What's "new" in perl

OLUG 2015-02-03

Jay Hannah
@deafferret
jay.hannah@iinteractive.com

perl ecosystem

OO (Moose, ...)
ORM (DBIx::Class, ...)
Web (Catalyst, PSGI, ...)
I/O (JSON, XML, ...)
31,246 dists on CPAN

1987 1.0
1988 2.0
1993 4.036
1994 5.000
2000 5.6
2002 5.8
2007 5.10.0
2010 5.12.0
2011 5.14
2012 5.16
2013 5.18
2014 5.20
2015-01-20 @mhorsfall 5.21.8

perldoc perlhist

perldoc.perl.org/index-history.html

```
perldoc -f say
```

```
print “hi\n”;
```

```
use v5.10;      # 2007
say “hi”;
```

perldoc perlop
“Logical Defined-Or”

```
$name ||= ‘UNKNOWN’;    # truth
```

```
$name = defined($name) ?
   $name : ‘UNKNOWN’;
```

```
use v5.10;      # 2007
$name //= ‘UNKNOWN’;    # defined
```

perldoc perlsyn
“Switch Statements”

```
use v5.10.1;    # 2009
for ($var) {
    when (/^abc/) { $abc = 1 }
    when (/^def/) { $def = 1 }
    when (/^xyz/) { $xyz = 1 }
    default       { $nothing = 1 }
}
# see also: given, continue, break
```

perldoc perlop

“Smartmatch Operator”

```
use v5.10.1; #2009

@array = (1, 2, 3, undef, 4, 5);
say "some elements undefined"
   if undef ~~ @array;
```

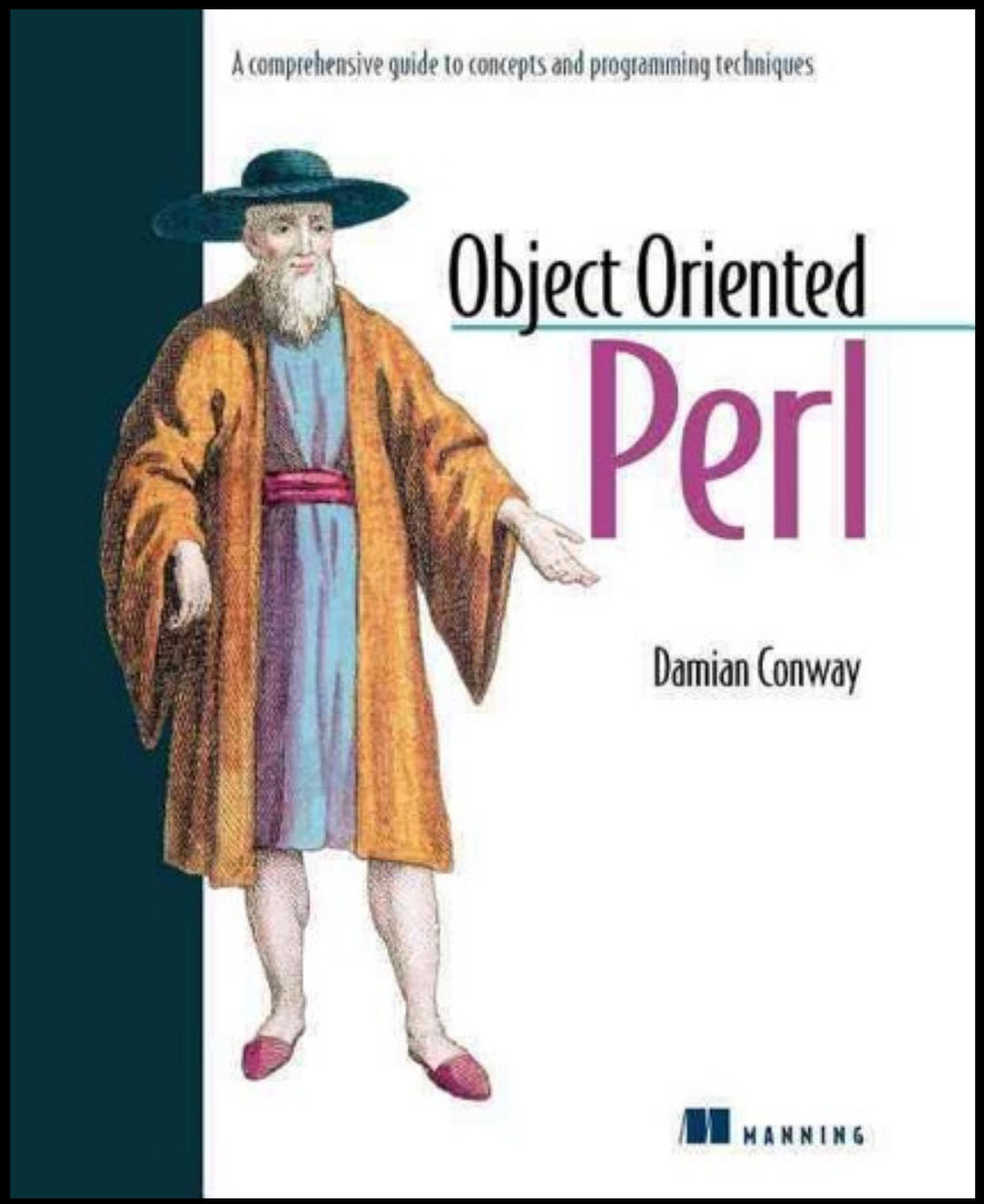
http://perldoc.perl.org/perlop.html#Smartmatch-Operator

perldoc perlre
“non-destructive substitution”

```
use v5.14;    # 2011
my $string = "Dear NAME, come to OLUG!”;
foreach my $friend (@friends) {
   my $message = $string =~ s/NAME/$friend/r;
   send($message);
}


# Perl Training Australia
# http://perltraining.com.au/tips/2012-08-08.html
```

OO Perl in 2000



perldoc perlsub

```
sub AUTOLOAD {
    my $program = $AUTOLOAD;
    $program =~ s/.*::///;
    system($program, @_);
}
```

perldoc perlootut

“OO Programming in Perl Tutorial”

2011 / 2013

By default, Perl's built-in OO system is very minimal, leaving you to do most of the work. This minimalism made a lot of sense in 1994, but in the years since Perl 5.0 we've seen a number of common patterns emerge in Perl OO. Fortunately, Perl's flexibility has allowed a rich ecosystem of Perl OO systems to flourish.

Moose, Moo, Class::Accessor, Class::Tiny, Role::Tiny

perldoc perlre

“automatic IO::File (IO::Handle)”

```
use v5.14;   # 2011
open my $fh, ">", $file;
$fh->flush;            # flush to disk right now
$fh->autoflush(1);     # Turn on auto-flushing
$fh->autoflush(0);     # Turn off auto-flushing
STDOUT->autoflush(1); # Turn on auto-flushing
                       #   for STDOUT

# Perl Training Australia
# http://perltraining.com.au/tips/2012-08-08.html
```

perldoc perlre

“Extended Patterns”

```
(?^...)  # Switch off modifiers locally

use v5.14;    # 2011
“ABC” =~ /abc/i;          # true
“ABC” =~ /a(?^:b)c/i;  # false
“ABC” =~ /a(?^i:b)c/i; # true
```

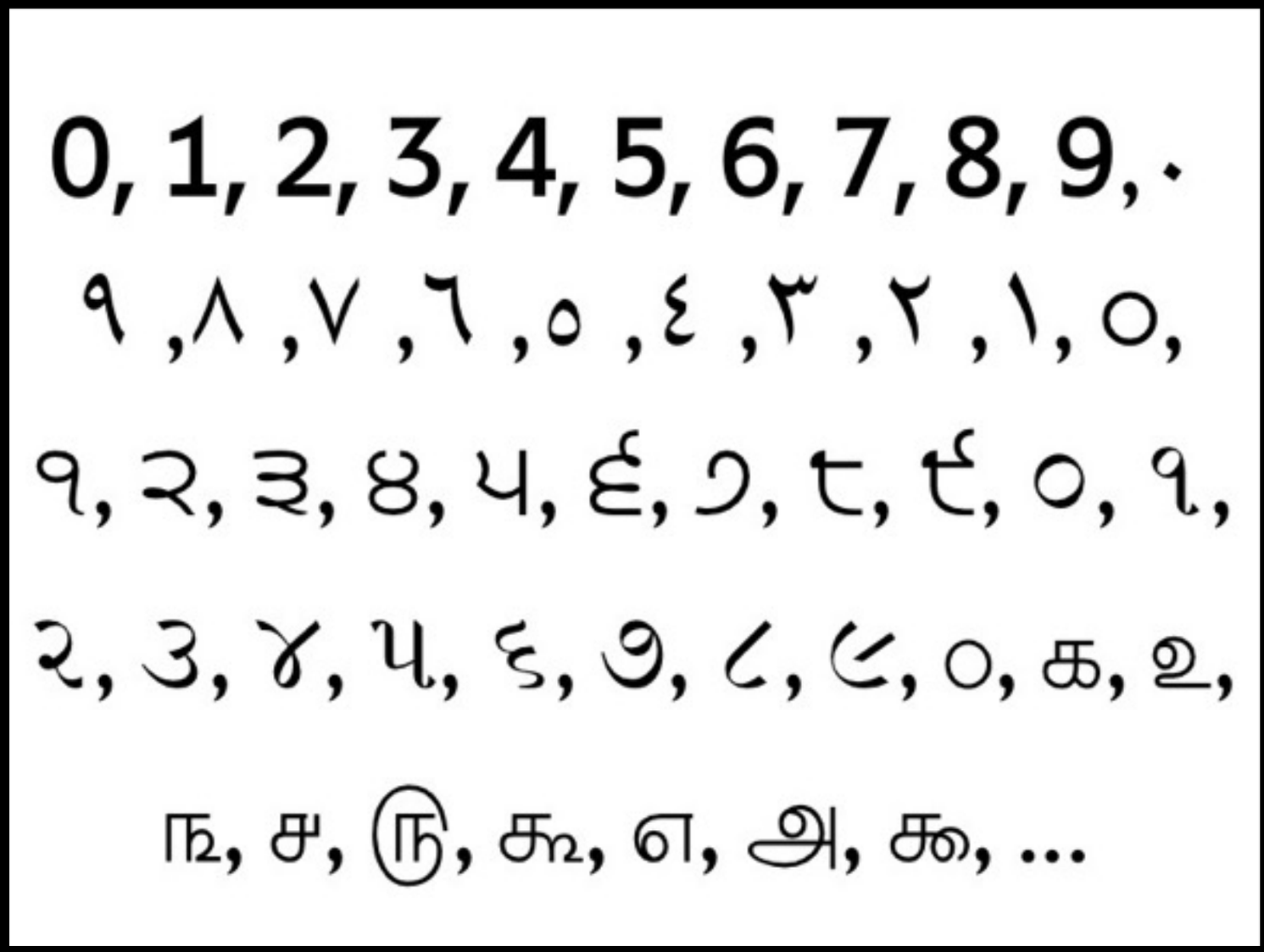
http://www.slideshare.net/andy.sh/whats-new-in-perl-514

perldoc perlre

/u means to use Unicode

```
use v5.14;
/\d/u

# any symbol
# marked DIGIT
# in Unicode
```



http://www.slideshare.net/andy.sh/whats-new-in-perl-514

perldoc perldiag

Auto de-referencing

```
my $a = [];
push @$a, 3;
```

```
use v5.14;          # 2011
no warnings "experimental::autoderef";
push $a, 7;         # pop, shift, unshift…
                    # keys, values, each…

# Would You Miss Autoderef in 5.20?
# http://www.modernperlbooks.com/mt/2013/11/would-you-miss-autoderef-in-520.html
```

perldoc perlsub

“Signatures”

```
sub foo {
  die "Too many args" unless @_ <= 2;
  die "Too few args" unless @_ >= 2;
  return $_[0] + $_[1];
}

use v5.20;  # 2014
use feature 'signatures';
sub foo ($left, $right) {
  return $left + $right;
}
```

perldoc perlref

"Postfix Dereference Syntax"

```
my $nested_array_ref = [[[[[1,2,3]]]]];

# circumfix dereference - usual way
push @{$nested_array_ref->[0]->[0]->[0]->[0]}, 4;
```

```
use v5.20;    # 2014
use experimental 'postderef';
# postfix dereference - new way
push $nested_array_ref->[0]->[0]->[0]->[0]->@*, 5;
```

http://perltricks.com/article/92/2014/5/27/Perl-v5-20---what-you-need-to-know

perldoc perldata

“Key/Value Hash Slices”

```
use v5.20;   # 2014
my %raindrops =
  (splish => 4, splash => 9, splosh => 7);
my %hash_slice =
  %raindrops{'splish', ‘splosh’};
# hash_slice is (splish => 4, splosh => 7)


# http://perltricks.com/article/92/2014/5/27/Perl-v5-20---what-you-need-to-know
```

perldoc android

perldoc perlexperiment

http://perldoc.perl.org/perlexperiment.html

perldoc perlre

```
# Delete (most) C comments
$program =~ s{
    /\*  # Match opening delim
    .*?  # Non-greedy
    \*/  # Match closing delim
}[]gsx;
```