

Perl 5.10

и 5.12

за 30 минут

Часть I

Краткая история

5.6.0 — 22 марта 2000

5.6.0 — 22 марта 2000

5.8.8 — 31 января 2006

5.6.0 — 22 марта 2000

5.8.8 — 31 января 2006

5.10.0 — 18 декабря 2007

5.6.0 — 22 марта 2000

5.8.8 — 31 января 2006

5.10.0 — 18 декабря 2007

5.12.0 — 12 апреля 2010

5.6.0 — 22 марта 2000

5.8.8 — 31 января 2006

5.10.0 — 18 декабря 2007

5.12.0 — 12 апреля 2010

6.0 — ?

5.6.0 — 22 марта 2000

5.8.8 — 31 января 2006

5.10.0 — 18 декабря 2007

5.12.0 — 12 апреля 2010

6.0 — ?

«Глупо бросать девушку,
которая стала еще красивее,
умнее, стала следить за собой
и ХОДИТЬ В СОЛЯРИЙ»

Анатолий Шарифулин





CPAN

[Home](#) · [Authors](#) · [Recent](#) · [News](#) · [Mirrors](#) · [FAQ](#) · [Feedback](#)

in All CPAN Search

- | | | | | |
|--|---|-------------------------|---|----------------------------|
| Archiving | File Name Systems | Locking | Option Parameter Config | Processing |
| Bundles (and SDKs) | Graphics | | Perl6 | |
| Commercial Software Interfaces | Internationalization | Locale | Pragmas | |
| Control Flow Utilities | Language Extensions | | Security | |
| Data and Data Types | Language Interfaces | | Server Daemon Utilities | |
| Database Interfaces | Mail and Usenet News | | String Language Text Processing | |
| Development Support | Miscellaneous | | User Interfaces | |
| Documentation | Networking Devices IPC | | World Wide Web | |
| File Handle Input/Output | Operating System Interfaces | | | |



Всего на СРАП

~80 000 модулей

Всего на СРАН

~20 000 дистрибутивов

Всего на СРАН

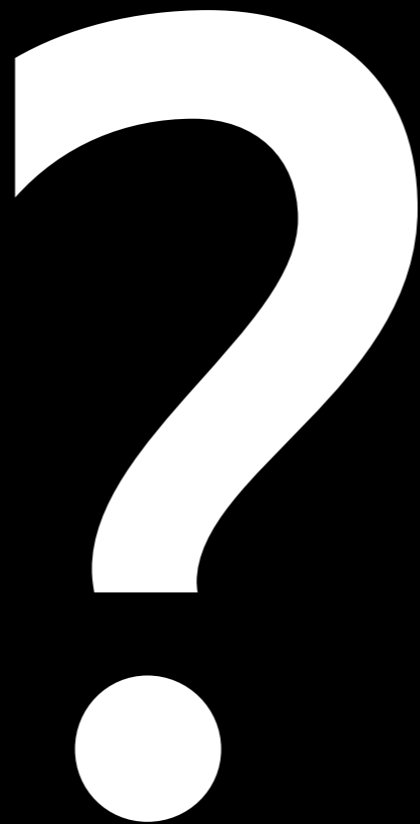
~8000 авторов

Всего на СРАН

~8000 авторов

~4800 АКТИВНЫХ

Как часто
используют
фичи Perl 5.10?



Пошел третий год

**~ 200 модулей
используют 5.10**

~ 100 авторов
ИСПОЛЬЗУЮТ 5.10

Часть II

Фичи Perl 5.10

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';
```

Кроме того

**Много интересных
НОВЫХ ВОЗМОЖНОСТЕЙ
регулярных выражений**

Часть III

Фи́чи Perl 5.12

~~\$VERSION~~

package Earth::Gravity 9.8;

use Earth::Gravity 9.8;

... (Yada Yada)

```
sub planned_functionality {
```

```
  ...
```

```
}
```

\N

"abc\ndef" ~ ~ /(\N+)/;

each @a

while ((\$a, \$b) = each @a)

} when

say \$result when \$ready;

Часть IV

Как пишут другие

Как включить

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

сокращенно — v-string



v-string

Search

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

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



say для отладки

```
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));
  }
}
```

// и // =

для значений по умолчанию

```
$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.";
```

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

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

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

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

// внутри return

```
return $self->_get_infection( $disease->id ) // 0;  
  
my $val = $self->_get($key) // $default->{$key};  
  
return @{ $self->_players // [] };
```

return

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

return URI::Title::title

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

return внутри //

```
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
);
```

```
return (elt_to_email ($item->first_child('au
// elt_to_email ($item ->first_child(
// elt_to_email ($item ->first_child(
// non_empty ($item->first_child_text

// elt_to_email ($channel->first_child
// elt_to_email ($channel->first_child
// elt_to_email ($channel->first_child
// elt_to_email ($channel->first_child

// elt_to_email ($item ->first_child(
```

```
// non_empty ($channel->first_child  
  
# RFC822  
// 'nobody@'.$self->uri_to_host  
);
```

Несколько //

```
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 со скалярром

```
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";  
        }  
    }  
}
```

```
} Test::Regexp — 09 Dec 2009  
Test your regular expressions
```

```
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 для выбора варианта

```
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')}";  
    }  
}
```

Term::DBPrompt — 18 Dec 2009

Commandline prompt for a database application

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

when
с булевым выражением

```
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 {  
      . . .  
    }  
  }  
}
```

```
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 {  
      . . .  
    }  
  }  
}
```

when

с регулярным выражением

```
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 и ref

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

Package::FromData — 14 Jan 2008

generate a package with methods and variables from a data structure

when n 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 ];  
  }  
}
```

Вложенные 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

for n when

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

22

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

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

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

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

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

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

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

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

Именованные сохраняющие скобки

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

Часть V

АНТИПАТТЕРНЫ

```
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
```

```
use Switch;
```

```
switch($stream->codec_type){
```

```
  case "video" {
```

```
    bless $stream, 'Video::FFmpeg::AVStream';
```

```
    push @streams, $stream;
```

```
  }
```

```
  case "audio" {
```

```
    bless $stream, 'Video::FFmpeg::AVStream';
```

```
    push @streams, $stream;
```

```
  }
```

```
  case "subtitle" {
```

```
    bless $stream, 'Video::FFmpeg::AVStream';
```

```
    push @streams, $stream;
```

```
  }
```

```
    }  
    else {  
        push @streams, $stream;  
    }  
}
```

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

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

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

```
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_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_parses') {
    when ('version') {
    when ('semantics') {
    when ('lex_preamble') {
```

Parse::Marpa::Internal

```
my $attrz
```

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

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

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

```
    ;
```

```
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)  
  ;
```

Часть VI

Где применить

Часть VI

Где применить

и как не наступить на грабли

Как включить

```
use v5.10;
```

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

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

```
use v5.10;  
use strict;
```

vs.

```
use v5.12;
```

```
use v5.10;  
use strict;
```

vs.

```
use v5.12;
```

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

~~ для проверки
вхождения в список

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

~~ для сравнения списков

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
```

Осторожно, это не поэлементное сравнение

~~ для проверки аргументов

```
% ./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;
```

Цепочка //

```
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};
```

Легко обновлять

state для счетчиков

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

say count(); # 1

say count(); # 2

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

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

for n 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
```

```
use v5.12;
```

```
for ('a'..'z', 'A', 'Z') {  
    say when /[aeiou]/i;  
}
```

use v5.12;

~~\$string =~ /(.+)/s;~~

\$string =~ /(\N+)/;

```
> 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

Кроме того

**Много интересных
НОВЫХ ВОЗМОЖНОСТЕЙ
регулярных выражений**

__END__

Андрей Шитов
andy@shitov.ru

talks.shitov.ru