Gültigkeit von Variablen in Perl

Grundsätzlich sollte in Perl immer `use strict;` verwendet werden. Damit ist der Geltungsbereich von Variablen eingeschränkt, und es verhindert Typos beim Programmieren.

In Perl können Variablen mit `my` und mit `our` definiert werden. Im Grunde sind beide Variablenarten "global", es hängt nur davon ab, wo sie definiert wurden. Auch eine `my`-Variable wird immer in alle Subs durchvererbt, die aus dem Bereich der Definiton der `my`-Variable aufgerufen werden. Gleiches gilt für Blöcke, das sind Bereiche, die mit {geschwungenen Klammern} eingefasst sind.

Wenn es möglich ist, sollte man `my` statt `our` verwenden. In Ausnahmefällen muss our verwendet werden. Als pragmatische Regel gilt: Wenn's mit my nicht geht, probiers mal mit our

Beispiele mit my

```
use strict;
my $name = "Christian";
printmyname();

sub printmyname
{
  print "My name is $name";
}

# Output:
# My name is Christian
```

Die mit `my` definierte Variable ist auch im Sub gültig.

```
use strict;
setmyname();
print "My name is $name";

sub setmyname
{
    my $name = "Christian";
}

# Output:
# Global symbol "$name" requires explicit package name at test.pl line 3.
# Execution of test.pl aborted due to compilation errors.
```

Wird die Variable im Sub definiert, ist sie im aufrufenden Code nicht verfügbar.

```
use strict;
my $name = "Christian";
```

```
printmyname();
print "main: My name is $name";

sub printmyname
{
    my $name = "Andreas";
    print "sub: My name is $name";
}

# Output:
# sub: My name is Andreas
# main: My name is Christian
```

Eine Variable kann im Scope eines Subs bzw. Blocks durch eine gleiche Variablendefinition nochmals definiert werden. Die ursprüngliche Variable im Hauptprogramm bleibt bestehen.

```
use strict;
my $name = "Christian";
printmyname();
print "main: My name is $name";

sub printmyname
{
    $name = "Andreas";
    print "sub: My name is $name";
}

# Output:
# sub: My name is Andreas
# main: My name is Andreas
```

Hier wird im Sub keine neue Variable definiert, sondern die aus dem aufrufenden Programm definierte Variable $name geändert.

```
use strict;
my @names = ("Christian", "Andreas", "Michael"); # @ definiert einen Array
foreach my $name (@names)
{
    print "My name is $name";
}
print "Current name is $name";

# Output:
# Global symbol "$name" requires explicit package name at test.pl line 7.
# Execution of test.pl aborted due to compilation errors.
```

Die Variable $name wird innerhalb der foreach-Schleife definiert und ist deswegen nur dort gültig. Außerhalb der Schleife ist $name nicht mehr definiert, deswegen der Perl-Fehler.

```
use strict;
my $name;
my @names = ("Christian", "Andreas", "Michael"); # @ definiert einen Array
foreach $name (@names)
{
    print "My name is $name";
}
print "Current name is $name";

# Output:
# My name is Christian
# My name is Andreas
# My name is Michael
# Current name is
```

Es kommt kein Fehler mehr, weil die Variable bereits zuvor definiert wurde. Dennoch enthält die Variable nicht den letzten Wert der foreach-Schleife.

```
use strict;
my $testcase = 1;
if ($testcase)
{
    my $error = 1;
}

print "Error code: $error";

# Output:
# Global symbol "$error" requires explicit package name at test.pl line 8.
# Execution of test.pl aborted due to compilation errors.
```

Bei der Fehlersuche: Nie vergessen, dass eine Variablendefinition innerhalb eines Blocks {} auch nur innerhalb des Blocks gilt.

```
use strict;
my $testcase = 1;
my $error;
if ($testcase)
{
    $error = 1;
}

print "Error code: $error";

# Output:
# Error code: 1
```

Soll eine Variable innerhalb eines Blocks gesetzt werden, muss diese außerhalb definiert, und innerhalb des Blocks gesetzt werden.