Fehler in PHP

Warning
- Programm läuft weiter
- abfangbar

Error
- Programm bricht ab
- abfangbar

Fatal Error
- Programm bricht ab
- nicht abfangbar

Parse Error
- Programm startet nicht
- nicht abfangbar

Exception
- Programm bricht ab
- abfangbar
- Moderne Fehlerhandling

Implizite Typumwandlung

String → Zahl (Ganzzahl od. Fließkomma)

Bsp.:

'4.2' → 4.2 (float)

'42' → 42 (int)

'4a' → 4 (gsf. Warning)

'abc' → Fatal Error / Error

'a4' → Fatal Error / Error

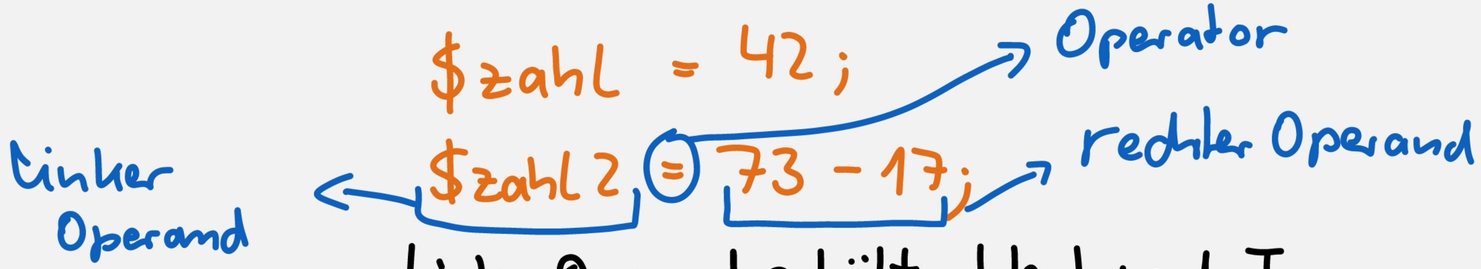
Konstanten in PHP

- Schlüsselwort const
- Konstanten sind unveränderlich
- Wert muss direkt bei Definition zugewiesen werden
- Namenskonventionen:
    - Wie bei Variablen, aber ohne $
- Best practice
    - Immer alles in Großbuchstaben

Zuweisungsoperatoren

$(3 + 7) \times 2 = 20$

- Einfache Zuweisung



Linker Operand erhält Wert und Typ des rechten Operanden

- Kombinierte Zuweisung

```
$var = $var + $var2
$var += $var2
```

} identisches Ergebnis

möglich mit + - * / ** % .

Verzweigung mit if/elseif/else

Syntax:

```
if (Bedingung) {
    Anweisungen;
}
elseif (Bedingung) {
    Anweisungen;
}
else {
    Anweisungen;
}
```

if (Bedingung) { Anweisungen; } — Pflicht

elseif (Bedingung) { Anweisungen; } — Optional, 0...∞

else { Anweisungen; } — Optional, 0...1mal

{} dürfen weggelassen werden, wenn nur eine Anweisung im Block ist. NICHT empfehlenswert!!

Bedingung ist immer ein boolescher Ausdruck (Wahrheitswert true/false)

z.B.:

```
1 == $var
true
3 == 5
$var  // bool oder implizite Typkonvertierung nach bool
```

Implizite Konvertierung nach bool

false ist:

- 0
- 0.0
- '0'
- '0.0'
- '' (Leerstring)
- null
- false

true ist

- alles andere

Vergleichsoperatoren

li == re — true, wenn Wert von li und re gleich ist

```
1 == 1      true
2 == 4      false
3 == '3'    true
```

li != re — true, wenn Wert von li und re nicht gleich ist

```
1 != 1      false
2 != 4      true
3 != '3'    false
```

li > re — größer als

li < re — kleiner als

li >= re — größer oder gleich

li <= re — kleiner oder gleich

li === re — true, wenn Wert und Typ von li und re gleich sind

```
1 === 1     true
2 === 4     false
3 === '3'   false
```

li !== re — Ungleichprüfung mit Typ-Kontrolle

Fallunterscheidung mit switch/case

- dient zum Vergleich eines Ausdrucks mit verschiedenen festen Werten
- Einzelne Anweisungsblöcke mit `break` abschließen
- Syntax:

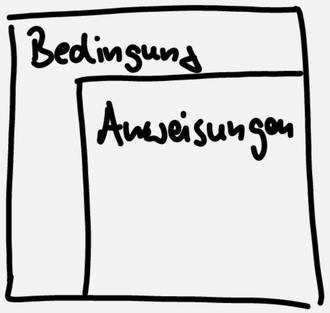
```
switch (Vergleichswert) {
    case Konstante:
        Anweisungen;
        break;
    case Konstante:
        Anweisungen;
        break;
    default:
        Anweisungen;
        break;
}
```

Schleife mit While

Syntax:

```
While ( Bedingung )
{
    Anweisungen;
}
```

- Wird solange ausgeführt, solange die Bedingung true ist
- kopfgesteuerte Schleife
- läuft min. Ø mal
- Laufbedingung



- Inkrementoperator ++ (erhöhen um eins)
  Dekrementoperator -- (verringern um eins)