ТЕХНОСФЕРА

Модульность и повторное использование

Содержание

1. **"include"**
2. Блоки фаз
3. `package`
4. Экспорт
5. Версии
6. Pragmatic modules
7. `no`
8. Symbol Tables
9. CPAN
10. ДЗ

eval

```
my $u;

eval '
  $u = 5;
  my $y = 10;
  sub m_3 {
    my ($x) = @_;
    return $x * 3;
  }
';

$u; # 5
$y; # Undefined
m_3(2); # 6
```

do

```
do 'sqr.pl';
```

```
# sqr.pl
$u = 5;
my $y = 10;
sub m_3 {
  my ($x) = @_;
  return $x * 3;
}
```

```
$u; # 5
$y; # Undefined
m_3(2); # 6
```

require

```
require 'sqr.pl';
require Local::Sqr; # Local/Sqr.pm
```

```
# Local/Sqr.pm
$u = 5;
my $y = 10;
sub m_3 {
  my ($x) = @_;
  return $x * 3;
}

1; # return value!
```

```
$u; # 5
$y; # Undefined
m_3(2); # 6
```

Файл модуля

```
require Module; # Module.pm
require Module::My # Module/My.pm
```

Поиск модулей

```
perl -e 'print join "\n", @INC'
/etc/perl
/usr/local/lib/perl/5.14.2
/usr/local/share/perl/5.14.2
/usr/lib/perl5
/usr/share/perl5
/usr/lib/perl/5.14
/usr/share/perl/5.14
/usr/local/lib/site_perl
```

```
$ PERL5LIB=/tmp/lib perl ...
$ perl -I /tmp/lib ...
```

Содержание

1. "include"
2. **Блоки фаз**
3. `package`
4. Экспорт
5. Версии
6. Pragmatic modules
7. `no`
8. Symbol Tables
9. CPAN
10. ДЗ

BEGIN

```
BEGIN {
  require Some::Module;
}

sub test1 {
  return 'test1';

  sub test2 {
    return 'test2';

    BEGIN {...}
  }
}
```

END

```
open(my $fh, '>', $file);

while (1) {
  # ...
}

END {
  close($fh);
  unlink($file);
}
```

Другие блоки

```
CHECK {}
UNITCHECK {}
INIT {}

${^GLOBAL_PHASE}
```

use Module;

```perl
use My_module;        # My_module.pm
use Data::Dumper;     # Data/Dumper.pm
BEGIN { push(@INC, '/tmp/lib'); }
use Local::Module;    # Local/Module.pm
```

```perl
sub sqr {
  my ($number) = @_;

  return $number ** 2;
}

my $load_time = time();

1; # return value!
```

Содержание

1. "include"
2. Блоки фаз
3. `package`
4. Экспорт
5. Версии
6. Pragmatic modules
7. `no`
8. Symbol Tables
9. CPAN
10. ДЗ

package

```
package Local::Multiplier;

sub m2 {
  my ($x) = @_;
  return $x * 2;
}

sub m3 {
  my ($x) = @_;
  return $x * 3;
}
```

```
use Local::Multiplier;

print Local::Multiplier::m3(8); # 24
```

package — inline

```
{
  package Multiplier;
  sub m_4 { return shift() * 4 }
}

print Multiplier::m_4(8); # 32
```

__PACKAGE__

```
package Some;

print __PACKAGE__; # Some
```

our

```
{
  package Some;
  my $x = 1;
  our $y = 2; # $Some::y;

  our @array = qw(foo bar baz);
}

print $Some::x; # ''
print $Some::y; # '2'

print join(' ', @Some::array); # 'foo bar baz'
```

my, state

```
my $x = 4;
{
  my $x = 5;
  print $x; # 5
}
print $x; # 4
```

```
use feature 'state';

sub test {
  state $x = 42;
  return $x++;
}

printf(
  '%d %d %d %d %d',
  test(), test(), test(), test(), test()
); # 42 43 44 45 46
```

main package

```perl
our $size = 42;

sub print_size {
  print $main::size;
}

package Some;
main::print_size(); # 42
```

use Module LIST;

```perl
use Local::Module ('param1', 'param2');
use Another::Module qw(param1 param2);
```

```perl
BEGIN {
  require Module;
  Module->import(LIST);
  # ~ Module::import('Module', LIST);
}
```

```perl
use Module ();
# BEGIN { require Module; }
```

Содержание

1. "include"
2. Блоки фаз
3. `package`
4. **Экспорт**
5. Версии
6. Pragmatic modules
7. `no`
8. Symbol Tables
9. CPAN
10. ДЗ

Экспорт

```perl
use File::Path qw(make_path remove_tree);

# File::Path::make_path
make_path('foo/bar/baz', '/zug/zwang');

# File::Path::remove_tree
remove_tree('foo/bar/baz', '/zug/zwang');
```

Exporter

```
package Local::Multiplier;

use Exporter 'import';
our @EXPORT = qw(m2 m3 m4 m5 m6);

sub m2 { shift() ** 2 }
sub m3 { shift() ** 3 }
sub m4 { shift() ** 4 }
sub m5 { shift() ** 5 }
sub m6 { shift() ** 6 }
```

```
use Local::Multiplier;

print m3(5); # 125
print Local::Multiplier::m3(5); # 125
```

Exporter — EXPORT_OK

```perl
package Local::Multiplier;

use Exporter 'import';
our @EXPORT_OK = qw(m2 m3 m4 m5 m6);

sub m2 { shift() ** 2 }
sub m3 { shift() ** 3 }
sub m4 { shift() ** 4 }
sub m5 { shift() ** 5 }
sub m6 { shift() ** 6 }
```

```perl
use Local::Multiplier qw(m3);

print m3(5); # 125
print Local::Multiplier::m4(5); # 625
```

%EXPORT_TAGS

```perl
our %EXPORT_TAGS = (
  odd  => [qw(m3 m5)],
  even => [qw(m2 m4 m6)],
  all  => [qw(m2 m3 m4 m5 m6)],
);
```

```perl
use Local::Multiplier qw(:odd);

print m3(5);
```

Содержание

1. "include"
2. Блоки фаз
3. `package`
4. Экспорт
5. **Версии**
6. Pragmatic modules
7. `no`
8. Symbol Tables
9. CPAN
10. ДЗ

use Module VERSION;

```perl
package Local::Module;

our $VERSION = 1.4;
```

```perl
use Local::Module 1.5;
```

```
$ perl -e 'use Data::Dumper 500'
Data::Dumper version 500 required--
this is only version 2.130_02 at -e line 1.
BEGIN failed--compilation aborted at -e line 1.
```

sub VERSION

```perl
use Local::Module 500;
# Local::Module->VERSION(500);
# ~ Local::Module::VERSION('Local::Module', 500);
```

```perl
package Local::Module;

sub VERSION {
  my ($package, $version) = @_;

  # ...
}
```

v-strings

```perl
use Local::Module v5.11.133;
```

```
v102.111.111; # 'foo'
102.111.111;  # 'foo'
v1.5;
```

use VERSION;

```
use 5.12.1;
use 5.012_001;

$^V # v5.12.1
$]  # 5.012001
```

Содержание

1. "include"
2. Блоки фаз
3. `package`
4. Экспорт
5. Версии
6. **Pragmatic modules**
7. `no`
8. Symbol Tables
9. CPAN
10. ДЗ

Pragmatic modules

```
use strict;
use warnings;
```

use strict 'refs';

```perl
use strict 'refs';

$ref = \$foo;
print $$ref;  # ok
$ref = "foo";
print $$ref;  # runtime error; normally ok
```

use strict 'vars';

```perl
use strict 'vars';

$Module::a;
my  $a = 4;
our $b = 5;
```

use strict 'subs';

```
use strict 'subs';
```

```
print Dumper [test]; # 'test'
```

```
sub test {
  return 'str';
}
print Dumper [test]; # 'str'
```

use warnings

```perl
use warings;
use warnings 'deprecated';
```

```
$ perl -e 'use warnings; print(5+"a")'
Argument "a" isn't numeric in addition (+) at -e line 1.
```

```
$ perl -we 'print(5+"a")'
Argument "a" isn't numeric in addition (+) at -e line 1.
```

use diagnostics;

```perl
use diagnostics;
```

```
$ perl -e 'use diagnostics; print(5+"a")'
Argument "a" isn't numeric in addition (+) at -e line 1 (
    (W numeric) The indicated string was fed as an argume
    that expected a numeric value instead.  If you're for
    will identify which operator was so unfortunate.
```

use lib;

```perl
use lib qw(/tmp/lib);

BEGIN { unshift(@INC, '/tmp/lib') }
```

FindBin

```perl
use FindBin qw($Bin);
use lib "$Bin/../lib";
```

use feautre;

```perl
use feature qw(say);

say 'New line follows this';
```

use bignum;

```
use bignum;
use bigint;
use bigrat;
```

```
$ perl -E 'use bigint; say 500**50'
8881784197001252323389053344726562500000000000000000000000

$ perl -E 'say 500**50'
8.8817841970012 5e+134
```

Содержание

1. "include"
2. Блоки фаз
3. `package`
4. Экспорт
5. Версии
6. Pragmatic modules
7. `no`
8. Symbol Tables
9. CPAN
10. ДЗ

no Module;

```
no Local::Module LIST;

# Local::Module::unimport('Local::Module', LIST);
```

no VERSION;

```
no 5.010;
```

no pragma;

```perl
no strict;
no feature;
```

Содержание

1. "include"
2. Блоки фаз
3. `package`
4. Экспорт
5. Версии
6. Pragmatic modules
7. `no`
8. **Symbol Tables**
9. CPAN
10. ДЗ

Symbol Tables

```perl
{
  package Some::Package;
  our $var = 500;
  our @var = (1,2,3);
  our %func = (1 => 2, 3 => 4);
  sub func { return 400 }
}


use Data::Dumper;
print Dumper \%Some::Package::;
```

```
$VAR1 = {
          'var' => *Some::Package::var,
          'func' => *Some::Package::func
        };
```

Typeglob

```
                       +-------> SCALAR - $bar
                       |
                       +-------> ARRAY  - @bar
                       |
                       +-------> HASH   - %bar
                       |
Foo:: -----> bar  -----+-------> CODE   - &bar
                       |
                       +-------> IO     - bar (FH)
                       |
                       +-------> GLOB   - *bar
```

Typeglob — операции

```
*Some::Package::foo = *Some::Package::var

*Some::Package::foo = \$bar;
*Some::Package::foo = \@bar;

*Some::Packge::func = sub { ... }
```

caller

```
# 0         1           2
($package, $filename, $line) = caller;
```

```
(
    $package,    $filename,   $line,
    $subroutine, $hasargs,    $wantarray,
    $evaltext,   $is_require, $hints,
    $bitmask,    $hinthash
) = caller($i);
```

AUTOLOAD

```perl
{
  package Some::Package;

  sub AUTOLOAD {
    our $AUTOLOAD;
    return $AUTOLOAD;
    # return $Some::Package::AUTOLOAD;
  }
}

print Some::Package::foo();
# 'Some::Package::foo'

print Some::Package::test();
# 'Some::Package::test'
```

local

```perl
{
  package Test;
  our $x = 123;

  sub bark { print $x }
}

Test::bark(); # 123
{
  local $Test::x = 321;
  Test::bark(); # 321
}
Test::bark(); # 123
```

local — варианты

```
# localization of values
local $foo;                    # make $foo dynamically
local (@wid, %get);            # make list of variables
local $foo = "flurp";          # make $foo dynamic, and
local @oof = @bar;             # make @oof dynamic, and
local $hash{key} = "val";      # sets a local value for
delete local $hash{key};       # delete this entry for 
local ($cond ? $v1 : $v2);     # several types of lvalu

# localization of symbols
local *FH;                     # localize $FH, @FH, %FH
local *merlyn = *randal;       # now $merlyn is really 
                               #     @merlyn is really 
local *merlyn = 'randal';      # SAME THING: promote 'r
local *merlyn = \$randal;      # just alias $merlyn, no
```

Содержание

1. "include"
2. Блоки фаз
3. `package`
4. Экспорт
5. Версии
6. Pragmatic modules
7. `no`
8. Symbol Tables
9. **CPAN**
10. ДЗ

CPAN

The Comprehensive Perl Archive Network

http://cpan.org

Metacpan

http://metacpan.org

meta::cpan

Установка из пакета в Debian

```
$ apt-cache search libjson-perl
libjson-perl - module for manipulating
  JSON-formatted data
libjson-pp-perl - module for manipulating
  JSON-formatted data (Pure Perl)
libjson-xs-perl - module for manipulating
  JSON-formatted data (C/XS-accelerated)

$ apt-get install libjson-perl
```

Установка из пакета в CentOS

```
$ yum search perl-json
======================== Matched: perl-json =======
perl-JSON-XS.x86_64 : JSON serialising/deserialisin
  done correctly and fast
perl-JSON.noarch : Parse and convert to JSON
  (JavaScript Object Notation)
perl-JSON-PP.noarch : JSON::XS compatible pure-Perl

$ yum install perl-JSON-XS
```

Утилита cpan

```
$ cpan
Terminal does not support AddHistory.

cpan shell -- CPAN exploration and modules installa
Enter 'h' for help.
```

```
$ cpan install JSON
```

```
perl -MCPAN -e shell
```

Утилита cpanm

```
curl -L https://cpanmin.us | \
    perl - --sudo App::cpanminus
```

```
cpanm Data::Printer
cpanm MIYAGAWA/Plack-0.99_05.tar.gz
cpanm ~/dists/MyCompany-Enterprise-1.00.tar.gz
```

cpantesters

module-starter

```
module-starter --module Local::PerlCourse
  --author Vadim --email vadim@pushtaev.ru
```

```
$ tree Local-PerlCourse/
Local-PerlCourse/
├── Changes
├── ignore.txt
├── lib
│   └── Local
│       └── PerlCourse.pm
├── Makefile.PL
├── MANIFEST
├── README
└── t
    ├── 00-load.t
    ├── boilerplate.t
    ├── manifest.t
    ├── pod-coverage.t
    └── pod.t
```

ExtUtils::MakeMaker

```
use 5.006;
use strict;
use warnings;
use ExtUtils::MakeMaker;

WriteMakefile(
    NAME                => 'Local::PerlCourse',
    AUTHOR              => q{Vadim <vadim@pushtaev.
    VERSION_FROM        => 'lib/Local/PerlCourse.pm
    ABSTRACT_FROM       => 'lib/Local/PerlCourse.pm
    ($ExtUtils::MakeMaker::VERSION >= 6.3002
      ? ('LICENSE'=> 'perl')
      : ()),
    PL_FILES            => {},
    PREREQ_PM => {
        'Test::More' => 0,
    },
    dist                => { COMPRESS => 'gzip -9f'
    clean               => { FILES => 'Local-PerlCo
);
```

Module::Install

```perl
use inc::Module::Install;

# Define metadata
name           'Your-Module';
all_from       'lib/Your/Module.pm';

# Specific dependencies
requires       'File::Spec'   => '0.80';
test_requires  'Test::More'   => '0.42';
recommends     'Text::CSV_XS'=> '0.50';
no_index       'directory'    => 'demos';
install_script 'myscript';

WriteAll;
```

Module::Build

```
use Module::Build;
my $build = Module::Build->new
  (
   module_name => 'Foo::Bar',
   license  => 'perl',
   requires => {
                 'perl'          => '5.6.1',
                 'Some::Module'  => '1.23',
                 'Other::Module' => '>= 1.2, != 1.
                },
  );
$build->create_build_script;
```

```
perl Build.PL
./Build
./Build test
./Build install
```

Содержание

1. "include"
2. Блоки фаз
3. `package`
4. Экспорт
5. Версии
6. Pragmatic modules
7. `no`
8. Symbol Tables
9. CPAN
10. **ДЗ**

ДЗ 3.1

```perl
# http://jsonlines.org/
# use JSON;

use Local::PerlCourse::JSONL qw(
  encode_jsonl
  decode_jsonl
);

$string = encode_jsonl($array_ref);
$array_ref = decode_jsonl($string);
```

ДЗ 3.2

```perl
use Local::PerlCourse::Currency qw(set_rate);

set_rate(
  usd => 1,
  rur => 65.44,
  eur => 1.2,
  # ...
);

$rur = Local::PerlCourse::Currency::usd_to_rur(42);
$cny = Local::PerlCourse::Currency::gbp_to_cny(30);
```

ДЗ 3.3

```perl
package Local::SomePackage;

use Local::PerlCourse::GetterSetter qw(x y);
# scalar only

set_x(50);
$Local::SomePackage::x; # 50

our $y = 42;
get_y(); # 42
set_y(11);
get_y(); # 11
```