Perl 6

„Perl 5 was my rewrite of Perl.
I want Perl 6 to be the community's rewrite of Perl... and of the community.“
– Larry Wall, State of the Onion speech, OSCon 2000

Hagen Paul Pfeifer

https://www.jauu.net ● hagen@jauu.net

14. April 2005

Überblick

- Grundsätzliches
- Syntax
- Quellen

Grundsätzliches

- Perl 5: „`easy things easy, and hard things possible`“
- Perl 6: „`easy things trivial, hard things easy, and impossible things merely hard`“ –Damian Conway ;-)
- „Community Rewrite of Perl“
- Parrot
- Prozess:
  - Apocalypses (Larry Wall) (nicht leicht zu lesen!)
  - Exegeses (Damian Conway) (exemplified)
  - Festgelegt durch Synopsis (Entwicklerteam)
- Nicht kompatibel zu Perl 5

Perl 5 – Schwächen

- Keine Typbindung
- Kein case Kontollflußstatement
- Objekt Orientiertes Modell ist Schwach (Exceptions, ...)
- Benannte Argumente (@_)
- Sigils ($, @, %)

Parrot

- Virtuelle Mashine
- Sprachneutrale Laufzeitumgebung (Phyton, Ruby, Scheme, ...)
- Threading und Unicode Support
- Exception Support
- Erzeugt Bytecode und Binarys

Typbindung

- Problem Perl5: welchen Typ hat die Variable; wie soll ich sie behandeln; kann ich mir sicher sein?
- Typ Deklaration bei `my` und `our` möglich

```
my Int $number;
my Widget $obj_ref;
my Str @strings;
my Array of Num %counters;
```

- Wie immer: Perl6 ist mächtiger:

```
my Num @observations is SparseArray # storage type && implementation
```

- Mal anders:

```
sub hist(Int @vals) returns Array of Int {
  my Int @histogram;
  for @vals { @histogram[$_]++ }
  return @histogram;
}
```

Sigils

- Perl5: `keys %hash; $hash{'key'}; ...`
- Komplizierter bei zunehmender Referenzierung
- Krass bei Referenzen auf Funktionen, etc.
- Perl6: `keys %hash; %hash{"name"};`
- Ergebniss: weniger Fehleranfällig und besser für /dev/brain

Switch Statement

- Ansatz:

```
$val = 'G4';
given $val {
  when 'A4' { print "paper" }
  when 'B4' { print "prior" }
  when 'C4' {   die "BOOM!" }
  default   { print "huh??" }
}
```

- ...es wird Spannender!

```
$val = 'G4';
given $val {
  when 'A4' { print "paper" } # String compare
```

```
    when %B4  { print "prior" } # true if %B4{$val} exists
    when /C4/ {   die "BOOM!" } # regex, klaro ;-)
    when &D4  { print "huh??" } # jetzt auch klar
}
```

- KAUM ZU GLAUBEN:

```
$guess = given $first_char {
  when [0..9]    { 'dec' }
  when /<[A-F]>/ { 'hex' }
  when &is_ASCII { '7-bit' }
  when %known    { %known{$_} }
  default        { die Cannot::Guess }
}
```

- when ist aber mächtiger

```
for (@events) {
  when Mouse::Over       { change_focus($_) }
```

```
        when Mouse::Click       { make_selection() }
        when Window::Enter      { change_focus($_) }
        when Window::Close      { delete_window()  }
        when /unknown\s+event/  { log_event($_)    }
    }
```

Objekte

- Perl5 Objekte: Hashreferezen welche via `bless()` an einem Namensraum gebunden worden ist
-
- Alles sind Objekte in Perl (z.B.: `@aaa.length`)
- `$var = 0;` nun auch wahr!
- Überladen von Methoden möglich
- `@all = %hash.kv;`
- Perl6 Klassendeklaration:

```
class Staff::Record { # Klassen Deklaration (class)
  has Str  $.name;    # Attribute Deklaration (has)
  has Int  $.rank;
  has Hash $.pref;
```

```
        method CREATE(Str $name, Int $rank, Hash $cereal_pref) {
            .check_rank($rank);
            ($.name, $.rank, $.pref) = ($name, $rank, $cereal_pref);
        }

        method name() returns Str {
          return $.name;
        }

        method destroy() is private {
          die "Failure in class Staff::Record";
        }
      }
```

- $_ = $object; .printContent()
-

Vergleiche

- Fehleranfällig in Perl 5 (war es ein String oder eine Zahl?)
- `eq, ==`
- Perl6: `$x ~~ 23` oder `$var ~~ /perl/`
- Noch besser: `if (@a ~~ 23)` Ist eine Zahl im Array
- Listen Identisch? `@a ~~ @b`!

Vectorised operators

- Jedes Element einer Liste bearbeiten?!
- `@values ^+= 1;`

Funktionen

- Aehnlich zu Perl5
- `sub foo { print @_ }`
- `sub foo (*@_) { print @_ }`
- Besser: `sub swap (*@_ :rw) { @_[0,1] = @_[1,0] };`
- Auch möglich: `sub add { return $^first + $^second;}`

Kontrollfluss

- `for @data -> $var { print $var}`
- oder: `for (1 .. 10) -> $var { print $^var}`
- `for %demohash { print .key}`
- ach ja Stringkonkatenation jetzt _ nicht mehr `.` (Java!)

RegEX

- `$a =~ /end/i` Perl6: `$x ~~ m:i/end/`
- `$a =~ s/(\w+)/ucfirst(\1)/eg` wird: `$a ~~ s:g/(\w+)/$(ucfirst($1))/`
- Modifiers:
  - :2x _zwei_ mal austauschen
  - :2rd zweiten match austauschen
  - :e2rd jeden zweiten match austauschen
- Speicherlose Klammer nun: `[]`

Was noch?

- Introspection
- Starke Kapselung bei Klassen
- Funktionsüberlagerung
- hirachisches Destrukter und Konstrukter
- Byte Code und Binarys durch Parrot
- . . . . . .

Abschliessendes

- Ich bin heiss und möchte mein WE sinnvoll gestalten!
  Geduld: wie es z.Z. aussieht, wird es einen ersten Compiler gegen Ende des Jahres geben.