

Perl 5.10

OSDC.tw 2009 4/19

scw _at_ csie dot org

<http://scw.tw/perl510.pdf>

Who

Who?

Larry Wall

Rafaël Garcia-Suarez

When

When?

Perl 5.8.0: July 18, 2002

When?

Perl 5.8.0: July 18, 2002

Perl 5.8.9: Dec 14, 2008

When?

Perl 5.10.0: Dec 18, 2007

When?

Perl 5.10.0: Dec 18, 2007
Perl 6: By **Xmas**

What

What?

Compatible (almost)
major upgrade

What?

New language features
Interpreter improvements

What?

Features and inspiration from Perl 6

Where

Where?

Debian 5.0 lenny

Ubuntu 8.10 intrepid

Fedora 9

Where?

FreeBSD ports (2009-03-28)
ActivePerl 5.10.0.1004

How

How?

```
apt-get install perl  
yum update perl
```

How?

```
portsnap fetch update  
cd lang/perl5.10  
make install clean
```

How?

```
portsnap fetch update
```

```
cd lang/perl5.10
```

```
make install clean
```

```
read /usr/ports/UPDATING !!!
```

00:48 <scw> portsnap update
00:48 <scw> cd lang/perl5.10
00:48 <scw> make install clean
00:49 <scw> 這樣是對的齣?
00:50 <rafan> 不藍
00:50 <rafan> 你是要 upgrade or ?
00:50 <rafan> portsan pfetch update
00:50 <rafan> 要 5.8 -> 5.10 請看 /usr/ports/UPDATING

Not root?

Not root!

```
$ wget http://www.cpan.org/authors/id/\
R/RG/RGARCIA/perl-5.10.0.tar.gz
```

```
$ tar zxf perl-5.10.0.tar.gz; cd perl-5.10.0
```

```
$ ./Configure -Dprefix=~/.usr/local\  
-Duserlocatableinc
```

```
$ make -j 8 all test install
```

Why





Why?

`$a // "Comment?"`

Why?

`$a // "Comment?"`

defined `$a ?`

`$a : "Comment?"`

defined-or

Apply defined-or patch against
perl 5.8 works, too

defined-or

```
my $price = mysql_select(...)  
           // "not found";  
say "Price at $price"
```

say

say = print + -l

Not enabled by default

feature

use feature qw(
say, switch, state

feature

```
use feature ":5.10"
```

```
use 5.010
```

```
use v5.10
```

```
perl -E
```

feature

switch

In correct English!!

switch

```
given($foo) {  
  when (undef) { say '$foo is undefined' }  
  when ("foo") { say '$foo is str "foo"' }  
  when ([1,3,5,7,9]) {  
    say '$foo is an odd digit'; continue;  
  }  
  when ($_ < 100) { say '$foo under 100' }  
  when (\&func) { say 'func($foo) true' }  
  default { die q(I don't know what to do
```

when

```
my $count = 0;
for (@array) {
    when ("foo") { ++$count }
}
say "array contains $count copies of 'foo'";
```

given, when & default

given(EXPR) = **do**{ \$ _ = EXPR; ... }

when(\$foo) = **when**(\$ _ ~ \$foo)

default = **when**(1 == 1)

smart match

”Smart matching in detail” in perlsyn

$\$a \sim\sim \b the same to $\$b \sim\sim \a

keys for hash

content for array

PerIRE

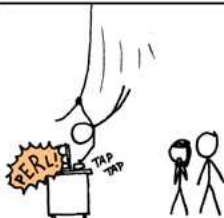
SCENARIOS WHERE IT
LETS ME SAVE THE DAY.



EVERYBODY STAND BACK.



I KNOW REGULAR
EXPRESSIONS.



Engine

De-recursiveised

Engine

Trie

Engine

Aho-Corasick

Named Capture

Named capture

```
/(?<ip> \d+\.\d+\.\d+\.\d+)  
.-.-.\[(?<time>.*?)\  
."(?<method>GET|POST)  
.(?<request>[^ ]+)  
/x
```

Named capture

```
/(?<ip> \d+\.\d+\.\d+\.\d+)  
.-.-.\[(?<time>.*?)\  
."(?<method>GET|POST)  
.(?<request>[^\s]+)  
/x  
$+{'ip', 'time', 'method', 'request'}
```

backreference

`/(\.)\1/`

`/(\.)\g1/`

`/(\.)\g{1}/`

`/(\.)\g{-1}/`

`/(?<letter>.)\g{letter}/`

`/(?<letter>.)\k<letter>/`

Regex::Keep

Regex::Keep

`s/(.*)\?.*/$1/g`

Regex::Keep

s/(.*)\?.*/\$1/g

s/.*\K\?.*/g

PerlRE **not** Regular

Recursive

```
/^(  
  <  
    (? :  
      [^ <>]+ # one or more non angle b  
      | # ... or ...  
      (?1) # recurse to bracket 1 and try  
    )* # 0 or more times.  
  > # match a closing angle bracket  
) # end capture buffer one  
$/x # end of line
```

Recursive

```
/^(  
  <  
    (? :  
      (? >  
        [^ <>]+ # one or more non angle b  
      )  
    | # ... or ...  
      (?1) # recurse to bracket 1 and try  
    )* # 0 or more times.  
  >  
) # end capture buffer one  
$/x # end of line
```

Recursive

```
/^(  
    <  
    (? :  
        [^ <>] ++ # one or more non angle b  
    |  
    (?1) # ... or ...  
    )* # recurse to bracket 1 and tr  
    > # 0 or more times.  
    > # match a closing angle bracket  
    ) # end capture buffer one  
$/x # end of line
```

Backtracking Control Verbs

(*FAIL)

(*ACCEPT)

On (*FAIL)...

On (*FAIL)...

(*PRUNE)

restart this trial (:)

On (*FAIL)...

aaaaaaaaaaaaaaaa
/a*(*PRUNE)a/

On (*FAIL)...

(*THEN)

next branch (::)

On (*FAIL)...

aaaaaaaaaaaaaaaa

/a*(*THEN)a/

On (*FAIL)...

(*COMMIT)

fail the match (:::)

On (*FAIL)...

aaaaaaaaaaaaaaaaaaaa
/a*(*COMMIT)a/

demo
parse string

man perlre

man perldelta

Thank you!