

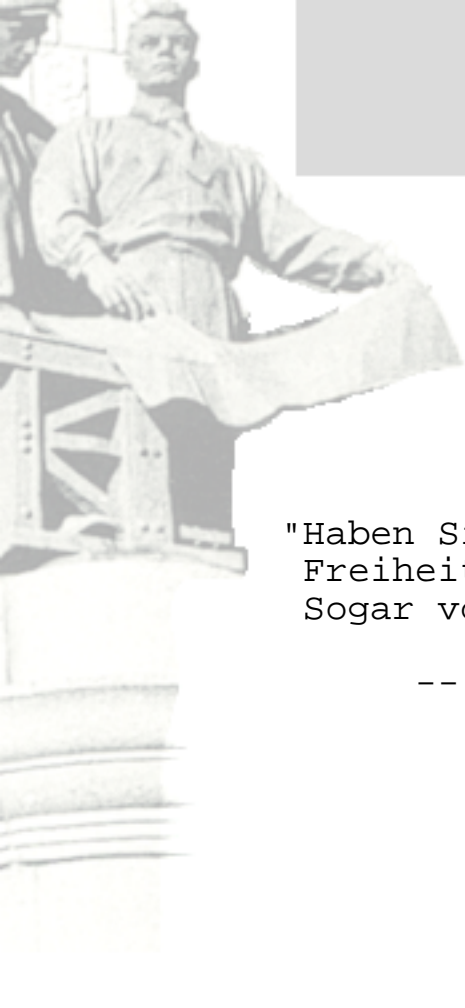
A faded, grayscale image of a statue of a man in a suit, likely a historical figure, is positioned on the left side of the slide. The statue is seated and appears to be holding a document or a book. The background of the slide is white with a gray header and footer bar.

Linux Info Tag

Perl - Eine Free-Software-Kultur

-- Codeästhetik und Perl Fun --

Steffen Schwigon <schwigon@webit.de>
Dresden Perl Mongers

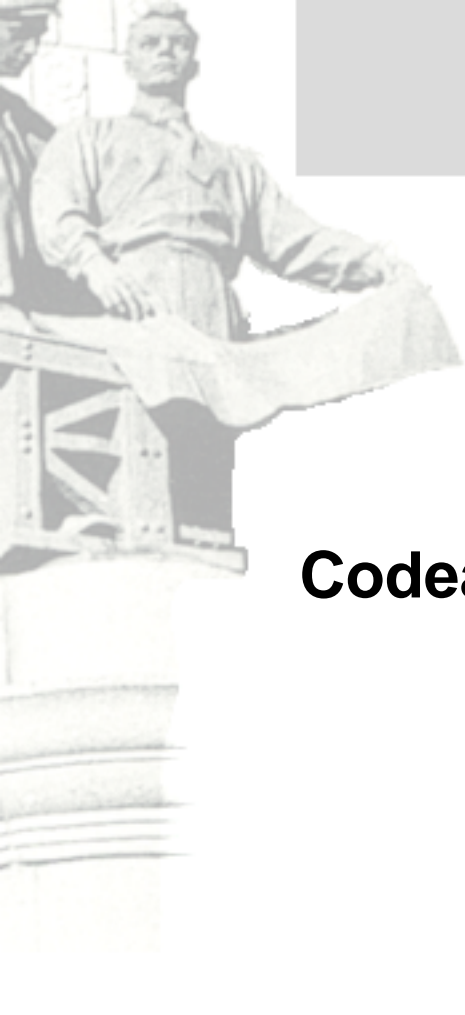


"Haben Sie jemals über wirkliche Freiheiten nachgedacht?
Freiheiten von den Meinungen anderer?
Sogar von den EIGENEN Meinungen?"

-- Apocalypse Now

Übersicht

- Ästhetik durch Syntaxvielfalt
- Perl-Fun (Acme-Module)
- Sinnvolle Anwendungen



Codeästhetik durch Syntaxvielfalt

Ästhetik und Perl-Mythen

- Codeästhetik
 - ◆ Wenig überflüssiges Zeichenrauschen
 - ◆ minimalistisch aber konsistent (Lisp)
 - ◆ Farben
 - minimalistisch
 - mehr bold/non-bold
- Mythos: Perl schwer lesbar
 - ◆ zu kurzer Code
 - ◆ zu viel implizite Tricks
- Vielfalt ermöglicht Mißbrauch; kein prinzipieller Nachteil
 - ◆ kurzer Code kann Lesbarkeit fördern
 - ◆ Integration von Programmier-Patterns in Sprache
 - ◆ Frage des persönlichen Kung Foo

Ästhetik und Perl-Mythen

- Codeästhetik
 - ◆ Wenig überflüssiges Zeichenrauschen
 - ◆ minimalistisch aber konsistent (Lisp)
 - ◆ Farben
 - minimalistisch
 - mehr bold/non-bold
- Mythos: Perl schwer lesbar
 - ◆ zu kurzer Code
 - ◆ zu viel implizite Tricks
- Vielfalt ermöglicht Mißbrauch; kein prinzipieller Nachteil
 - ◆ kurzer Code kann Lesbarkeit fördern
 - ◆ Integration von Programmier-Patterns in Sprache
 - ◆ Frage des persönlichen Kung Foo
 - ◆ Mißbrauch kann Spaß machen

Gesucht: Targets eines Makefiles

```
THEMA = acme
IMAGES = *.png
ALLSLIDES = $(THEMA).pdf

.SUFFIXES: .pdf .axp

%.pdf: %.axp
  axpoint $< > $@

all: $(ALLSLIDES)

$(THEMA).pdf: $(THEMA).axp $(IMAGES)

clean:
  -rm -f $(ALLSLIDES)
dist:
  @echo "You don't need a dist."
```

```
$ ./nostyle.pl
all
clean
dist
```

Nackter häßlicher Code

```
#!/usr/bin/perl

sub readfile {
    my $ok = open (FH, "<Makefile");
    if (defined $ok) {
        while ($line = <FH>) {
            if ($line !~ /^\\w*:/) {
                next;
            } else {
                $line =~ s/(.*):.*/$1/;
            }
            print $line;
        }
    } else {
        die "Cannot open file.";
    }
}

readfile();
```

```
$ ./nostyle.pl
all
clean
dist
```

Mit Syntax-Highlighting

```
#!/usr/bin/perl -w

sub readfile {
    my $ok = open (FH, "<Makefile");
    if (defined $ok) {
        while ($line = <FH>) {
            if ($line !~ /^\\w*:/) {
                next;
            } else {
                $line =~ s/(.*):.*/$1/;
            }
            print $line;
        }
    } else {
        die "Cannot open file.";
    }
}

readfile();
```

- -> open or die

open or die

```
#!/usr/bin/perl -w

sub readfile {
    open (FH, "<Makefile") or die "Cannot open file.";
    while ($line = <FH>) {
        if ($line !~ /^\\w*/) {
            next;
        } else {
            $line =~ s/(.*)\\.*/$1/;
        }
        print $line;
    }
}

readfile();
```

- -> Funktionsklammern

Funktionsklammern weglassen

```
#!/usr/bin/perl -w

sub readfile {
    open FH, "<Makefile" or die "Cannot open file.";
    while ($line = <FH>) {
        if ($line !~ /^\\w*/) {
            next;
        } else {
            $line =~ s/(.*)\\.*/$1/;
        }
        print $line;
    }
}

readfile;
```

- -> Standard-Iterator

Standard-Iterator \$_ verwenden

```
#!/usr/bin/perl -w

sub readfile {
    open FH, "<Makefile" or die "Cannot open file.";
    while (<FH>) {
        if ($_ !~ /^\/\w*:/) {
            next;
        } else {
            s/(.*)::*/$1/;
        }
        print;
    }
}

readfile;
```

- -> nachgestelltes if

Nachgestelltes if

```
#!/usr/bin/perl -w

sub readfile {
    open FH, "<Makefile" or die "Cannot open file.";
    while (<FH>) {
        next if ($_ !~ /^w*:/);
        s/(.*)..*/$1/;
        print;
    }
}

readfile;
```

- -> if ! wird unless

if not vs. unless

```
#!/usr/bin/perl -w

sub readfile {
    open FH, "<Makefile" or die "Cannot open file.";
    while (<FH>) {
        next unless (/^\w*:/);
        s/(.*):.*/$1/;
        print;
    }
}

readfile;
```

- -> Klammern weg bei unless

Klammern weglassen

```
#!/usr/bin/perl -w

sub readfile {
    open FH, "<Makefile" or die "Cannot open file.";
    while (<FH>) {
        next unless /^\\w*:/;
        s/(.*)\\.*/$1/;
        print;
    }
}

readfile;
```

- eigentlich ok
- -> Kombination match/subst

Kombination match und subst

```
#!/usr/bin/perl -w

sub readfile {
    open FH, "<Makefile" or die "Cannot open file.";
    while (<FH>) {
        print if /^w*/ and s/(.*)..*/$1/;
    }
}

readfile;
```

- Geschmackssache
- -> zu noisy? match+subst optimieren

subst enthält implizites match

```
#!/usr/bin/perl -w

sub readfile {
    open FH, "<Makefile" or die "Cannot open file.";
    while (<FH>) {
        print if s/^(\\w*):.*$/1/;
    }
}

readfile;
```

- wieder ok



Perl Fun

Acme::Bleach

\$ —



Acme::Bleach

```
$ cat hellobleach.pl _
```

Acme::Bleach

```
$ cat hellobleach.pl  
use Acme::Bleach;  
print "Hello world.\n";  
$ _
```

Acme::Bleach

```
$ cat hellobleach.pl  
use Acme::Bleach;  
print "Hello world.\n";  
$ perl hellobleach.pl _
```

Acme::Bleach

```
$ cat hellobleach.pl
use Acme::Bleach;
print "Hello world.\n";
$ perl hellobleach.pl
$ _
```

Acme::Bleach

```
$ cat hellobleach.pl  
use Acme::Bleach;  
print "Hello world.\n";  
$ perl hellobleach.pl  
$ perl hellobleach.pl _
```

Acme::Bleach

```
$ cat hellobleach.pl
use Acme::Bleach;
print "Hello world.\n";
$ perl hellobleach.pl
$ perl hellobleach.pl
Hello world.
$ _
```

Acme::Bleach

```
$ cat hellobleach.pl
use Acme::Bleach;
print "Hello world.\n";
$ perl hellobleach.pl
$ perl hellobleach.pl
Hello world.
$ cat hellobleach.pl _
```

Acme::Bleach

```
$ cat hellobleach.pl
use Acme::Bleach;
print "Hello world.\n";
$ perl hellobleach.pl
$ perl hellobleach.pl
Hello world.
$ cat hellobleach.pl
use Acme::Bleach;
```

```
$ _
```

Acme::Bleach

```
$ cat hellobleach.pl
use Acme::Bleach;
print "Hello world.\n";
$ perl hellobleach.pl
$ perl hellobleach.pl
Hello world.
$ cat hellobleach.pl
use Acme::Bleach;
```

```
$ perl hellobleach.pl _
```

Acme::Bleach

```
$ cat hellobleach.pl
use Acme::Bleach;
print "Hello world.\n";
$ perl hellobleach.pl
$ perl hellobleach.pl
Hello world.
$ cat hellobleach.pl
use Acme::Bleach;
```

```
$ perl hellobleach.pl
Hello world.
$ _
```

Acme::Bleach

```
$ cat hellobleach.pl
use Acme::Bleach;
print "Hello world.\n";
$ perl hellobleach.pl
$ perl hellobleach.pl
Hello world.
$ cat hellobleach.pl
use Acme::Bleach;
```

```
$ perl hellobleach.pl
Hello world.
$ _
```

- Autor: Damian Conway

Acme::DoubleHelix

\$ —



Acme::DoubleHelix

```
$ cat hellohelix.pl _
```



Acme::DoubleHelix

```
$ cat hellohelix.pl  
use Acme::DoubleHelix;  
print "Hello world.\n";  
$ _
```



Acme::DoubleHelix

```
$ cat hellohelix.pl  
use Acme::DoubleHelix;  
print "Hello world.\n";  
$ perl hellohelix.pl _
```

Acme::DoubleHelix

```
$ cat hellohelix.pl
use Acme::DoubleHelix;
print "Hello world.\n";
$ perl hellohelix.pl _
$ _
```

Acme::DoubleHelix

```
$ cat hellohelix.pl
use Acme::DoubleHelix;
print "Hello world.\n";
$ perl hellohelix.pl
$ perl hellohelix.pl _
```

Acme::DoubleHelix

```
$ cat hellohelix.pl
use Acme::DoubleHelix;
print "Hello world.\n";
$ perl hellohelix.pl
$ perl hellohelix.pl
Hello world.
$ _
```

Acme::DoubleHelix

```
$ cat hellohelix.pl
use Acme::DoubleHelix;
print "Hello world.\n";
$ perl hellohelix.pl
$ perl hellohelix.pl
Hello world.
$ cat hellohelix.pl _
```

Acme::DoubleHelix

```
$ cat hellohelix.pl
use Acme::DoubleHelix;
print "Hello world.\n";
$ perl hellohelix.pl
$ perl hellohelix.pl
Hello world.
$ cat hellohelix.pl
use Acme::DoubleHelix;
```

```
CG
T--A
A---T
A----T
C----G
  T----A
    A---T
      G--C
        AT
          CG
            C--G
              G---C
                G----C
                  C----G
                    A----T
                      C---G
                        [...]
```

Acme::Lingua::NIGERIAN



Acme::Lingua::NIGERIAN

use Acme::Lingua::NIGERIAN;

DEAR SIR,

I AM THE SON OF LATE PRESIDENT ONE-OR-THE-OTHER OF NIGERIA.

TRANSFER_DISCREETLY SWISS_BANK_ACCOUNT "H!!!ELLO N!!!IGERIA\n";

TRANSFER_DISCREETLY SWISS_BANK_ACCOUNT US\$17 MILLION, "\n";

TRANSFER_DISCREETLY SWISS_BANK_ACCOUNT US\$17 MILLION + 25, "\n";

Acme::Lingua::NIGERIAN

```
use Acme::Lingua::NIGERIAN;
```

```
DEAR SIR,
```

```
I AM THE SON OF LATE PRESIDENT ONE-OR-THE-OTHER OF NIGERIA.
```

```
TRANSFER_DISCREETLY SWISS_BANK_ACCOUNT "H!!!ELLO N!!!IGERIA\n";
```

```
TRANSFER_DISCREETLY SWISS_BANK_ACCOUNT US$17 MILLION, "\n";
```

```
TRANSFER_DISCREETLY SWISS_BANK_ACCOUNT US$17 MILLION + 25, "\n";
```

```
$ perl hellonigerian.pl
```

```
Hello Nigeria
```

```
0
```

```
25
```

Lingua::Romana::Perligata



Lingua::Romana::Perligata

- Autor: Damian Conway



Lingua::Romana::Perligata

- Autor: Damian Conway

use **Lingua::Romana::Perligata**;

adnota Illud Cribrum Eratotheris

maximum tum val inquementum, tum biguttam, tum stadium egresso scribe.

vestibulo perlegementum da meo maximo .

maximum tum novumversum egresso scribe.

da II tum maximum conscribementa meis listis.

dum damentum nexto listis decapitamentum, fac sic

lista sic hoc tum nextum, recidementum cis vannementa da listis.

next tum biguttam, tum stadium, tum nextum, tum novumversum,
scribe egresso.

cis

Lingua::Romana::Perligata

- Autor: Damian Conway

```
use Lingua::Romana::Perligata;
```

```
adnota Illud Cribrum Eratothernis
```

```
maximum tum val inquementum, tum biguttam, tum stadium egresso scribe.
```

```
vestibulo perlegementum da meo maximo .
```

```
maximum tum novumversum egresso scribe.
```

```
da II tum maximum conscribementa meis listis.
```

```
dum damentum nexto listis decapitamentum, fac sic
```

```
    lista sic hoc tum nextum, recidementum cis vannementa da listis.
```

```
    next tum biguttam, tum stadium, tum nextum, tum novumversum,  
        scribe egresso.
```

```
cis
```

```
$ perl perligata.pl
```

```
maximum val:      10
```

```
10
```

```
next:      2
```

```
next:      3
```

```
next:      5
```

```
next:      7
```



Sinnvolle Anwendungen

Der Whitespace-Disput

- Syntaktischer Whitespace
- Wer's mag, muss zu Python greifen, oder?
- Drüber streiten?
- Fehlt was in Perl, gibt's bestimmt ein CPAN-Modul ...

Der Whitespace-Disput

- Syntaktischer Whitespace
- Wer's mag, muss zu Python greifen, oder?
- Drüber streiten?
- Fehlt was in Perl, gibt's bestimmt ein CPAN-Modul ...
- Und in der Tat:
- **Acme::Pythonic**

Pythonic Syntactic Whitespace

```
sub readfile {  
    open FH, "<Makefile" or die "Cannot open file."  
    while (<FH>) {  
        print if s/^(\\w*):.*$/1/  
    }  
}
```

```
readfile;
```

```
use Acme::Pythonic;
```

```
sub readfile:  
    open FH, "<Makefile" or die "Cannot open file."  
    while (<FH>):  
        print if s/^(\\w*):.*$/1/
```

```
readfile
```

Schliesst sich der Kreis?

"Haben Sie jemals über wirkliche Freiheiten nachgedacht?
Freiheiten von den Meinungen anderer?
Sogar von den EIGENEN Meinungen?"

-- Apocalypse Now

Schliesst sich der Kreis?

"Haben Sie jemals über wirkliche Freiheiten nachgedacht?
Freiheiten von den Meinungen anderer?
Sogar von den EIGENEN Meinungen?"

-- Apocalypse Now