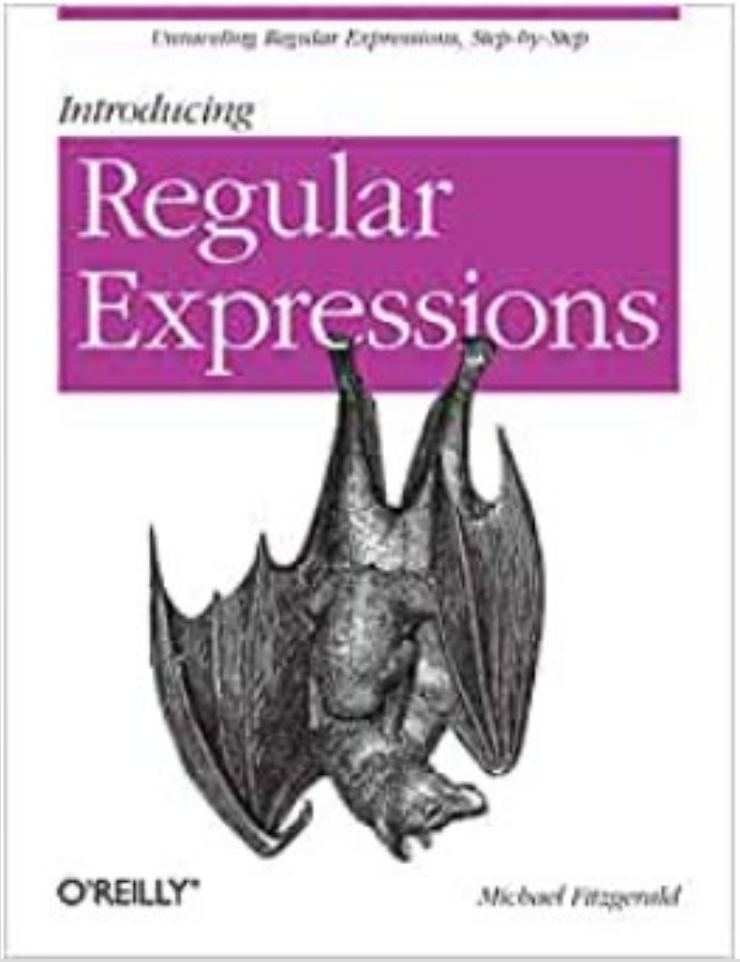
Regular Expressions Fundamentals

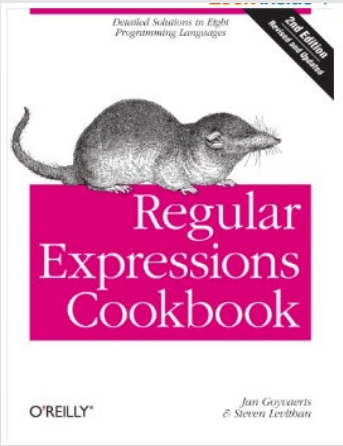
Develop Better Regular Expressions for Triage in Incident Response


Regular Expressions for Perl, Ruby, PHP, Python, C, Java, and .NET
2nd Edition
Regular Expression
Pocket Reference
O'REILLY
Tony Stubblebine

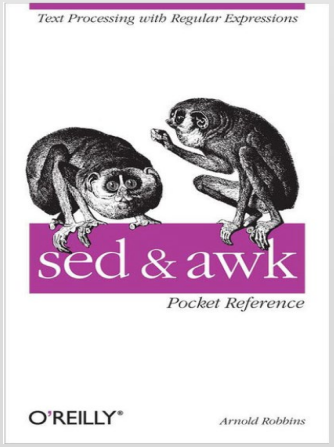

Tutorial & Quick Reference
Oracle Regular Expressions
Pocket Reference
O'REILLY
Jonathan Gennick & Peter Linsley


Understand Your Data and Be More Productive
3rd Edition
Mastering
Regular Expressions
O'REILLY
Jeffrey E.F. Friedl


Introducing
Regular Expressions
O'REILLY
Michael Fitzgerald

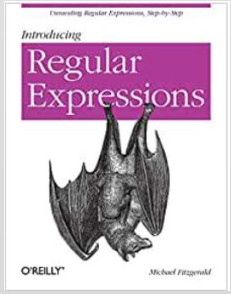

Detailed Solutions in Eight Programming Languages
2nd Edition
Regular Expressions Cookbook
O'REILLY
Jan Goyvaerts & Steven Levithan

O'REILLY Short Cuts
Automating InDesign with Regular Expressions
Contents


Text Processing with Regular Expressions
sed & awk
Pocket Reference
O'REILLY
Arnold Robbins

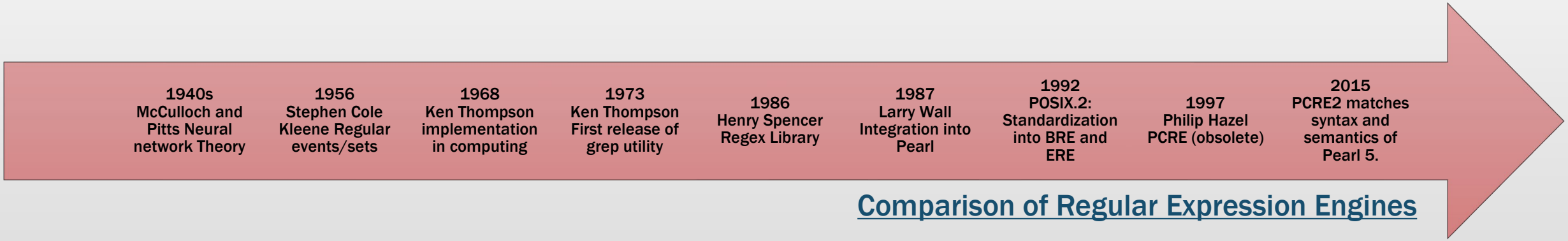
What are Regular Expressions?

- Regular expressions help [responders] find **patterns** in text strings. More precisely, regular expressions are specially encoded text strings that match **patterns** in sets of strings.
  - Think of regular expressions as a coding language, in which each character is a sub-routine.



Fitzgerald, M. (2012). *Introducing regular expressions*. O'Reilly.

Brief History of Regular Expressions



Comparison of Regular Expression Engines

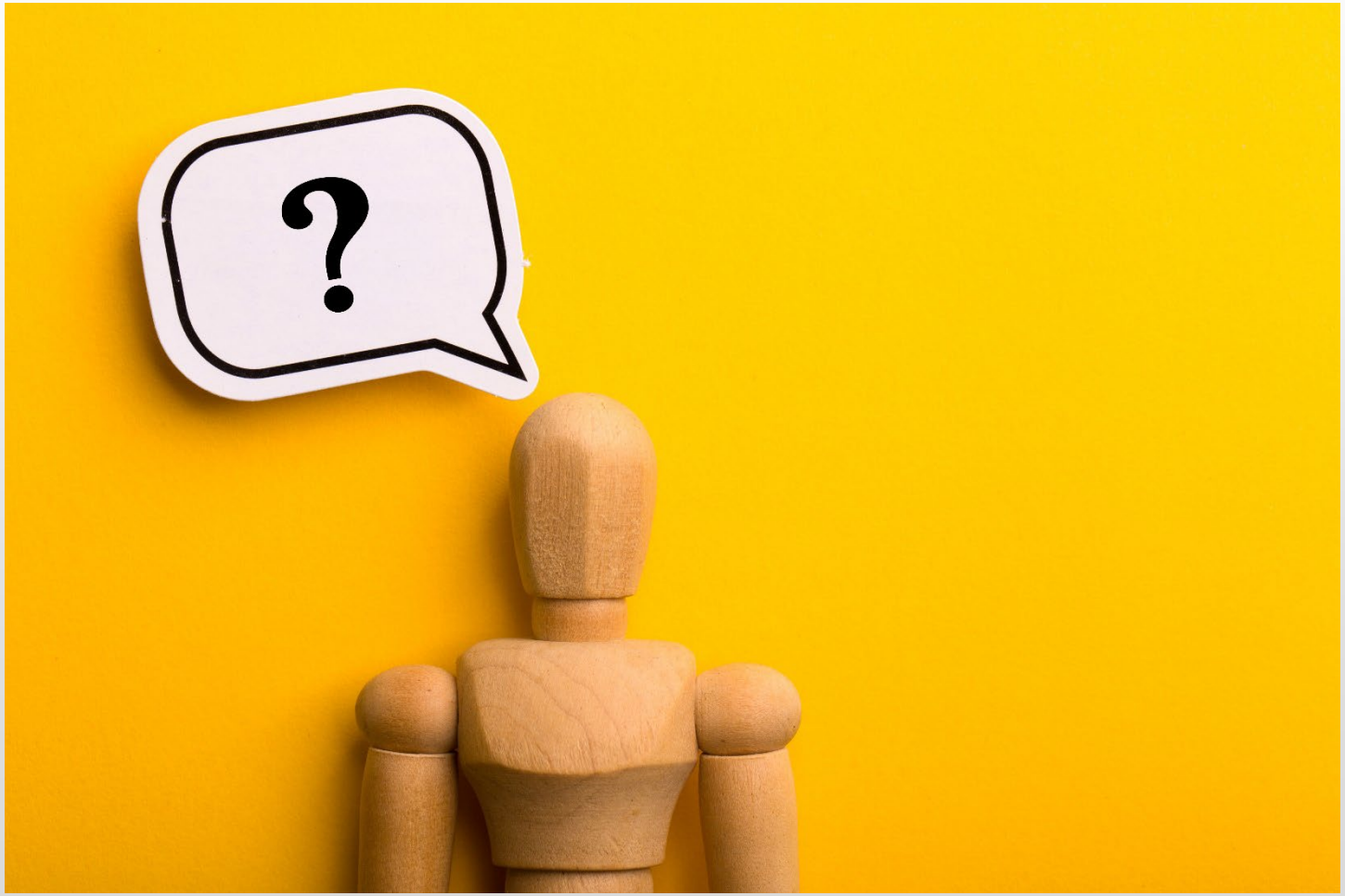
https://github.com/PhilipHazel
https://github.com/PCRE2Project/pcre2

Use Cases for Regular Expressions in Triage

- URLS
- Date/Time
- IP Addresses
- Email Addresses
- Social Security Numbers
- Credit Card Numbers

Group Activity
Finding Credit Card Numbers With Regex

QUESTIONS?



Part 1
Regex Fundamentals

Character Classes

[ ]

Positive	Negative
▪[abcdef]	▪[^abcdef]
▪[A-Za-z0-9]	▪[^A-Za-z0-9]
▪[a-f_%0-9]	▪[^a-f_%0-9]

Matching Characters

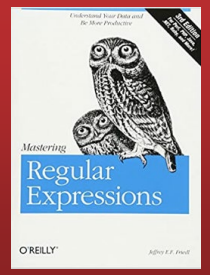
Using ordinary characters to match just and only a literal string is not very efficient and normally there are more appropriate functions available to do just that.

Ordinary Characters

- The letter A
- The number 0
- The string ab
- The letter a|b

Matching Characters

F., F. J. E. (2006). *Mastering regular expressions*. O'Reilly.



Special Characters

- [A-Za-z_-]
- [- -/A-Z]
- [A-Z+- -]
- [A-Za-z^]
- [[] [A-Z\[] [A-Z\]]

Be wary of order you supply the characters to the character range (ASCII | Unicode) e.g., [A-z]

Hex	Dec	Char		Hex	Dec	Char	Hex	Dec	Char	Hex	Dec	Char
0x00	0	NULL	null	0x20	32	Space	0x40	64	@	0x60	96	`
0x01	1	SOH	Start of heading	0x21	33	!	0x41	65	A	0x61	97	a
0x02	2	STX	Start of text	0x22	34	"	0x42	66	B	0x62	98	b
0x03	3	ETX	End of text	0x23	35	#	0x43	67	C	0x63	99	c
0x04	4	EOT	End of transmission	0x24	36	$	0x44	68	D	0x64	100	d
0x05	5	ENQ	Enquiry	0x25	37	%	0x45	69	E	0x65	101	e
0x06	6	ACK	Acknowledge	0x26	38	&	0x46	70	F	0x66	102	f
0x07	7	BELL	Bell	0x27	39	'	0x47	71	G	0x67	103	g
0x08	8	BS	Backspace	0x28	40	(	0x48	72	H	0x68	104	h
0x09	9	TAB	Horizontal tab	0x29	41	)	0x49	73	I	0x69	105	i
0x0A	10	LF	New line	0x2A	42	*	0x4A	74	J	0x6A	106	j
0x0B	11	VT	Vertical tab	0x2B	43	+	0x4B	75	K	0x6B	107	k
0x0C	12	FF	Form Feed	0x2C	44	,	0x4C	76	L	0x6C	108	l
0x0D	13	CR	Carriage return	0x2D	45	-	0x4D	77	M	0x6D	109	m
0x0E	14	SO	Shift out	0x2E	46	.	0x4E	78	N	0x6E	110	n
0x0F	15	SI	Shift in	0x2F	47	/	0x4F	79	O	0x6F	111	o
0x10	16	DLE	Data link escape	0x30	48	0	0x50	80	P	0x70	112	p
0x11	17	DC1	Device control 1	0x31	49	1	0x51	81	Q	0x71	113	q
0x12	18	DC2	Device control 2	0x32	50	2	0x52	82	R	0x72	114	r
0x13	19	DC3	Device control 3	0x33	51	3	0x53	83	S	0x73	115	s
0x14	20	DC4	Decive control 4	0x34	52	4	0x54	84	T	0x74	116	t
0x15	21	NAK	Negative ack	0x35	53	5	0x55	85	U	0x75	117	u
0x16	22	SYN	Synchronous idle	0x36	54	6	0x56	86	V	0x76	118	v
0x17	23	ETB	End transmission block	0x37	55	7	0x57	87	W	0x77	119	w
0x18	24	CAN	Cancel	0x38	56	8	0x58	88	X	0x78	120	x
0x19	25	EM	End of medium	0x39	57	9	0x59	89	Y	0x79	121	y
0x1A	26	SUB	Substitute	0x3A	58	:	0x5A	90	Z	0x7A	122	z
0x1B	27	FSC	Escape	0x3B	59	;	0x5B	91	[	0x7B	123	{
0x1C	28	FS	File separator	0x3C	60	<	0x5C	92	\	0x7C	124	|
0x1D	29	GS	Group separator	0x3D	61	=	0x5D	93	]	0x7D	125	}
0x1E	30	RS	Record separator	0x3E	62	>	0x5E	94	^	0x7E	126	~
0x1F	31	US	Unit separator	0x3F	63	?	0x5F	95	_	0x7F	127	DEL

Meta Characters

Escaping Meta Characters

- Remove special meaning from meta-characters
- Give special meaning to ordinary characters

Meta Characters

Escaping Meta Characters

Special Characters

[ ] ( ) | . ? * + { } ^ $ \/

Literal Characters

\[\] \(\) \| \. \? * \+ \{ \} \^ \$ \\ \/

Regular Expression Basics

.	Any character except newline
a	The character a
ab	The string ab
a|b	a or b
a*	0 or more a's
\	Escapes a special character

Regular Expression Quantifiers

*	0 or more
+	1 or more
?	0 or 1
{2}	Exactly 2
{2, 5}	Between 2 and 5
{2,}	2 or more

Default is greedy. Append ? for reluctant.

Regular Expression Groups

(...)	Capturing group
(?P<Y>...)	Capturing group named Y
(?:...)	Non-capturing group
(?>...)	Atomic group
(?|...)	Duplicate group numbers
\Y	Match the Y'th captured group
(?P=Y)	Match the named group Y
(?R)	Recurse into entire pattern
(?Y)	Recurse into numbered group Y
(?&Y)	Recurse into named group Y
\g{Y}	Match the named or numbered group Y
\g<Y>	Recurse into named or numbered group Y
(?#...)	Comment

Regular Expression Character Classes

[ab-d]	One character of: a, b, c, d
[^ab-d]	One character except: a, b, c, d
[\b]	Backspace character
\d	One digit
\D	One non-digit
\s	One whitespace
\S	One non-whitespace
\w	One word character
\W	One non-word character

Regular Expression Assertions

^	Start of string
\A	Start of string, ignores m flag
$	End of string
\Z	End of string, ignores m flag
\b	Word boundary
\B	Non-word boundary
\G	Start of match
(?=...)	Positive lookahead
(?!...)	Negative lookahead
(?<=...)	Positive lookbehind
(?<!...)	Negative lookbehind
(?()|)	Conditional

Regular Expression Escapes

\Q..\E	Remove special meaning

Regular Expression Flags

i	Ignore case
m	^ and $ match start and end of line
s	. matches newline as well
x	Allow spaces and comments
J	Duplicate group names allowed
U	Ungreedy quantifiers
(?iLmsux)	Set flags within regex

Regular Expression Special Characters

\n	Newline
\r	Carriage return
\t	Tab
\0	Null character
\YYY	Octal character YYY
\xYY	Hexadecimal character YY
\x{YY}	Hexadecimeal character YY
\cY	Control character Y

Regular Expression Posix Classes

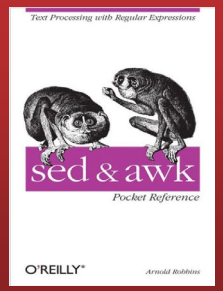
[:alnum:]	Letters and digits
[:alpha:]	Letters
[:ascii:]	Ascii codes 0 - 127
[:blank:]	Space or tab only
[:cntrl:]	Control characters
[:digit:]	Decimal digits
[:graph:]	Visible characters, except space
[:lower:]	Lowercase letters
[:print:]	Visible characters
[:punct:]	Visible punctuation characters
[:space:]	Whitespace
[:upper:]	Uppercase letters
[:word:]	Word characters
[:xdigit:]	Hexadecimal digits

PCRE Regex Cheat Sheet

debuggex.com/cheatsheet/regex/pcre

Meta Characters

Dougherty, D., & Robbins, A. (1997). *Sed & Awk*. O'Reilly & Associates.



Wildcard (Dot)

Will match any character except a new line \n

A dot has no superpowers from inside a class e.g., [.] <- That is simply a period ☺

A dot will have additional non-matches depending on the regular expression engine you are using (e.g., awk and sed) Consult your regular expression engine to ensure proper functionality.

Meta Characters

Anchors and Boundaries

- Zero-width
  - Cannot be Quantified
  - Cannot be captured

^ Start of a string

$ End of a string

Be aware that $ might match \n at the end of the string. This could have security implications, like if you inject a matched email address into an email header.

\z can be used instead of $ to ensure there is no \n at the end of the string but is not available in all regex engines.

Meta Characters

Repetitive Quantifiers

?

Match zero or one times

*

Match zero or more times
(Unlimited)

+

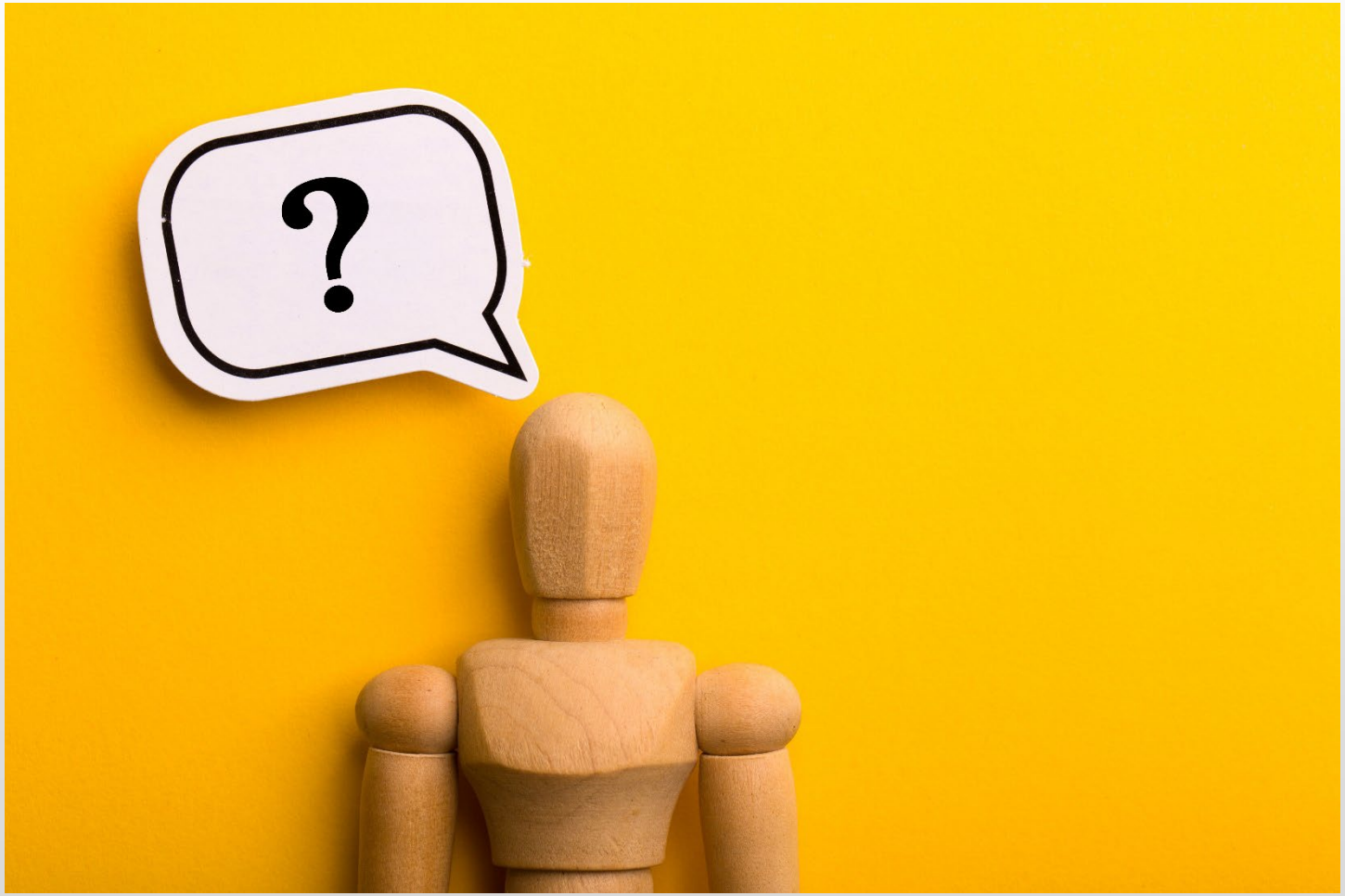
Match one or more times
(Unlimited)

⚠ Not to be confused with the *

Quantifiers and Greediness

Instructor Demonstration
Analysis of Log Files with grep

QUESTIONS?



Part 2
Better Regex for Triage

Develop Better Regex for Triage

\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\

RFC1918 Ip Address Validation

```
/\b(?:(?:2(?:[0-4][0-9]|5[0-5])|[0-1]?[0-9]?[0-9])\.){3}(?:(?:2([0-4][0-9]|5[0-5])|[0-1]?[0-9]?[0-9]))\b/ig
```

(?<email>.+@.+\.w+)\s

RFC2822 Email Validation

```
/[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*@(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?/g
```

We're Going the Distance. We're Going for Speed

- We are performing Triage we do not need to be Thorough
- We can allow false positives
- We do not want to allow false negatives

Method for Developing Better Regex for Triage

Identify Anchors

(http|ftp|https)://www.google.com/pictures/cat_pictures
jack.turner@turnerhomestead.com

Focus on the Negative

- (http|ftp|https):\/\/(?<domain>[^\s\n]*)[^\/]

It is often faster to be Exclusive rather than Inclusive when testing regular expressions

Method for Developing Better Regex for Triage

Inclusive

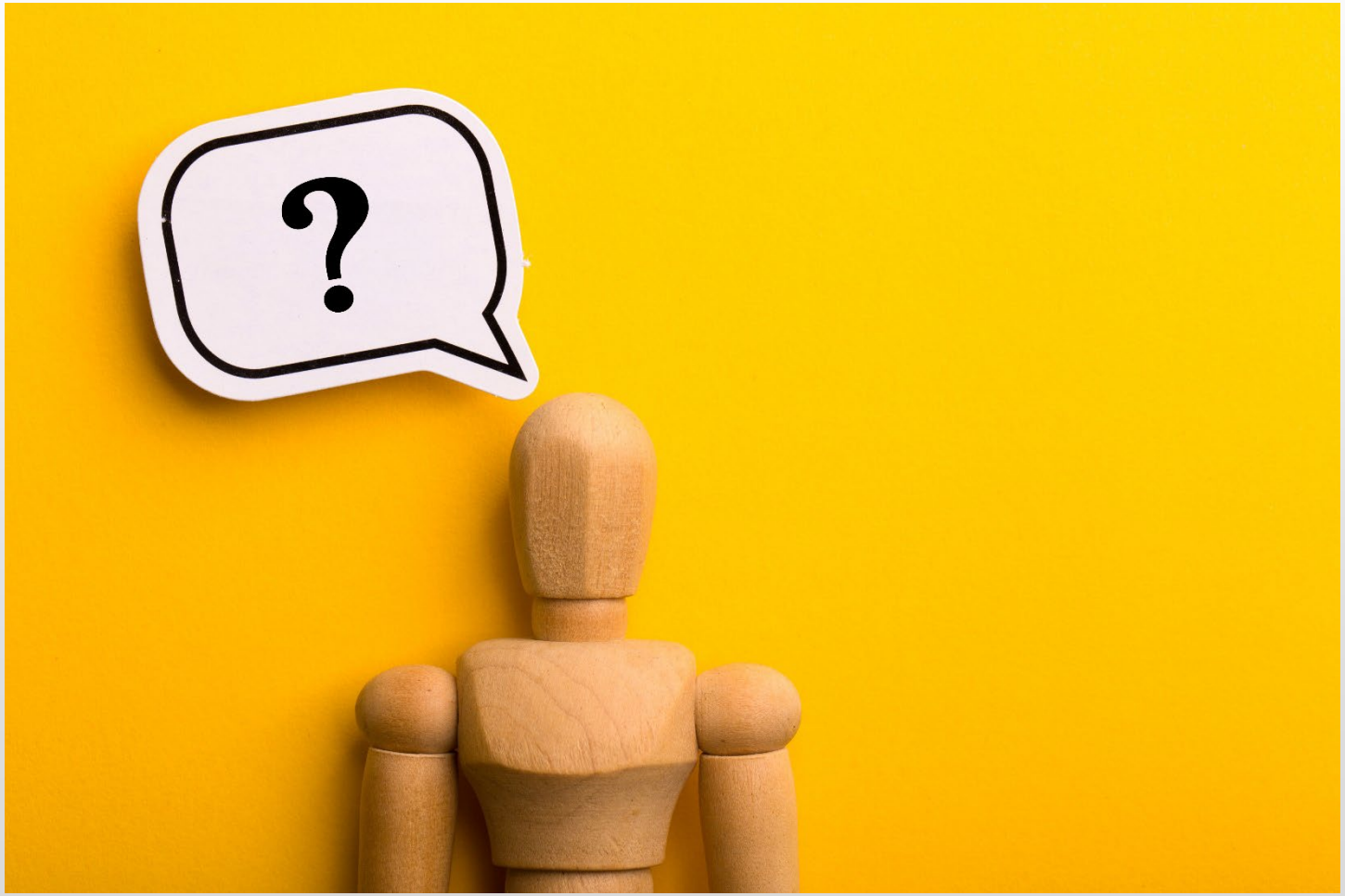
([a-zA-Z0-9_.-]+)@[a-zA-Z0-9_.-]+)\.([a-zA-Z{2,5}])

Exclusive

- [^\s]+@[^\s]+\.[^\s]+

Backtracking

When developing a pattern, consider how backtracking might affect the performance of the regular expression engine, particularly if your regular expression is designed to process unconstrained input or unstructured data.

Extracting Data with Splunk and Regex

QUESTIONS?