regex

regular expressions

Everywhere

```
grep -P    perl style regex
sed -r     extended regex
awk        ~ /regex/
```

Everywhere

Java, Perl, Python

JavaScript, Ruby

PHP:Programmable Hyperlinked PASTA

Ctrl + F

search: “color”

(exact matches)

Ctrl + F

colou?r

(metacharacters)

Special Chars

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

Literal Chars

\

escape character

(to match special chars)

Literal Chars

```
\

Slash\\dot\.
```

Dot

.

matches any 1 character
(very powerful)

Dot

```
file-201311..\.txt
```

Count Modifiers

? * +

allow for repetition

Count Modifiers

?

0 or 1
(optional)

Count Modifiers

?

colou?r

color colour

Count Modifiers

+

1 or more
(at least 1)

Count Modifiers

+

ap+le

aple apple appple

ale apppp pple

Count Modifiers

*

any amount
(0 or more)
(includes zero)
(did i mention zero?)

Count Modifiers

*

ap*le

aple apple appple
ale

Exact Counts

{ }

custom ranges :

{3} {2,5}

{4,} {,10}

Exact Counts

{ }

ap{3}le

appple

aple apple ale

Exact Counts

{ }

hel{2,3}o

hello helllo

helo heo

Exact Counts

`{ }`

`?`	`+`	`*`
`{0,1}`	`{1,}`	`{0,}`

Character Sets

[ ]

groups set of chars:
(treat them as a unit)

Character Sets

[ ]

gr[ea]y

grey gray

greey graay gry groy

Character Sets

```
[ ]
file-201[23]0101\.txt
```

```
file-20120101.txt
file-20130101.txt
file-20110101.txt
```

Character Sets

mix w/ count modifiers

[abcdef0123456789]+

a0c4f5f51a

zebra3000

Character Sets

mix w/ count modifiers

`[xo]{2,}`

xo ox oo xx oxoxo

x o

Character Sets

[ ]

ranges using ‘-’:

[a-z] [A-Z] [0-9]

Character Sets

[ ]

[a-f0-9]{32}

Character Sets

[ ]

[a-f0-9]{32}

MD5 strings

Character Sets

[ ]

he[1!lL]{2}[oO0]

Character Sets

[ ]

he[1!lL]{2}[oO0]

he!!o heL10 heLLO
hello he!10

Character Sets

[^]

negation sets:
matches any NOT in the set

(get more than you think)

Character Sets

```
[^ ]
file-201[^01]0101\.txt
```

file-20100101.txt

file-20110101.txt

file-20120101.txt

file-20130101.txt

Character Sets

```
[^ ]

he[^1!]{2}[^0]

hexxj help! he999 ...
```

Short hand

\d	[0-9]
\w	[A-Za-z0-9_]
\s	[\t\r\n\f]

Short hand

```
file-20130[1-6]\d\d\.txt
```

```
hello,?\s*world
```

Short hand

```
file-20130[1-6]\d\d\.txt
```

(first half of 2013*)

```
hello,?\s*world
```

Short hand

file-20130[1-6]\d\d\.txt
(first half of 2013)

hello,?\s*world
helloworld hello,world
hello world hello, world

Short hand

```
\w+@\w+\.\w+
```

Short hand

```
\w+@\w+\.\w+
```

(very weak email regex)

! Short hand

Negation Short hand	
\D	[^0-9]
\W	[^A-Za-z0-9_]
\S	[^ \t\r\n\f]

! Short hand

careful :
[\d\D]*

! Short hand

careful :
[\d\D]*

(matches everything)

Anchors

^ \b $

^ Start of the string
$ End of the string
\b “word boundaries”

Anchors

^ $

^\d+$

^[a-f\d]{32}$

^\d{3}-\d{2}-\d{4}$

^14:17:05.*

.*the end$

Anchors

\b

“whole words only”

\bword\b

\bhello, world\b

well, hello, world!

Anchors

zero width

\bhello\b..\bworld\b

well, hello, world!

Anchors

zero width

\bhello\b\bworld\b
(syntax error)

Alternate

|

OR several regexs

“or”

Alternate

|

cat|dog

cat dog

Alternate

|

```
^\d+$|^[a-f\d]{10,}$
```

Alternate

|

```
^\d+$|^[a-f\d]{10,}$
```

1230843

af765cf75c

Capture Groups

( )

“a regex within a regex”

mix w/ ? * + and |

Capture Groups

( )

Get GetValue

Capture Groups

( )

Get GetValue

Get|GetValue

Capture Groups

( )

Get(Value)?

Capture Groups

( )

Get(Value)?

Get GetValue

Capture Groups

```
( )

(I am|We are) (not )?happy
```

Capture Groups

( )

(I am|We are) (not)?happy

I am not happy
We are happy

Capture Groups

( )

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

Capture Groups

( )

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

(weak IP addr regex)

Capture Groups

( )

"capture" meaning to be able to reference it later

Backreferences

\1 \2 \3 ...
refer to a matched group

(one|toe)-to-\1

Backreferences

\1 \2 \3 ...
refer to a matched group

(one|toe)-to-\1

one-to-one toe-to-toe
one-to-toe toe-to-one

Backreferences

\1 \2 \3 ...

(x|X)(o|0)\1\2\1\2

Backreferences

\1 \2 \3 ...

(x|X)(o|O)\1\2\1\2

xoxoxo XOXOXO

xOxOxO XoXoXo

Backreferences

\1 \2 \3 ...

Non-matching vs optional

(x?)hi\1	hi	xhix
(x)?hi\1	hi	xhix

Lookarounds

Zero Width Assertions

Assert that pattern is followed or preceded by something

Lookahead

`(?=...)`
`(?!...)`

Assert you will (not) be followed

Lookahead

(?=...)

“positive lookahead”

q(?=u)

quiet quilt iraq

Lookahead

(?!...)

“negative lookahead”

q(?!u)

quiet quilt iraq

Lookahead

```
^(?=US|BR|CA)\w{2}$
```

Lookahead

```
^(?=US|BR|CA)\w{2}$
```

US BR CA

BZ RA CH

Lookahead

“6 letter word containing the word ‘cat’ ”

Lookahead

“6 letter word containing the word ‘cat’ ”

```
\b\w{6}\b
\w*cat\w*
```

Lookahead

“6 letter word containing the word ‘cat’ ”

```
(?=\b\w{6}\b)\b\w*cat\w*\b
```

Look Behind

`(?<=...)`
`(?<!...)`

asserts that not preceded by something

Look Behind

```
(?<=...)
(?<!...)
```

```
(?<!a)b
```

b

ab

Look Behind

```
(?<=...)
(?<!...)
```

```
(?<=a)b
```

ab

cb

Look arounds

(?!...)

assertion is not part of the match