

Perl 6

by example

A talk about
a kind of philosophy
of Perl 6
learning process

proto.perl6.org


Proto - A module installer for Perl 6

http://proto.perl6.org/

Google

How to get Rakudo Perl 6 | Rakud...Proto - A module installer for Perl 6

Proto - A module installer for Perl 6

































Proto is a hyper-lightweight dependency tracking and module installation system. Its only purpose is to help you set up a running environment where you can play with Perl 6 modules with minimal hassle. It is not to be taken too seriously. It is not a full-fledged module installation system, as Parrot's [Plumage](#) aims to be. It is a prototype; in fact, its very name was chosen to indicate this fact. It is currently trying to prototype ideas for the loadable module ecosystem of [Rakudo*](#), which can also handle multiple copies of modules distinguished by :auth and :ver name parts.

The proto source code can be found [on github](#). The [README](#) file documents its usage.

Project list

[JSON version of this list.](#)

ABC	ABC music notation tools for Perl 6	  
Algorithm-Viterbi	Perl 6 HMM decoder	  
benchmark	Simple Perl 6 version of Benchmark.pm	  
bioperl6	reimplementation of BioPerl classes in Perl6	  
csv	A parser for CSV (comma-separated values) files for Perl 6	  
druid	A connection-oriented board game written in Perl 6	  
fakedbi	a subset of Perl 5 DBI ported to Perl 6 to use while experts build the Real Deal	  
faz	Faz action dispatching framework	  
form	A Perl 6 implementation of Perl 6-style text formatting	  
gamebase	A Perl 6 based 2D game engine	  

*Proto is a hyper-
lightweight
dependency tracking
and module
installation system*

pls is its
new name

We don't care
of all that

What we
do care of are

54

Perl 6 projects

are on proto.perl6.org

history.back()

2003

perl6.ru

launched

Июньский релиз Rakudo — «Kiev»

30-й релиз [Rakudo Perl 6](#) вышел под кодовым названием «Kiev».

Решение о названии принято после того, как [на хакмите в Киеве](#) в разработку было успешно вовлечено несколько желающих (никогда раньше этим не занимавшихся) и сделано несколько коммитов.

Сейчас компилятор проходит 33 378 тестов из спецификации, что составляет около 83% их общего числа.

[rakudo, kiev](#) — 18 июня 2010

[От Perl 5 к Perl 6 — введение](#)

Perl 6 еще недостаточно документирован. Не удивительно, потому что (в отличие от спецификации) написание компилятора для Perl 6 представляется гораздо более приоритетной задачей, чем написание документации, ориентированной на пользователя.

[От Perl 5 к Perl 6 — Строки, массивы, хеши](#)

Perl 6 — почти как Perl 5, только лучше. Инструкции разделяются точками с запятыми. После последней инструкции в блоке или после закрывающей фигурной скобки в конце строки точка с запятой не обязательна. Имена переменных, как и раньше, начинаются с сигиллов (например, \$, @, %), и многие встроенные функции Perl 5 почти не были изменены в Perl 6.

[Карл Мзак про Ракудо](#)

Тех, кто уже сегодня собирается использовать Rakudo, ждет несколько сюрпризов. Сейчас доступны совсем немногие возможности Perl 6. Но некоторые из них уже сами по себе являются поводом для того, чтобы развернуть Rakudo, а некоторые по-настоящему приводят в восторг — настолько, что кажется, что писать обычный код уже никогда не будет интересно.



[Интервью с Дамианом Конвеем](#)

Конференция O'Reilly по программному обеспечению с открытым кодом, OSCON, столь же славится умными беседами в кулуарах и за обедами, сколько и семинарами, заседаниями и программными выступлениями. Терри Камерленго пригласил на ненавязчивую беседу Дамиана Конвея, автора «Perl Best Practices» и «Perl Hacks», чтобы узнать о его мыслях касательно самых разных тем, начиная с подробностей про Perl 6 и заканчивая тем, что по его мнению важно для следующего поколения ученых в области компьютерных наук.



It observed
the code from
parrot/languages/
perl6/examples
folder

Record #15

Example research.

loop()

mandel.p6

was the most exciting

+

Journal Pre-proof

• • •

• • •

Вот так выглядит результат выполнения программы:

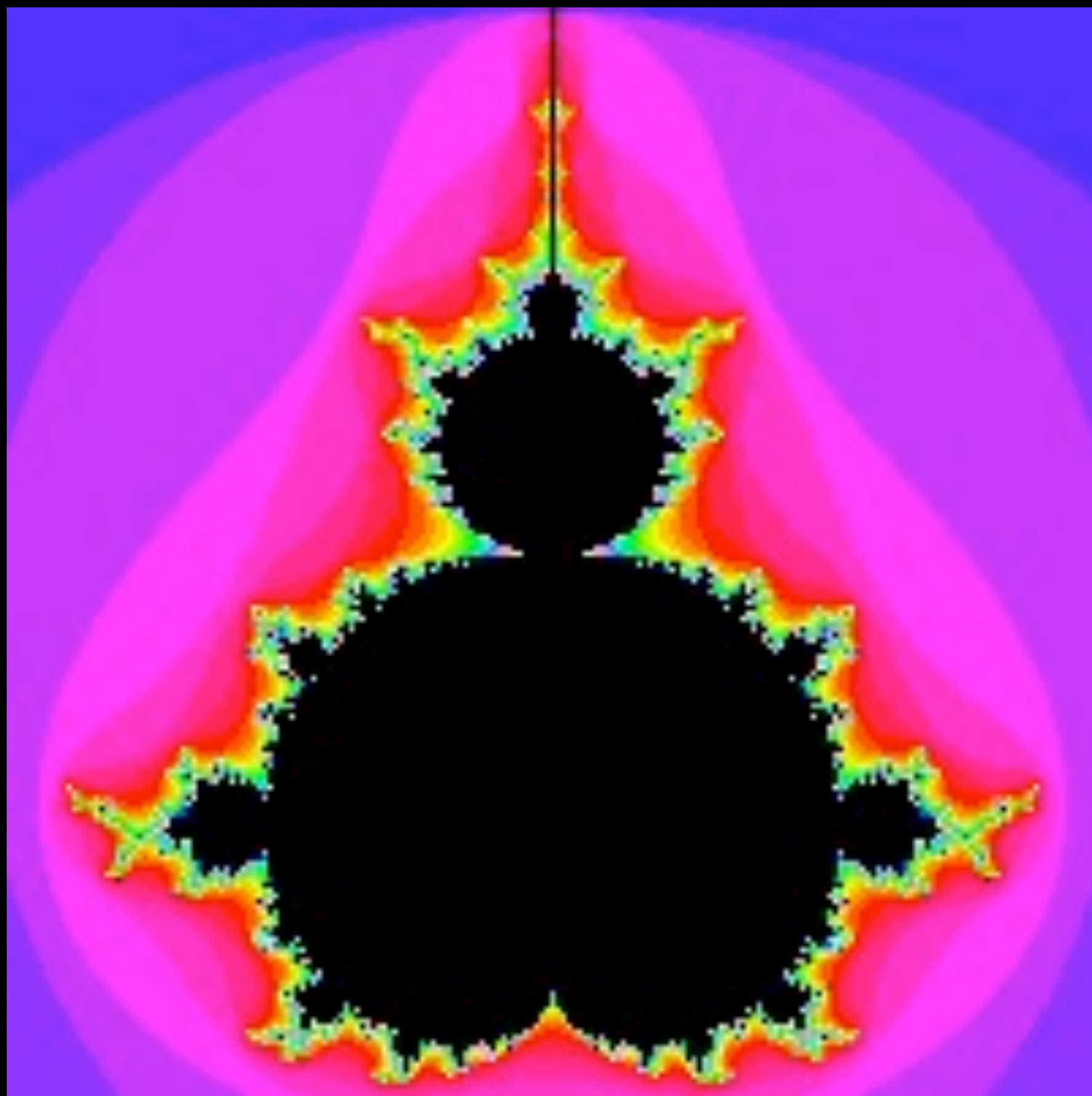
[illegible]


```
loop ($y=30; $C = $y*0.1 - 1.5, $y--;) {  
    loop ($x=0; $c = $x*0.04 - 2.0, $z=0.0,  
        $Z=0.0, $x++ < 75;) {  
        loop ($r=$c, $i=$C, $k=0;  
            $t = $z*$z - $Z*$Z + $r,  
            $Z = 2.0*$z*$Z + $i, $z=$t, $k<112;  
            $k++) {
```

...

[illegible]

history.now()



The method:
try to understand
what you
don't understand

You'll find
impressive things

mandelbrot project

Mandelbrot set in Perl 6

my \$height = @*ARGS[0] // 31;

my \$width = \$height;

my \$max_iterations = 50;

my \$upper-right = $-2 + (5/4)i$;

my \$lower-left = $1/2 - (5/4)i$;


```
my $height = @*ARGS[0] // 31;  
my $width = $height;  
my $max_iterations = 50;  
  
my $upper-right = -2 + (5/4)i;  
my $lower-left = 1/2 - (5/4)i;
```

Global variable and defined-or

```
my $height = @*ARGS[0] // 31;  
my $width = $height;  
my $max_iterations = 50;  
  
my $upper-right =  $-2 + (5/4)i$ ;  
my $lower-left =  $1/2 - (5/4)i$ ;
```

Complex numbers (wow!)

```
sub mandel(Complex $c) {  
    my $z = 0i;  
    for ^$max_iterations {  
        $z = $z * $z + $c;  
        return 1 if ($z.abs > 2);  
    }  
    return 0;  
}
```

```
sub mandel(Complex $c) {  
    my $z = 0i;  
    for ^$max_iterations {  
        $z = $z * $z + $c;  
        return 1 if ($z.abs > 2);  
    }  
    return 0;  
}
```

0..\$max_iterations range

```
for subdivide($upper-right.re,  
    $lower-left.re, $height) -> $re {  
    . . .  
    (@line, $middle,  
        @line.reverse).join(' ').say;  
}
```

```
for subdivide($upper-right.re,  
$lower-left.re, $height) -> $re {  
  . . .  
  (@line, $middle,  
    @line.reverse).join(' ').say;  
}
```

Hyphens in variable names

```
for subdivide($upper-right.re,  
  $lower-left.re, $height) -> $re {  
  . . .  
  (@line, $middle,  
    @line.reverse).join(' ').say;  
}
```

for loop and its variable

```
for subdivide($upper-right.re,  
  $lower-left.re, $height) -> $re {  
  . . .  
  (@line, $middle,  
    @line.reverse).join(' ').say;  
}
```

Nested method calls


```
(@line, $middle, @line.reverse).map  
( { @color_map[$_] } ).join(' ').say;
```

Method calls on a list

53

Perl 6 projects

still left

FakeDBI and FakeDBD

```
class FakeDBI:auth<mberends>:ver<0.0.1> {  
  has $!err;  
  has $!errstr;  
  method connect(  
    $dsn, $username, $password,  
    :$RaiseError=0, :$PrintError=0,  
    :$AutoCommit=1 ) {
```

Declaring and defining a class

```
class FakeDBI:auth<mberends>:ver<0.0.1> {  
  has $!err;  
  has $!errstr;  
  method connect(  
    $dsn, $username, $password,  
    :$RaiseError=0, :$PrintError=0,  
    :$AutoCommit=1 ) {
```

Who is the author

```
class FakeDBI:auth<mberends>:ver<0.0.1> {  
  has $!err;  
  has $!errstr;  
  method connect(  
    $dsn, $username, $password,  
    :$RaiseError=0, :$PrintError=0,  
    :$AutoCommit=1 ) {
```

Version number

```
class FakeDBI:auth<mberends>:ver<0.0.1> {  
  has $!err;  
  has $!errstr;  
  method connect(  
    $dsn, $username, $password,  
    :$RaiseError=0, :$PrintError=0,  
    :$AutoCommit=1 ) {
```

Class variables

```
class FakeDBI:auth<mberends>:ver<0.0.1> {  
  has $!err;  
  has $!errstr;  
  method connect(  
    $dsn, $username, $password,  
    :$RaiseError=0, :$PrintError=0,  
    :$AutoCommit=1 ) {
```

Not too easy to guess


```
bash-3.2$ grep '$!' * -r
```

S02-bits.pod:

\$!foo object attribute

private storage

```
class FakeDBI:auth<mberends>:ver<0.0.1> {  
  has $!err;  
  has $!errstr;  
  method connect(  
    $dsn, $username, $password,  
    :$RaiseError=0, :$PrintError=0,  
    :$AutoCommit=1 ) {
```

Twigils indicate private variables

```
class FakeDBI:auth<mberends>:ver<0.0.1> {  
  has $!err;  
  has $!errstr;  
  method connect(  
    $dsn, $username, $password,  
    :$RaiseError=0, :$PrintError=0,  
    :$AutoCommit=1 ) {
```

Class method

```
class FakeDBI:auth<mberends>:ver<0.0.1> {  
  has $!err;  
  has $!errstr;  
  method connect(  
    $dsn, $username, $password,  
    :$RaiseError=0, :$PrintError=0,  
    :$AutoCommit=1 ) {
```

Positional arguments

```
class FakeDBI:auth<mberends>:ver<0.0.1> {  
  has $!err;  
  has $!errstr;  
  method connect(  
    $dsn, $username, $password,  
    :$RaiseError=0, :$PrintError=0,  
    :$AutoCommit=1 ) {
```

Named arguments

```
class FakeDBI:auth<mberends>:ver<0.0.1> {  
  has $!err;  
  has $!errstr;  
  method connect(  
    $dsn, $username, $password,  
    :$RaiseError=0, :$PrintError=0,  
    :$AutoCommit=1 ) {
```

Default values

```
given $drivername {  
    when 'CSV'      { . . . }  
    when 'mysql'    { . . . }  
    default         { . . . }  
}
```

given/when known from Perl 5.10 :-)

52

Perl 6 projects

still left

Really 52 left?

Much more!

perl6-examples on github

cookbook

doc

euler

games

interpreters

lib

module-management

parsers

perlmonks

shootout

tutorial

wsg

perl6-examples on github

cookbook

doc

euler

games

interpreters

lib

module-management

parsers

perlmonks

shootout

tutorial

wsg

eulerproject.net

README

prob001-cspencer.pl	prob005-unobe.pl	prob025-polettix.pl
prob001-eric256.pl	prob006-polettix.pl	prob029-polettix.pl
prob001-hexmode.pl	prob007-polettix.pl	prob052-duff.pl
prob001-unobe.pl	prob008-duff.pl	prob053-duff.pl
prob002-eric256.pl	prob008-duff2.pl	prob063-moritz.pl
prob002-hexmode.pl	prob009-polettix.pl	prob063-polettix.pl
prob003-eric256.pl	prob010-polettix.pl	prob081-matrix.txt
prob003-hexmode.pl	prob011-moritz.pl	prob081-moritz.pl
prob003-lanny.p6	prob012-polettix.pl	prob092-moritz.pl
prob004-unobe.pl	prob017-duff.pl	prob104-moritz.pl
	prob024-moritz.pl	

eulerproject.net

README

prob001-cspencer.pl

prob001-eric256.pl

prob001-hexmode.pl

prob001-unobe.pl

prob002-eric256.pl

prob002-hexmode.pl

prob003-eric256.pl

prob003-hexmode.pl

prob003-lanny.p6

prob004-unobe.pl

prob005-unobe.pl

prob006-polettix.pl

prob007-polettix.pl

prob008-duff.pl

prob008-duff2.pl

prob009-polettix.pl

prob010-polettix.pl

prob011-moritz.pl

prob012-polettix.pl

prob017-duff.pl

prob024-moritz.pl

prob025-polettix.pl

prob029-polettix.pl

prob052-duff.pl

prob053-duff.pl

prob063-moritz.pl

prob063-polettix.pl

prob081-matrix.txt

prob081-moritz.pl

prob092-moritz.pl

prob104-moritz.pl

```
# A simple implementation  
# of Eratosthenes' sieve  
sub primes_iterator {  
    return sub {  
        state %D;  
        state $q //= 2;  
        $q //= 2;
```

Again, pure Perl 5.10 :-)

```
$ perl6 prob007-polettix.pl
```

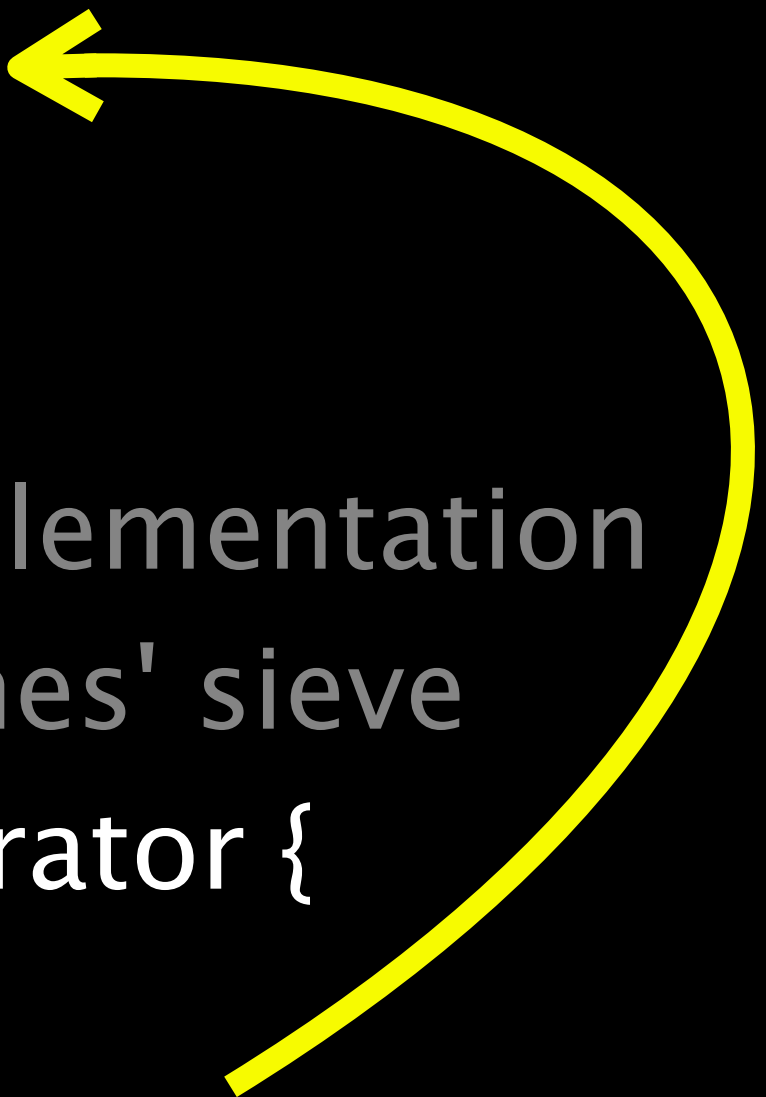
```
===SORRY!===
```

```
Symbol '%D' not predeclared in primes_iterator  
(prob007-polettix.pl:17)
```



```
my %D;  
my $q;
```

```
# A simple implementation  
# of Eratosthenes' sieve  
sub primes_iterator {  
    return sub {  
        #state %D;  
        #state $q //= 2;  
        $q //= 2;
```



OK, let's use global variables this time

```
$ perl6 prob007-polettix.pl
```

```
... Time passed. . .
```

```
result: 104743
```

```
my $it = primes_iterator();  
for 1 .. $nth - 1 -> $i {  
    $it();  
    say "found $i primes so far" unless $i % 100;  
}  
say 'result: ', $it();
```

Subroutine reference in a scalar

51 + ∞

Perl 6 projects

still left

SVG.pm

```
my $svg = :svg[
  :width(200), :height(200),
  circle => [
    :cx(100), :cy(100), :r(50)
  ],
  text => [
    :x(10), :y(20), "hello"
  ]
];
```

Hash reference of hash references?

`.perl()` explains

```
say $svg.perl;
```

```
"svg" => ["width" => 200,  
"height" => 200, "circle" =>  
["cx" => 100, "cy" => 100, "r"  
=> 50], "text" => ["x" => 10,  
"y" => 20, "hello"]]
```


50 + ∞

Perl 6 projects

still left

Andrew Shitov

talks.shitov.ru | andy@shitov.ru

