

Perl

'the Swiss Army chainsaw of scripting languages'

Denis Moskal

UnFUG

November 10, 2011

Inhaltsverzeichnis

1 Was ist Perl?

Inhaltsverzeichnis

1 Was ist Perl?

2 Programmbeispiele

Inhaltsverzeichnis

- 1 Was ist Perl?
- 2 Programmbeispiele
- 3 RegExMagic

Inhaltsverzeichnis

- 1 Was ist Perl?
- 2 Programmbeispiele
- 3 RegExMagic
- 4 Quellen

Einzelheiten

- ▶ 1987
- ▶ Larry Wall
- ▶ Verarbeitung von Textdateien

Larry Wall



Die wichtigsten Fakten

- ▶ Entwickler: Larry Wall
- ▶ Aktuelle Version: 5.14.2
- ▶ Lizenz: GPL und/oder Artistic License
- ▶ Sonstiges: großes Software- und Modularchiv CPAN

Merkmale

- ▶ freie, plattformunabhängige und interpretierte Programmiersprache
- ▶ "There is more than one way to do it" TIMTOWTDI("Tim Toady")
- ▶ "Perl makes easy jobs easy and hard jobs possible"

Vorteile

- ▶ Programmierer hat viele Freiheiten
- ▶ Viele Module/Erweiterungen vorhanden
- ▶ mächtige RegEx-Syntax

Vorteile

- ▶ Programmierer hat viele Freiheiten
- ▶ Viele Module/Erweiterungen vorhanden
- ▶ mächtige RegEx-Syntax

Nachteile

- ▶ manchmal mangelnde Lesbarkeit des Codes
- ▶ Performanceverlust bei zu vielen importierten Modulen

Einsatzgebiete

- ▶ Webserver
- ▶ 'data munging'

```
#!/usr/bin/perl

use warnings;
use v5.10;

my $s = "Hello World";

say $s;
```

```
#!/usr/bin/perl  
..  
#Ganzzahlen  
my $a = 4*2;  
say $a;
```

```
#!/usr/bin/perl
..
#Ganzzahlen
my $a = 4*2;
say $a;
#Gleitkommazahlen
my $b = 4.2*4.2;
say $b;
```

```
#!/usr/bin/perl
..
#Ganzzahlen
my $a = 4*2;
say $a;
#Gleitkommazahlen
my $b = 4.2*4.2;
say $b;
#Andere Zahlensysteme
say 0xff;
say 0377;
say 0b11111111;
```

```
#!/usr/bin/perl  
..  
#Zeichenketten  
say '\t42';  
say "\t42";
```

```
#!/usr/bin/perl  
..  
#Zeichenketten  
say '\t42';  
say "\t42";  
say qq<"42">;
```

```
#!/usr/bin/perl
..
#Zeichenketten
say '\t42';
say "\t42";
say qq<"42">;
say "Print " . "one " . "string " . "here";
say "Print ", "several ", "strings ", "here";
```

```
#!/usr/bin/perl
..
#Listen
say "Counting up: ", (0..1,2,3);
say "Counting up: ", (0 .. 3);
```

```
#!/usr/bin/perl
..
#Listen
say "Counting up: ", (0..3);
say "Counting up: ", (0 .. 3);
#Arrays
my @array = (0 .. 3);
my $scalar = @array;
say "Counting up: ", @array;
say '@array hat eine Größe von ', $scalar;
```

```
#!/usr/bin/perl  
..  
#Hash  
my %hash = (  
Test => "erfolgreich"  
);  
say $hash{Test};
```

```
#!/usr/bin/perl
..
# if-Abfrage
if (..) {...}
# while-Schleife
while(..) {...}
# foreach-Schleife
for $scalar(@array) {say $scalar;}
# unless-Abfrage;
say "File not found" unless(defined $file);
```

```
#!/usr/bin/perl  
..  
my $lineno = 1;  
#STDIN  
my $input = <STDIN>; # $input = <>
```

```
#!/usr/bin/perl
..
my $lineno = 1;
#STDIN
my $input = <STDIN>; # $input = <>
# Filehandles
open FILE, "test.txt" or die $!;
while (<FILE>) {
print $lineno++;
print ": $_";
}
```

```
#!/usr/bin/perl
..
my $lineno = 1;
#STDIN
my $input = <STDIN>; # $input = <>
# Filehandles
open FILE, "test.txt" or die $!;
while (<FILE>) {
print $lineno++;
print ": $_";
}
#print FILE list;
```

```
#!/usr/bin/perl
..
#references
my @array = (1..3);
my $arr_ref = array;
```

```
#!/usr/bin/perl
..
#references
my @array = (1..3);
my $arr_ref = array;
#deref
my @arr1 = @{$arr_ref};
my @arr2 = @$arr_ref;
```

```
#!/usr/bin/perl
..
#references
my @array = (1..3);
my $arr_ref = array;
#deref
my @arr1 = @{$arr_ref};
my @arr2 = @$arr_ref;
my $value = ${$arr_ref}[1];
my $value2 = $arr_ref->[1];
```

```
#!/usr/bin/perl  
..  
my ($a,@array) = total(1..100);  
say @array;
```

```
#!/usr/bin/perl
..
my ($a,@array) = total(1..100);
say @array;
sub total { #total($$$)
my $total;
$total += $_ for @_;
say "The total is $total";
```

```
#!/usr/bin/perl
..
my ($a,@array) = total(1..100);
say @array;
sub total { #total($$$)
my $total;
$total += $_ for @_;
say "The total is $total";
return ($total,@_);
}
```

```
#!/usr/bin/perl  
use Person;  
..
```

```
#!/usr/bin/perl  
use Person;  
..  
my $person = Person->new();  
$person->method();  
..
```

```
package Person;
```

```
..
```

```
package Person;  
  
..  
sub new {  
my $class = shift;  
my $self = {..};  
bless $self, $class;  
return $self;  
}
```

kurzer überblick

- ▶ Regular Expression
- ▶ Suchmuster
- ▶ `$test =~ /pattern/`

[..]

- ▶ [ab] - matches a or b
- ▶ [^ab] - other than a or b
- ▶ [a-z] - matches character between a and z

[..]

- ▶ [ab] - matches a or b
- ▶ [^ab] - other than a or b
- ▶ [a-z] - matches character between a and z

\..

- ▶ /^../ - beginning
- ▶ /..\$/ - end
- ▶ \d - a digit
- ▶ \w - a 'word' character
- ▶ \s - whitespace
- ▶ \b - boundary character

- ▶ '.' - any character (not \n)
- ▶ (abc) - abc as group
- ▶ '?' - 0 or 1
- ▶ '+' - 1+
- ▶ '*' - 0+

- ▶ `'.'` - any character (not `\n`)
- ▶ `(abc)` - abc as group
- ▶ `'?'` - 0 or 1
- ▶ `'+'` - 1+
- ▶ `'*'` - 0+

`{..}`

- ▶ `{x,y}` - between x and y times
- ▶ `{,y}` - at most y times
- ▶ `{x,}` - at least x times
- ▶ `{x}` - x times

Beispiele

- ▶ `[a-z]+`
- ▶ `\w+`
- ▶ `[a-z]+(.*)[a-z]+`
- ▶ `e(\w|n \w+)`

- ▶ www.perl.org
- ▶ perldoc.perl.org