Perl update 2017

Perl 5 & 6 update

Das Beste aus Perl 6

Defaults

Anfängerhilfen

Operatoren

Parallel & Async

Rollen

Regex

Grammatiken

Perl 5 & 6 update

Perl 5.22 .. 5.22.2

5.24.1 neu

6.c praktisch

Lichtkind:

/Artikel /Blog

/Code

/Doc

/Musik

/Vortrag

Perl 5 installieren

App::cpanminus

App::perlbrew

Perl 5 installieren

perlbrew init

perlbrew available

..brew install perl-5.24.1

..ew exec --with perl-5.12

Perl 5 installieren

perlbrew init

perlbrew available

..brew install perl-5.24.1

..ew exec --with perl-5.12

perlbrew available

```
  perl-5.25.10
i perl-5.24.1
  perl-5.22.3
  perl-5.20.3

  ..
  perl-5.6.2
  perl5.005_04
  perl5.004_05
```

perlbrew available

github.com/perl11/cperl/releases

Perl 5 installieren

perlbrew init

perlbrew available

..brew install perl-5.24.1

..ew exec --with perl-5.12

Perl 5 installieren

perlbrew init

perlbrew available

..brew install perl-5.24.1

..ew exec --with perl-5.12

Perl 5 installieren

perlbrew init

perlbrew available

..brew install perl-5.24.1

..ew exec --with perl-5.12

Perl 5 update

```
use v5.22;
use feature '...';

use experimental '...';
```

Perl 5.22

```
&. |. ^. ~.    <<>>
   qr/\b{...}/      //n
use re 'strict';
      \$c = \$d;
```

Perl 5.22

Unicode 7

Perl 5.24

Unicode 8

Perl 5.26

Unicode 9

Perl 5

\b

word boundary

Perl 5.22 (UTF 7)

\b{wb}

word boundary

Perl 5.22 (UTF 7)

\b{sb}

sentence boundary

Perl 5.22 (UTF 7)

\b{gcb}

grapheme cluster boundary

Perl 5.24 (UTF 8)

\b{lb}

line boundary

Perl 5.22

```
use re 'strict';
```

```
use re 'strict';

qr/\xABC/
```

Perl 5.22

/.../n

~~$& $` $' $1 $2 $+~~

Perl 5.22

/ \{ /

Perl 5.22

```
use feature 'bitwise';

&. |. ^. ~.
```

zeichenweise

&. AND

|. OR

^. XOR

~. NOT

Perl 5	Perl 6
&.	~&
|.	~|
^.	~^
~.	~!

Perl 5.22 (6)

```
\$c = \$d;
$c := $d;
```

Perl 5.22

<<>>

Perl 5.22

<<>> open .,,.,.,.;

<> open .. , .. ;

say <<>>

```
perl t.pl "<t.pl"
5.024001
Can't open <t.pl:...
```

say <>

```
perl t.pl "<t.pl"
5.018002
..........
```

Perl 5.24

```
->@*    <<  >>
   qr/\b{lb}/
#!bin/perl6
   chdir('');
```

schneller

/ \{\} /

+ - * \

subname();

seit Perl 5.20

use experimental
/postderef
postderef_qq/;

seit Perl 5.24

```
use feature
  /postderef
  postderef_qq/;
```

adaptiert Perl 6

```
use feature
  /postderef
  postderef_qq/;
```

Skalarreferenz

```
$sref->$*;

${ $sref };
```

Arrayreferenz

```
$aref->@*;

@{ $aref };
```

Arrayreferenz

```
$aref->$#*;

$#{ $aref };
```

Arrayreferenz

$aref->@[...];

@$aref[...];

Hashreferenz

```
$href->@{ ... };
@$href{ ... };
```

Hashreferenz

```
$href->%*;

%{ $href };
```

Arrayreferenz

$aref->%[...];

%$aref[...];

Hashreferenz

```
$href->%{ ... };

%$href{ ... };
```

Kodereferenz

```
$href->&*;

&{ $href };
```

Globferenz

```
$gref->**;

*{ $gref };
```

Globferenz

```
$gref->*{ ... };

*{ $gref }{ ... };
```

seit Perl 5.24

```
use feature
  /postderef
  postderef_qq/;
```

Autodereferenzieren

~~**use** experimental 'autoderef';~~

push $ref, ...;

lexikalischer Kontext

~~**use** experimental 'lexical_topic';~~

my $_;

Nein

```
my ($val, our($dir));
our (my $x);
```

Nein

```
my $;
```

private compile time hints

```
my $^H;
```

Nein

```
chdir('');
chdir( undef );
```

Ja

~~chdir('');~~

~~chdir(undef);~~

chdir( );

Nein

Nein

~~ACI~~

uf8::encode

Nein unter UTF

```
syswrite($FH,$val);
```

UTF Zeilenumbruch

/b{lb}/
zugefügt

UTF Wortumbruch

/\b{wb}/

geändert

Zahlen

/ \d{} /

deprecated

match 4{}

/ \d\{\} /

match {}

/ \{\} /

muß in 5.26

/ \{\} /

Perl 5.26

<<~

//xx

Perl 5.26

ref Zuweisungen
split
schneller

Perl 5.26

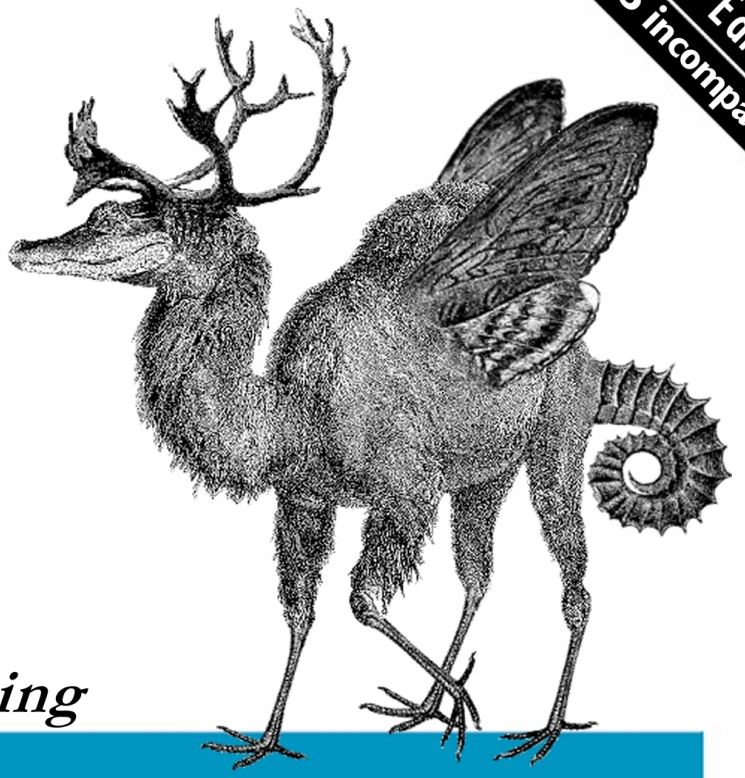
```
say <<~EOF;
Moin
EOF
```

Perl 5.26

/ [0-9 a-f] /xx

There's More Than One Way To Screw It Up

33rd Edition
Perl 5 incompatible



Designing

Perl 6

NOT´REALLY®

A committee - and look at the mess we ended up with

Perl 6 update

bessere Seite
mehr docs,
whateverable

Perl 6

committable6: 2016.05 say ‘hello’

<committable6> ¦2016.05: «hello»

Perl 6

unicodable6: ☺

<unicodable6> naxieAlDle, U+263A WHITE SMILING FACE [So] (☺)

Rakudo

schneller (~2x),
durchdachter,
bessere Fehler

Mehr Module

seit 6.c:

475 → 790

Profiler

perl6 --profile

Perl 6 installieren

perl6.org

rakudo.org

P6 von  installieren

git pull

perl Configure.pl --gen-moar --gen-nqp --backends=moar

make

make install

PATH gesetzt

perl6 hello.pl

auch von Hand

rakudobrew

Modul Installer Zef

rakudobrew
build zef
zef list

Das Übliche | Beste

zef install Task::Star

Sorglospaket

rakudo*

File IO

```
say
prompt
$in = slurp“datei.txt”;
spurt “datei.txt”,$out;
lines
words
```

Channel

```
my $channel = Channel.new;

$channel.send($msg);
my $msg = $channel.recieve;
my $msg = $channel.poll; # !block
$channel.close;
```

Channel

```
react {
    whenever $channel {
        "got a $_".say;
    }
}
```

Atrribute

```
class{
    has $.strength;
    method man(...){...}
}
role{
    has $.strength;
    method man(...){...}
}
```

Unicode Operatoren

« » × ÷ ≤ ≥ ≠ − ∘ ≅

?? !! π τ ∞ … ‘ ’ 「 」 + −

1 2 3 4 5 6 7 8 9 。

∅ ∈ ∉ ∋ ∌

⊆ ⊈ ⊂ ⊄ ⊇ ⊉ ⊃ ⊅

⩽ ⩾ ∪ ∩ ∖ ⊖ ⊍ ⊎

Arraymethode

elems end keys values kv pairs antipairs join map flat flatmap grep first head tail categorize classify Bool Str Int Numeric Capture pick roll eager reverse rotate sort unique repeated squish reduce produce combinations permutations rotor cross zip sub roundrobin sum fmt from to

Binomialkoeffizient

```
use v6;

sub infix:<choose> {
    [*] ($^n ... 0) Z/ 1 .. $^p
}

say 5 choose 3;
```

Binomialkoeffizient

```
use v6;

sub infix:<choose> {
    [*] ($^n ... 0) Z/ 1 .. $^p
}

say 5 choose 3; # 5! / 3!*2!  = 5 * 4 / 2! = 10
```

Infix operator

```
use v6;

sub infix:<choose> {
    [*] ($^n ... 0) Z/ 1 .. $^p
}

say 5 choose 3; # 5 & 3 sind Parameter
```

Infix operator

```
use v6;

sub infix:<choose> {
    [*] ($^n ... 0) Z/ 1 .. $^p
}

say 5 choose 3; # 5 & 3 sind Parameter
# keine Signatur → positionale Par. (@_)
```

Autobenannte Par.

```
use v6;

sub infix:<choose> {
    [*] ($^n ... 0) Z/ 1 .. $^p
}

say 5 choose 3; # 5 & 3 sind pos. Par.
# und landen in autobenannt.:  $^n  $^p
```

Autobenannte Par.

```
use v6;

sub infix:<choose> {
    [*] ($^n ... 0) Z/ 1 .. $^p
}

say 5 choose 3; # 5 & 3 pos. auto. Par.
# a .. n .. p .. z  → $^n = 5  ;  $^p = 3
```

Param einsetzen

```
use v6;

sub infix:<choose> {
    [*] (5 ... 0) Z/ 1 .. 3
}

say 5 choose 3;
```

range auflösen

```
use v6;

sub infix:<choose> {
    [*] (5 ... 0) Z/ 1, 2, 3
}

say 5 choose 3;
```

sequence auflösen

```
use v6;

sub infix:<choose> {
    [*] (5, 4, 3, 2, 1, 0) Z/ (1, 2, 3)
}

say 5 choose 3;  # Präzedenz beachten
# Z/ eine Zeile über  [*]
```

zip Metaopetrator Z

```
use v6;

sub infix:<choose> {
    [*] (5, 4, 3, 2, 1, 0) Z/ (1, 2, 3)
}

say 5 choose 3;
# erzeugt Tupel aus links & rechts El.
```

Z fordert =lang. Listen

```
use v6;

sub infix:<choose> {
    [*] (5, 4, 3) Z/ (1, 2, 3)
}

say 5 choose 3;
# kürze linke Liste
```

Z aufgelöst

```
use v6;

sub infix:<choose> {
    [*] (5 / 1), (4 / 2), (3 / 3)
}

say 5 choose 3;
# / ist Division
```

/ aufgelöst

```
use v6;

sub infix:<choose> {
    [*] (5, 2, 1)
}

say 5 choose 3;
# [ ] ist reduce meta op, * ist Multipl.
```

[ ] aufgelöst

```
use v6;

sub infix:<choose> {
    (5 * 2 * 1)
}

say 5 choose 3;
# [ ] ist reduce meta op, * ist Multipl.
```

* aufgelöst

```
use v6;

sub infix:<choose> {
    10
}

say 5 choose 3;
# Hurra wir haben unser Ergebnis
```

Euler #2

```
[+] grep * %% 2,
          (1, 1, *+* ...^ * > 4_000_000);

# Summe der geraden Fibonacci Zahlen
  bis 4 000 000
```

Summe

```
[+] grep * %% 2,
          (1, 1, *+* ...^ * > 4_000_000);

# Summe der geraden Fibonacci Zahlen
  bis 4 000 000
```

ausgewählter Zahlen

```
[+] grep * %% 2,
          (1, 1, *+* ...^ * > 4_000_000);

# Summe der geraden Fibonacci Zahlen
  bis 4 000 000
```

durch 2 teilbar

```
[+] grep * %% 2,
          (1, 1, *+* ...^ * > 4_000_000);

# Summe der geraden Fibonacci Zahlen
  bis 4 000 000
```

Fibonacci Reihe

```
[+] grep * %% 2,
          (1, 1, *+* ...^ * > 4_000_000);

# Summe der geraden Fibonacci Zahlen
  bis 4 000 000
```

bis obere Schranke

```
[+] grep * %% 2,
          (1, 1, *+* ...^ * > 4_000_000);

# Summe der geraden Fibonacci Zahlen
  bis 4 000 000
```

Twitter::API::Search

```
use Subset::Helper;
my subset TwitterGeoCode of Str
                                    where subset-is {
        m/^
            <[-+]>? \d+ [\. \d+]? ','
            <[-+]>? \d+ [\. \d+]? ','
                        \d+ [\. \d+]? [ 'mi' | 'km' ]
          $/
    }, 'helpsting...latitude,longitude,radius';
```

Grosser Parser

```
unit grammar XML::Grammar;

rule TOP {
  ^
  <xmldecl>?          [ <comment> | <pi> ]*
  <doctypedecl>?      [ <comment> | <pi> ]*
  <root=element>      [ <comment> | <pi> ]*
  $
}
```

in 100 Zeilen

```
token element {
    '<' \s* <name> \s* <attribute>*
    [
    | '/>'
    | '>' <child>* '/>' $<name> '>'
    ]
}
```

Unterelement

```
token child {
   | <element>
   | <cdata>
   | <text=textnode>
   | <comment>
   | <pi>
}
```

in Document.pm :

```
multi method new (Str $xml, :$filename){
    ...
    my $doc = XML::Grammar.parse($xml);
    ...
    $version = ~$doc<xmldecl><version><value>;
```

anderswo

```
use MONKEY-TYPING;

augment slang MAIN {
    ...
}
```

Ganz woanders

```
sub circumfix:<« »> (Str $xml) {
    XML::Grammar.parse($xml)
}
```

Ganz woanders

```
sub circumfix:<« »> (Str $xml) {
        XML::Grammar.parse($xml)
}

my $dom = « <?xml version=… »
```

Math::Matrix

```
multi sub circumfix:<|| ||>
(Math::Matrix $a --> Numeric) is
equiv(&prefix:<!>) is export {
        $a.norm();
}

my $norm = ||$matrix||;
```

C Kode

```
#!/usr/bin/env perl6
use soft;     # for now
use Inline;

my sub a_plus_b( Int $a, Int $b ) is
inline('C') returns Int {'
    DLLEXPORT int a_plus_b (int a, int b) {
        return a + b;
    }
'}
```

C Kode

```
use NativeCall;
our sub init() is native('foo')
              is symbol('FOO_INIT') { * }
```

Perl 6.c

perl6.org

rosettacode.org/wiki/category:Perl_6

Jonathan Worthington