Regular Expression Wizardry

Josiah Mory *@kickinbahk*

Let's set the stage...

/\A[\w+\-.]+@[a-z\d\-]+(\.[a-z\d\-]+)*\.[a-z]+\z/i

What is a Regular Expression?

Regular Expression

=

a pattern describing a specific string of text

/regex-pattern-here/

Regular Expression "Engine"

=

a piece of software which processes regular expressions and tries to match the pattern to the given string

The syntax and behavior of a particular engine

=

regular expression flavor

3 Things You Do with Regular Expressions

Search

a string to see if it matches your pattern

Extract

a string (or part of a string) that matches your pattern

Replace

a string by replacing parts that match with other text

“Find-and-Replace on Steroids”
- Dan Nguyen

Literal Characters

Most Characters:

- a - z
- A - Z
- 0 - 9

Special Characters

12 special or (meta) characters

\, ^, $, ., |, ?, *, +, (,), [, [,

* If you want to use any of these characters as a literal in a regex, you need to escape them with a backslash

DC Picture: AFP/GETTY

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

The escape character is usually \

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.

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

Groups and Ranges	
.	Any character except new line (\n)
(a|b)	a or b
(...)	Group
(?:...)	Passive (non-capturing) group
[abc]	Range (a or b or c)
[^abc]	Not (a or b or c)
[a-q]	Lower case letter from a to q
[A-Q]	Upper case letter from A to Q
[0-7]	Digit from 0 to 7
\x	Group/subpattern number "x"

Ranges are inclusive.

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

String Replacement	
$n	nth non-passive group
$2	"xyz" in /^(abc(xyz))$/
$1	"xyz" in /^(?:abc)(xyz)$/
$`	Before matched string
$'	After matched string
$+	Last matched string
$&	Entire matched string

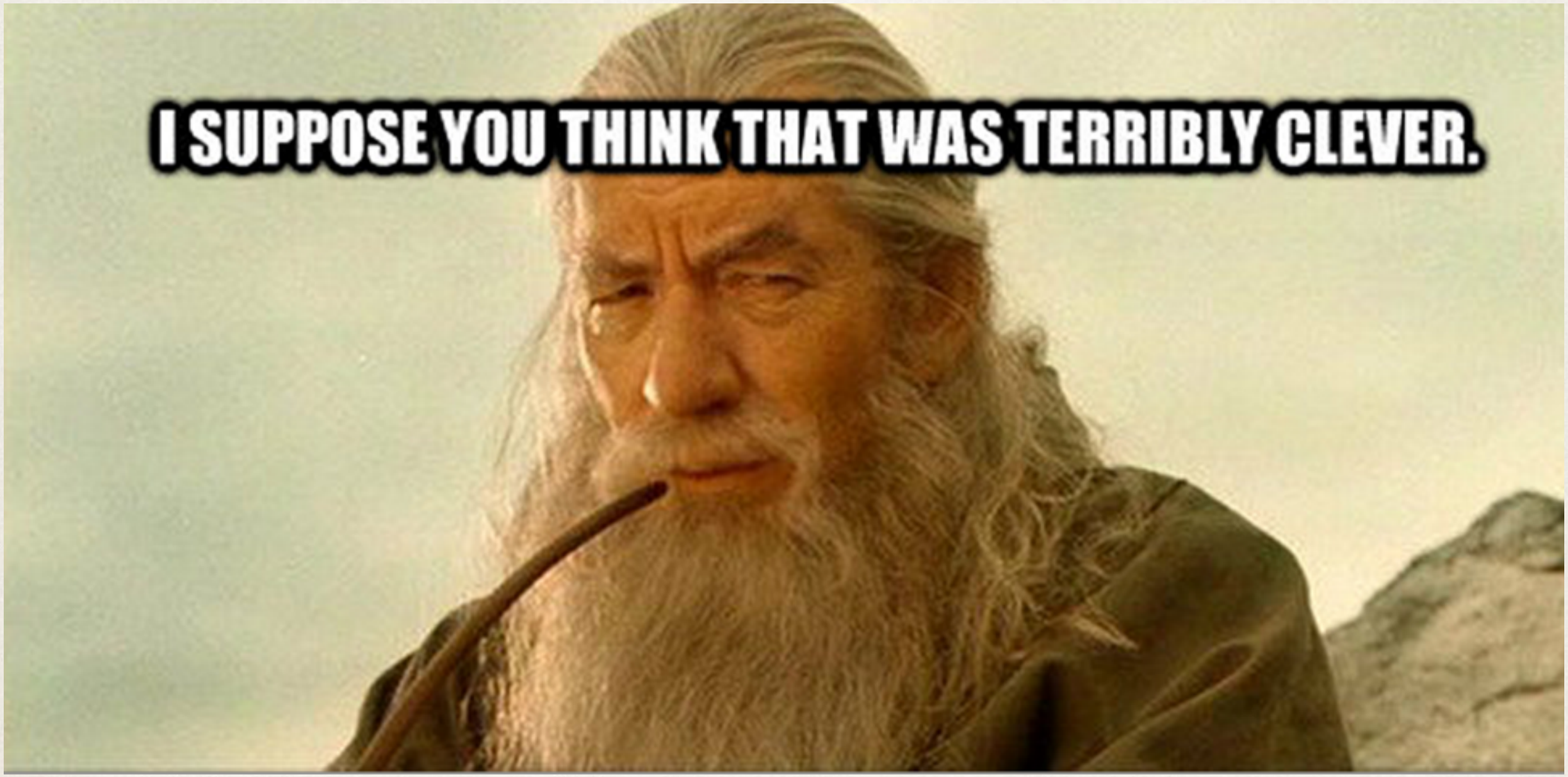
Some regex implementations use \ instead of $.

```
gandalf_quote1 = "You shall not pass! -Gandalf"

the_grey = “ The Grey”

console.log(gandalf_quote + the_grey);
```

~> You shall not pass! -Gandalf The Grey

gandalf_quote2 = "Yes, yes my dear sir and I do know your name Mr. Bilbo Baggins. And you do know my name, though you don't remember that I belong to it. I am Gandalf, and Gandalf, means me."

gandalf_quote2 = "Yes, yes my dear sir and I do know your name Mr. Bilbo Baggins. And you do know my name, though you don't remember that I belong to it. I am Gandalf ” + the_grey + “ and Gandalf ” + the_grey + “ means me."


I SUPPOSE YOU THINK THAT WAS TERRIBLY CLEVER.

Lookaheads

- Negative - ?!
- Positive - ?=

Capture Groups

() allow us to group a RegEx together

Groups are numbered 1-99

You can call a Group using $ and the number

E.g. $1

To not capture a group in parens

?:

```
/.*\.com&&|\/(?:(?:groups\/[^\/]+\/videos\/)|
(?:ondemand|channels)(?:(?:\/less\/)|
(?:user[0-9]+\/review\/)?([0-9]+).*|(?:\/\w*
\/))|(?:video\/))?([0-9]+).*$/
```

Javascript The Good Parts (Chap 7) - Douglas Crockford

RegEx Pal - RegEx Tester and Editor for Javascript

http://regexpal.com/

Eloquent Javascript (Chap 9) - Marijn Haverbeke

http://eloquentjavascript.net/09_regexp.html

MDN (Mozilla Developer Network) on Regular Expressions

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions

Josiah's Github

https://github.com/Regular_Expressions_SoCalCodeCamp_JS