Regular Expressions

If you have a problem that needs a regular expression, now you have two problems…

What?

- A micro language used to perform pattern matching on strings/text
- Developed in the 1950s
- Baked into Unix, now a popular feature of many programming languages

Where?

- POSIX as defined by the UNIX standard – original (grep's default engine)
- Perl-based (Perl, PHP)
- Others (mostly compatible with the Perl implementation)

Contact me: @EricPickupYP on Twitter

Where?

Language	Official website	Software license
.NET	MSDN	Proprietary
C++11 (C++)	C++ standards website	***?***
D	D	Boost Software License
Go	Golang.org	BSD-style
Haskell	Haskell.org	BSD3
Java	Java	GNU General Public License
JavaScript (ECMAScript)	ECMA-262	BSD3
Lua	Lua.org	MIT License
Mathematica	Wolfram	Proprietary

Contact me: @EricPickupYP on Twitter

Where?

Language	Official website	Software license
Free Pascal (Object Pascal)	www.freepascal.org	LGPL with static linking exception
Cocoa (Objective-C)	Apple	Proprietary
OCaml	Caml	LGPL
Perl	Perl.com	Artistic License, or GNU General Public License
PHP	PHP.net	PHP License
Python	python.org	Python Software Foundation License
Ruby	ruby-doc.org	GNU Library General Public License
SAP ABAP	SAP.com	Proprietary
Tcl	tcl.tk	Tcl/Tk License

Contact me: @EricPickupYP on Twitter

Who?

- Unleash your inner demon!
- Anyone who understands simplicity
- Elegant code

Contact me: @EricPickupYP on Twitter

Guilty as charged…

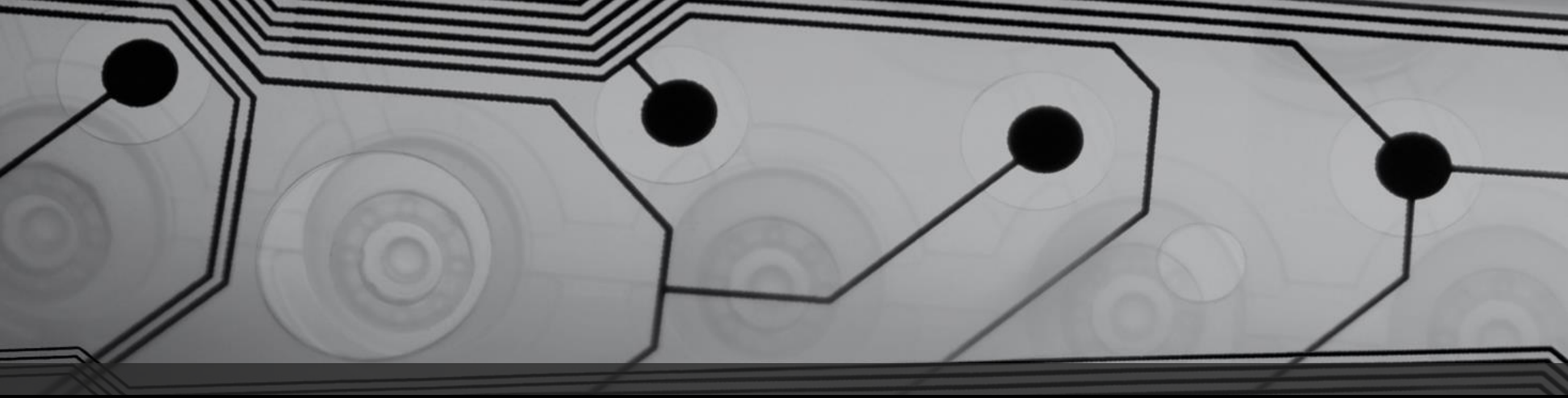
```
$string = “aa bb cc dd ee ff gg hh ii jj kk ll mm nn oo pp”;
foreach(preg_split(‘/\s/’, $string, $m)) {
      doSomething($m[1]);
}
```

Contact me: @EricPickupYP on Twitter

When?

- Input validation
- Parsing (log files, stupid html parsing, looking for subtrings in larger blocks of text)
- Searching large blocks of text
- Writing stupid micro/templating languages
- Text transformation

Contact me: @EricPickupYP on Twitter



The Guts and Glory

Ie the “HOW”

How?

Built-in Classes

Contact me: @EricPickupYP on Twitter

Built-in Classes

\d = any digit characters (0-9)

Built-in Classes

\D = any non-digit character

Built-in Classes

\w = "word" characters - any character legal in a variable name (0-9a-zA-Z_)

Built-in Classes

\W = any non-"word" character

Built-in Classes

\s = non-visible characters - spaces, tags, unix line breaks (not Windows/Mac line breaks)

Built-in Classes

\s = non-visible characters - spaces, tags, unix line breaks (not Windows/Mac line breaks)

Built-in Classes

. (dot) any character at all

How?

Custom Classes

Contact me: @EricPickupYP on Twitter

Custom Classes

[abc] = matches any a, b or c

Custom Classes

[a-p] any letter between a and p. ASCII ordering is used so E or G are not in that range

[a-pA-P] would match E and G from the previous example (along with e and g)

Custom Classes

[^a-pA-P] ^ negates in a custom class, this would match q through z and Q though Z

Custom Classes

[\w'"] you can build on built in classes. In this case a-z, A-Z, _, ' and " are matched

Custom Classes

[^\w'"] now it matches anything BUT a-z, A-Z, _, ' and "

How?

Quantifiers

Contact me: @EricPickupYP on Twitter

Quantifiers

* 0 or more of the previous character

Quantifiers

\+ 1 or more of the previous character/class

Quantifiers

{X} X of the previous character/class

Contact me: @EricPickupYP on Twitter

Quantifiers

{X,Y} minimum of X but no more than Y of the previous character

Contact me: @EricPickupYP on Twitter

Quantifiers

? makes the previous character optional

Putting it together

/[a-c]+/ one or more of the letter a, b or c

/[xyz]{5, 10}/ at least of z, y or z but no more than 10

/\s?eric/ “eric” with optional leading white space

Contact me: @EricPickupYP on Twitter

How?

Boundaries /Anchors

Contact me: @EricPickupYP on Twitter

Boundaries/Anchors

\b = word boundaries (ie beginning and end of words)

Boundaries/Anchors

^ matches at the start of a string

Boundaries/Anchors

$ matches at the end

Boundaries/Anchors

Using anchors when possible is better for performance

Contact me: @EricPickupYP on Twitter

Putting it together

/^[a-c]+/ one or more of the letter a, b or c (only at the start of a string)

/[xyz]{5, 10}$/ at least 5 of z, y or z but no more than 10 (but only at the end)

Contact me: @EricPickupYP on Twitter

How?

Alternatives

Contact me: @EricPickupYP on Twitter

Alternatives

| = OR

/Reci|eve/ will match “receive” or “receive”

/Rec[ie]ve/ is more performant

Contact me: @EricPickupYP on Twitter

Putting it together

/[abc]|[xyz]/ will match a, b or c or x, y z

/[a-cz-z] is more performant

Contact me: @EricPickupYP on Twitter

Putting it together

Use classes instead of alternatives for performance if possible

Contact me: @EricPickupYP on Twitter

How?

Groupings

Contact me: @EricPickupYP on Twitter

Groupings

/I (love|hate) you/

Groupings

/(Eric|Ann) is (beautiful|handsome|a geek)/

Groupings

Like alternatives, there is overhead.

Not as performant

Contact me: @EricPickupYP on Twitter

How?

Capture

Capture (more in examples)

- Anything in (parentheses) is captured for future use inside and outside the regex
- Ordering is based on the opening (.

How?

Modifiers

Contact me: @EricPickupYP on Twitter

Modifiers

Modifiers are appended at the end of your regex

/hello (everyone|world)/i

Contact me: @EricPickupYP on Twitter

Modifiers

/i makes the pattern match case insensitive

Modifiers

/g match all instead of the first

Modifiers

/s white space character (\s) matches new lines

Modifiers

/m the input is multiline, ^ and $ match the beginning and end of each line

Modifiers

/e execute the replacement string
(deprecated in PHP – use
preg_replace_callback)

How?

Complex Examples

Contact me: @EricPickupYP on Twitter

Complex Examples

/[\w.-]+@[\w-]\.\w{2,3}/ is a quick and dirty match for email addresses

Contact me: @EricPickupYP on Twitter

Complex Examples

/\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}/ will match an ip address

will also match 999.999.999.999 which isn't valid.

Complex Examples

```
/(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)/
```

Accurate IP4 IP address

Contact me: @EricPickupYP on Twitter

How?

Implementations

Contact me: @EricPickupYP on Twitter

Matching

```
$numMatches = preg_match('/hello (\n)\s(how are you?)?/i',
$string, array $matches);

echo $matches[1] . " " $matches[2];
```

```
var matches = text.match(/hello (\n)\s(how are you\?)$/);

alert(matches[1]) alert(matches[2])
```

Contact me: @EricPickupYP on Twitter

Splitting

```
$array = preg_split('/[|\s\d]+/',$string);
```

```
var array = string.split(/[|\s\d]+/);
```

Splits on a regular expression, useful when you can't split on a character.

Contact me: @EricPickupYP on Twitter

Replacing

```
preg_replace('/%b(\w+)%b/', '<b>$1</b>', $string);
```

```
var news = String.replace(/%b(\w+}%b', '<b>$2</b>');
```

Contact me: @EricPickupYP on Twitter

Replacing with Callbacks

```
preg_replace_callback ('/%(\w)(\w+)%(\w)/',function($m)
{….}, $string);
```

```
String.replace(/%(\w)(\w+)%(\w)/, function(tag, text,
closetag){…});
```

Contact me: @EricPickupYP on Twitter

Regular Expressions

Contact me: @EricPickupYP on Twitter