

Perl 5.10  
in 2010

**Part I**

**The History**

**and some**

**Statistics**

**5.10.0**

**18 December 2007**

Today is  
5 March 2010



810 days  
passed



**5.10.1**

**exists already**

**5.11.0, 5.11.1,  
5.11.2, 5.11.3, 5.11.4  
exist already**

**5.12**

**soon**

# Dutch Perl Workshop 2010

*Andrew Shitov*

Perl 5.10 in 2010

*Leon Timmermans*

What's new in Perl 5.12?

*Martin Berends*

Perl 6 implementations in March 2010

*Jonathan Worthington*

Solved in Perl 6

# Total on CPAN

~80 000 modules

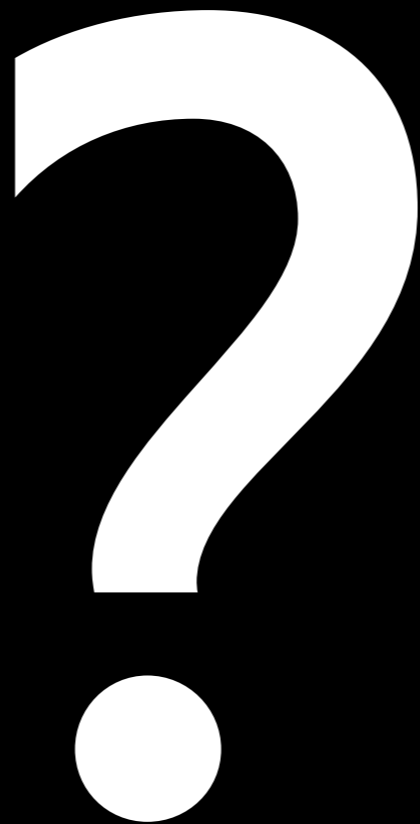
# Total on CPAN

~20 000 distributions

# Total on CPAN

~8000 authors

Are Perl 5.10  
features used  
frequently?



~ 200 modules

~ 100 authors

# Part II

# Perl 5.10 Features

say

say \$x

==

print "\$x\n"

~ ~

\$a ~ ~ /\d/

\$a ~ ~ @list

@list ~ ~ %hash

# switch

```
given($x) {  
  when(/a/) {...}  
  when('b') {...}  
  default  {...}  
}
```

# state

```
sub counter {  
    state $c = 0;  
    return ++$c;  
}
```

# regexes

(?<name>)

%+

\K

%-

\R

\g<name>

//

```
$city = $arg // 'Moscow';
```

```
$vacancy{city} // = 'Moscow';
```

# Part III

# Ontology

**How to enable**

use 5.010000;

use 5.01001;

use 5.010;

use 5.010\_000;

use 5.10.0;

use v5.10.0;

use v5.10;

use feature ':5.10';

5.10.0

v5.10.0

v5.10

vector string  
version string

abbreviated — v-string

5.10.0

v5.10.0

v5.10

vector string  
version string

abbreviated — **v-string**



v-string

Search

Web [+ Show options...](#)

Image results for **v-string** - [Report images](#)



say for debugging

```
given ($action) {
  when (/^include_cmd:/) {
    my $cmd = $child->content;
    $cmd =~ /^include_cmd:(\s*)/;
    my $ws = $1 || "";
    $cmd =~ s/^include_cmd:\s*//;
    #say("cmd:$ws$cmd");
    $cmd = cwd() . '/' . $cmd;
    @output = qx($cmd);
    $child->content($ws . join($ws, @output));
  }
}
```

// and //=  
for default values

```
$port //= 5432;  
$host //= 'localhost';
```

```
$col //= '';
```

```
$attrz{ maxjob } // = 1;
```

```
$value // = 1;
```

```
$attrz{ $_ } // = 0;
```

```
$attrz{ verbose } // = ";
```

```
$attrz{ debug } // = ";
```

```
$val // = 1;
```

```
$exit // = 0;
```

```
$skipz->{ $job_id } // = 'Skip on SIGHUP'
```

Parallel::Depend — 12 Aug 2009

Parallel-dependent dispatch of perl or shell code

```
sub import {  
  shift;  
  my %args = @_;  
  # we do not care about autoviv  
  $^H{fixedtime} = $args{epoch_offset} //  
    CORE::time;  
}
```

fixedtime — 14 Aug 2008

lexical pragma to fix the epoch offset for time related functions

```
say $answer //
```

```
"I don't know enough to answer you yet.";
```

Hailo — 29 Jan 2010

A pluggable Markov engine analogous to MegaHAL

```
my $marpa_version =  
    $Parse::Marpa::VERSION // 'undef';  
my $source_version =  
    $Parse::Marpa::Source::VERSION // 'undef';  
  
$options //= {};
```

```
my $nulling_symbol =
  $rhs_symbol->[Parse::Marpa::Internal::Symbol::NULL_ALIAS] // $rhs_symbol;

$action //= $default_action;

say {$trace_fh}
  'Problems compiling action for original rule: ',
  Parse::Marpa::brief_original_rule($rule);

my $clone = $clone_arg // 1;
my $current_parse_set = $parse_set_arg // $default_parse_set;

$choice //= 0;

$lines //= [0];
$source_options //= {};
```

```
my $trace_fh = $arg_trace_fh // (*STDERR);
```

```
my $trace_fh = shift;
```

```
$trace_fh // = *STDERR;
```

// inside return

```
return $self->_get_infection( $disease->id ) // 0;
```

```
my $val = $self->_get($key) // $default->{$key};
```

```
return @{ $self->_players // [] };
```

Games::Pandemic::City, Games::Pandemic::Config — 07 Sep 2009

Games::Risk — 18 Oct 2008

```
sub homedir {  
    my ($self) = @_;  
    require File::HomeDir;  
    return File::HomeDir->my_home  
        // croak 'File::HomeDir says you have no home  
            directory';  
}
```

```
sub config_filename {  
    my ($self) = @_;  
    return $self->{'config_filename'} // do {  
        require File::Spec;  
        File::Spec->catfile  
            ($self->homedir, '.rss2leafnode.conf');  
    };  
}
```

**return**

```
isodate_to_rfc822($date // $self->{'now822'});
```

**return** URI::Title::title

```
({ url => ($resp->request->uri // ""),  
  data => $resp->decoded_content  
  (charset => 'none')});
```

```
return inside //
```

```
my $b_time = $self->item_to_timet($b_item)  
    // return $a_item;
```

```
my $a_time = $self->item_to_timet($a_item)  
    // return $b_item;;
```

```
my $str = $self->item_to_date($item)  
    // return;
```

```
return (elt_to_email ($item->first_child('author'))
  // elt_to_email ($item ->first_child('dc:creator'))
  // elt_to_email ($item ->first_child('dc:contributor'))
  // non_empty ($item->first_child_text('wiki:username'))

  // elt_to_email ($channel->first_child('dc:creator'))
  // elt_to_email ($channel->first_child('author'))
  // elt_to_email ($channel->first_child('managingEditor'))
  // elt_to_email ($channel->first_child('webMaster'))

  // elt_to_email ($item ->first_child('dc:publisher'))
  // elt_to_email ($channel->first_child('dc:publisher'))

  // non_empty ($channel->first_child_text('title'))

  # RFC822
  // 'nobody@'.$self->uri_to_host
);
```

```
sub item_to_language {
    my ($self, $item) = @_;
    my $content;
    my $ret = (elt_to_language($item)
        // elt_to_language($item->first_child('content')));
    for (;;) {
        $item = $item->parent // last;
        $ret //= elt_to_language($item);
    }
    $ret //= $self->{'resp'}->content_language;
    return $ret;
}
```

Chained //

```
my $captures    = $arg {captures}    // [];  
my $comment     = escape $arg {comment} // $name // "";  
my $upgrade     = $arg {utf8_upgrade} // 1;  
my $downgrade  = $arg {utf8_downgrade} // 1;  
my $match       = $arg {match}       // 1;
```

when with a scalar

```
while (my ($key, $value) = each %arg) {
  given ($key) {
    when ("tests") {
      $Test -> plan ($value);
    }
    when ("import") {
      $self -> export_to_level (1, $self, $_) for @{$value || []};
    }
    default {
      die "Unknown option '$key'\n";
    }
  }
}
```

```
foreach (@hazards) {  
  when ($WUMPUS) {  
    $self -> lose;  
    push @messages => "Oops! Bumped into a Wumpus!";  
  }  
  when ($PIT) {  
    $self -> lose;  
    push @messages => "YYYIIIIIEEEEE! Fell in a pit!";  
  }  
  when ($BAT) {  
    push @messages =>  
      "ZAP! Super bat snatch! Elsewhereville for you!";  
  }  
}
```

**when for selection**

```

given ($k) {
    when ('file')    { $opt_file    = $v; }
    when ('argv')   { $opt_argv    = $v; }
    when ('inter')  { $opt_interact = $v; }
    when ('prompt') { $opt_prompt  = $v; }
    when ('quiet')  { $opt_quiet   = $v; }
    when ('tty_in') { $tty_in     = $v; }
    when ('tty_out') { $tty_out    = $v; }
    default {
        die "Error: in subroutine set_opt(), found invalid key {$k => '$v'}
            (not 'file', 'argv', 'inter', 'prompt', 'quiet',
             'tty_in' or 'tty_out')";
    }
}

```

```
given ($inp_typ)
  when ('f') . . .
  when ('a') . . .
  when ('i') . . .
  default {
    die "Internal error: type = '$inp_typ' (not 'f', 'a' or 'i')";
  }
}
```

when  
with a boolean value

```
unless ( 'itan' ~~ @list ) {
  given ( length $password ) {
    when ( 16 ) {
      # ok
    }
    when ( $_ < 4 ) {
      die('ERROR: Password is too short (Min 4 bytes required)');
    }
    when ( $_ > 16 ) {
      die('ERROR: Password is too long (Max 16 bytes allowed)');
    }
    default {
      while (1) {
        $password .= '0';
        last
          if length $password == 16;
      }
    }
  }
}
```

App::iTan::Utils — 26 Oct 2009

Secure management of iTans for online banking

```
unless ( 'itan' ~~ @list ) {
  given ( length $password ) {
    when ( 16 ) {
      # ok
    }
    when ( $_ < 4 ) {
      die('ERROR: Password is too short (Min 4 bytes required)');
    }
    when ( $_ > 16 ) {
      die('ERROR: Password is too long (Max 16 bytes allowed)');
    }
    default {
      while (1) {
        $password .= '0';
        last
          if length $password == 16;
      }
    }
  }
}
```

App::iTan::Utils — 26 Oct 2009

Secure management of iTans for online banking

when  
with regular expressions

```
sub range2list {
  my $_ = shift;
  given ($_ ) {
    when (/^\d)\-(\d)$/o ) { return "$1..$2" }
    when (/^\d\.\.\d$/o )   { return "$_" }
    when (/^\d$/o )         { return $_ }
    when (/^(.*?),(.*?)$/o ) { return range2list($1). ','
                                     .range2list($2)}
    default                  { return ""}
  }
}
```

when and ref

```
given(ref $fdef){  
  when('ARRAY'){
```

Package::FromData — 14 Jan 2008

generate a package with methods and variables from a data structure

when and undef

```
given ($1) {  
  when (undef) {return}  
  when ($left) { $depth++; }  
  when ($right) { $depth--; }  
}
```

```
given ($action) {  
  when (undef) {;} # do nothing  
                    # Right now do nothing  
                    # but find lex_q_quote  
  when ('lex_q_quote') {  
    $lexers[$ix] =  
      [ \&Parse::Marpa::Lex::lex_q_quote,  
        $prefix, $suffix ];  
  }  
}
```

Nested given/when

```
given($name) {
  when ('stream:stream') . . .
  when ('challenge') . . .
  when ('failure') . . .
  when ('stream:features') . . .
    given(my $clist = $node->getChildrenHash()) {
      when ('starttls') . . .
      when('mechanisms') . . .
        foreach($clist->{'mechanisms'}->
          [0]->getChildrenByTagName('*'))
          when($_->textContent() eq 'DIGEST-MD5'
            or $_->textContent() eq 'PLAIN')
        when('bind') . . .
        default . . .
      when ('proceed') . . .
      when ('success') . . .
    }
}
```

POE::Component::Jabber — 22 Mar 2009

A POE Component for communicating over Jabber

```
given($name) {
  when ('stream:stream') . . .
  when ('challenge') . . .
  when ('failure') . . .
  when ('stream:features') . . .
    given(my $clist = $node->getChildrenHash()) {
      when ('starttls') . . .
      when ('mechanisms') . . .
        foreach($clist->{'mechanisms'}->
          [0]->getChildrenByTagName('*'))
          when($_->textContent() eq 'DIGEST-MD5'
            or $_->textContent() eq 'PLAIN')
        when ('bind') . . .
      default . . .
    }
  when ('proceed') . . .
  when ('success') . . .
}
```

for and when

```
for ( catch ) {  
  when ( $_->isa('Getopt::Lucid::Exception::ARGV') ) {  
    say;  
    # usage stuff  
    return 1;  
  }  
  default { die $_ }  
}
```

≈ ≈

```
return _fail( $pkg, $sub ) if $_ == 0;
```

```
if ( $attr == /^Export_?Lexical$/i ) {
```

```
@exportz = grep { ! ( $_ ~~ @argz ) } @_;
```

```
$disp ~~ @exportz
```

```
or push @exportz, $disp;
```

```
@exportz = grep { ! ( $_ ~~ @argz ) } @_;
```

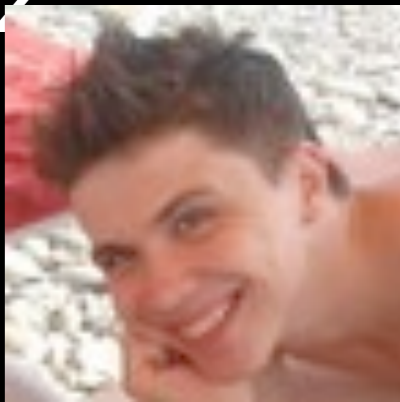
```
$disp ~~ @exportz  
or push @exportz, $disp;
```

Cool?

\$disp ~~ @exportz

or push @exportz, \$disp;

push @exportz, \$disp unless \$disp ~~ @exportz



```
for( @_ )
{
    index $_, ':'
    or next;

    if( $_ =~ @exportz )
    {
        my $source = qualify_to_ref $_, $source;
        my $install = qualify_to_ref $_, $caller;

        *$install = *$source;
    }
    else
    {
        die "Bogus $source: '$_' not exported";
    }
}
```

```
for( @_ )
{
    index $_, ':'
    or next;

    if( $_ =~ @exportz )
    {
        my $source = qualify_to_ref $_, $source;
        my $install = qualify_to_ref $_, $caller;

        *$install = *$source;
    }
    else
    {
        die "Bogus $source: '$_' not exported";
    }
}
```

**Maybe when?**

# 48 files, and only single one (Maplat::Helpers::CommandHelper) uses 5.10 feature

```
Maplat
Maplat::Helpers::BuildNum
Maplat::Helpers::CSVFilter
Maplat::Helpers::Cache::Memcached
Maplat::Helpers::Cache::Memcached::GetParser
Maplat::Helpers::CommandHelper
Maplat::Helpers::DBSerialize
Maplat::Helpers::DateStrings
Maplat::Helpers::Logo
Maplat::Helpers::MailLogger
Maplat::Helpers::Mascot
Maplat::Helpers::Padding
Maplat::Helpers::Strings
Maplat::Helpers::TextLogger
Maplat::Web
Maplat::Web::BaseModule
Maplat::Web::BrowserWorkarounds
Maplat::Web::CommandQueue
Maplat::Web::Debuglog
Maplat::Web::DirCleaner
Maplat::Web::DocsSearch
Maplat::Web::DocsSpreadSheet
Maplat::Web::DocsWordProcessor
Maplat::Web::Errors
Maplat::Web::Login
Maplat::Web::LogoCache
Maplat::Web::MemCache
Maplat::Web::MemCacheSim
Maplat::Web::PathRedirection
Maplat::Web::PostgresDB
Maplat::Web::SendMail
Maplat::Web::SessionSettings
Maplat::Web::StandardFields
Maplat::Web::StaticCache
Maplat::Web::Status
Maplat::Web::TemplateCache
Maplat::Web::UserSettings
Maplat::Web::VariablesADM
Maplat::Worker
Maplat::Worker::AdminCommands
Maplat::Worker::BaseModule
Maplat::Worker::Commands
Maplat::Worker::DirCleaner
Maplat::Worker::MemCache
Maplat::Worker::OracleDB
Maplat::Worker::PostgresDB
Maplat::Worker::Reporting
Maplat::Worker::SendMail
```

Maplat — 20 Jan 2010

The MAPLAT Web Framework

**This Module is actually a stub (don't use it)**

48 files, and only single one  
(Maplat::Helpers::CommandHelper)  
uses 5.10 feature

```
if($line->{id} ~~ %active)
```

Maplat — 20 Jan 2010

The MAPLAT Web Framework

**This Module is actually a stub (don't use it)**

# Named capturing parens

```
my $compiled_regex = qr{
  \G
  (?<mArPa_prefix>$prefix)
  (?<mArPa_match>$regex)
  (?<mArPa_suffix>$suffix)
}xms;
```

# Part IV

# Antipatterns

```
use 5.010;  
use feature ':5.10';
```

```
$show_line // = 1  
  if $style eq 'Regexp::Common';
```

```
SmartMatch::Sugar  
Regexp-CharClasses
```

```
use Switch;
```

```
switch($stream->codec_type){  
    case "video" {  
        bless $stream, 'Video::FFmpeg::AVStream::Video';  
        push @streams, $stream;  
    }  
    case "audio" {  
        bless $stream, 'Video::FFmpeg::AVStream::Audio';  
        push @streams, $stream;  
    }  
    case "subtitle" {  
        bless $stream, 'Video::FFmpeg::AVStream::Subtitle';  
        push @streams, $stream;  
    }  
    else {  
        push @streams, $stream;  
    }  
}
```

```
use if $] >= 5.011, 'deprecate';
```

```
Video::FFmpeg::AVFormat
```

```
my $prefix = $symbol_prefix // $default_prefix;  
$prefix = qr/$prefix/xms if defined $prefix;
```

```
my $suffix = $symbol_suffix // $default_suffix;  
$suffix = qr/$suffix/xms if defined $suffix;
```

```
$isPermaLink =  
  (lc($guid->att('isPermaLink') // 'true')  
  eq 'true');
```

```
$isPermaLink =  
  (lc($guid->att('isPermaLink') // 'true')  
   eq 'true');
```

```
say 'False' if length($bool_value) == 5;
```

```
given ( $params[0] // " ) {
  when (blessed $_ && $_->isa('Math::BigInt')) {
    $bit = $class->string2bit(shift(@params)->as_bin());
  }
  when (m/^\d+$/) {
    $bit = $class->int2bit(shift(@params));
  }
  when (m/^0[bB][01]+$/) {
    $bit = $class->string2bit(shift(@params));
  }
  when (m/^[01]+$/) {
    $bit = $class->bit2bit(shift(@params));
  }
}
```

Bitmask::Data — 03 Oct 2008

Handle unlimited length bitmasks in an easy and flexible way

```

while ( my ( $option, $value ) = each %{$args} ) {
  given ($option) {
    when ('rules') {
    when ('terminals') {
    when ('start') {
    when ('academic') {
    when ('default_null_value') {
    when ('default_action') {
    when ('default_lex_prefix') {
    when ('default_lex_suffix') {
    when ('ambiguous_lex') {
    when ('strip') {
    when ('trace_file_handle') {
    when ('trace_actions') {
    when ('trace_lex') {
    when ('trace_lex_tries') {
    when ('trace_lex_matches') {
    when ('trace_values') {
    when ('trace_rules') {
    when ('trace_strings') {
    when ('trace_predefineds') {
    when ('trace_iterations') {
    when ('trace_priorities') {
    when ('trace_completions') {
    when ('location_callback') {
    when ('opaque') {
    when ('cycle_action') {
    when ('cycle_depth') {
    when ('warnings') {
    when ('code_lines') {
    when ('allow_raw_source') {
    when ('max_parsing') {
    when ('version') {
    when ('semantics') {
    when ('lex_preamble') {
    when ('preamble') {
    default {

```

```
my $attrz
```

```
  = local $que->{ attrib }
```

```
  = $job2attrz{ $job_id }
```

```
  // = $que->merge_attrib( $job_id )
```

```
  ;
```

```

foreach my $shipment (@{$response->shipment}) {
    say
    "=====
=====.";
    say "| Shipment $count          |";
    say $shipment->serialize->draw;
    say """;
    if ($self->verbose) {
        say $shipment->xml->toString(1);
    }
    $count ++;
}

```

```
use 5.010000;
```

```
print "\n";
```

```
print " chapters\n";
```

```
do_something() // return NOT_FOUND;
```

```
do_something() // return NOT_FOUND;
```

```
is_leap_year()  
  ? ($n = 366)  
  : ($n = 365)  
  ;
```

# Part V

# Use cases

# Part V

# Use cases

# and what to care about

**How to enable**

```
use v5.10;
```

```
use Modern::Perl;
```

```
use common::sense;
```

```
use feature ':5.10.1';  
use feature ':5.10';
```

```
use feature '5.10.1';  
use feature '5.10';
```

**5.10 reads entire string**

```
use feature '5.10.1';  
use feature '5.10';
```

**5.10 reads entire string**

```
use feature '5.10.1';  
use feature '5.10';
```

**5.10.1 — first two parts only**

```
use feature '5.10.1';  
use feature '5.10';
```

**5.10 reads entire string**

```
use feature '5.10.1';  
use feature '5.10';
```

**5.10.1 — first two parts only**

```
use feature '5.10.7';
```

```
use everywhere q(feature ':5.10');  
use MyModule;  
MyModule->my_sub($$);
```

```
use everywhere q(feature ':5.10');  
use MyModule;  
MyModule->my_sub($$);
```

```
package MyModule;  
sub my_sub {  
    say $_[1];  
}  
1;
```

```
> perl -E "say $$;"
```

~~ to check  
if something is in the list

```
if ($last_name ~~ @attendees) {...}
```

`~~` to compare arrays

my @a = (1, 3, 5);

my @b = (1, 3, 5);

say @a ~~ @b; # 1

```
my @a = (1, 3, 5);  
my @b = (1, qr/\d/, 5);  
say @a ~~ @b; # 1
```

Note, this  
**is not**  
element by element comparision

```
my @a = (1, 3, 5);  
my @b = (1, qr/\d/, 5);  
say @a ~~ @b; # 1
```

Note, this  
is  
element by element **matching**

~~ to check  
command line arguments

```
% ./some_programme --debug -d
```

```
% ./some_programme --debug -d
```

```
say 'Debug' if '--debug' ~ @ARGV;
```

```
say 'Daemon' if '-d' ~ @ARGV;
```

```
% ./some_programme --debug -d
```

```
say 'Debug' if '--debug' ~ @ARGV;
```

```
say 'Daemon' if '-d' ~ @ARGV;
```

```
say 'Help'
```

```
  if /^(-h|--help)$/ ~ @ARGV;
```

**Chained //**

```
my $ip =  
    $ENV{X_HTTP_FORWARDED_FOR} //  
    $ENV{HTTP_X_REAL_IP} //  
    $ENV{REMOTE_ADDR};
```

```
my $ip =  
    $page->param('request_ip') //  
    $ENV{X_HTTP_FORWARDED_FOR} //  
    $ENV{HTTP_X_REAL_IP}      //  
    $ENV{REMOTE_ADDR};
```

**Easy to update**

```
my $ip =  
  $ENV{HTTP_X_REAL_IP} //  
  $page->param('request_ip') //  
  $ENV{X_HTTP_FORWARDED_FOR} //  
  $ENV{REMOTE_ADDR};
```

**Easy order control**

**state for counters**

```
sub count {  
    state $c;  
    return ++$c;  
}
```

say count(); # 1

say count(); # 2

```
state $count;
unless ($count) {
    # SQL-query select count(*) ...
}
```

```
my $offset = int rand $count;
# SQL-query
# select ... limit $offset, 1
```

for and when

```
use v5.10;
my @array = (1..20);
my $count = 0;
for(@array) {
    when(/[02468]$/) {
        $count++;
    }
    say;
}
say $count;
```

```
use v5.10;
my @array = (1..20);
my $count = 0;
for(@array) {
    when(/[02468]$/) {
        $count++;
    }
    say;
}
say $count; # 10
```

```
use v5.10;
my @array = (1..20);
my $count = 0;
for(@array) {
    when(/[02468]$/) {
        $count++;
    }
    say;
}
say $count; # 10
```

```
use v5.10;
my @array = (1..20);
my $count = 0;
for(@array) {
    when(/[02468]$/) {
        $count++;
    }
    say;
}
say $count; # 10
```

1  
3  
5  
7  
9  
11  
13  
15  
17  
19



```
use v5.10;
my @array = (1..20);
my $count = 0;
for(@array) {
    when(/[02468]$/) {
        $count++; continue;
    }
    say;
}
say $count; # 10
```

```
> perl5.10 -E \  
"say for 1..3"
```

```
> perl6 -e \  
"say for 1..3"
```

> perl5.10 -E \ 1

"say for 1..3" 2

3

> perl6 -e \ \n

"say for 1..3" \n

\n

**More**

**A number of  
fascinating and useful  
regex features**

# Part V.X

# \_\_END\_\_

Andrew Shitov  
andy@shitov.ru

talks.shitov.ru