

## Vorlesung



University of Applied Sciences

# DBSP

## Unit 7

### PHP IV

Operatoren und Kontrollstrukturen

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

1



University of Applied Sciences



**Prof. Dr. rer. nat.  
Nane Kratzke**

*Praktische Informatik und  
betriebliche Informationssysteme*

- Raum: 17-0.10
- Tel.: 0451 300 5549
- Email: [kratzke@fh-luebeck.de](mailto:kratzke@fh-luebeck.de)



@NaneKratzke

Updates der Handouts auch über Twitter #dbsp

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

2

## **Übergreifende Ziele der Lehrveranstaltung**



University of Applied Sciences

Client- und Serverseitige Entwicklung

PHP (Serverseitig)

JavaScript (Clientseitig)

„Hosten“ von Apps

Framework Erfahrungen

CMS (Drupal)

WebServices (Google-Maps)

jQuery

Datenbank-Integration

Berücksichtigung von Sicherheitsaspekten

HTML-Injections

SQL-Injections

Session Hijacking

Login-Systeme

Um sich weitere Web-Technologien autodidaktisch erarbeiten zu können.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

3

## **Units**



University of Applied Sciences

**Unit 1**  
Cloud Computing IaaS

**Unit 2**  
CMS Drupal

**Unit 3**  
HTML und CSS

**Unit 4 - 7**  
PHP I - IV

**Unit 8**  
Sessions, Cookies, Formulare und Login-System

**Unit 9**  
JavaScript

**Unit 10**  
Drupal Module Development

**Unit 11**  
Datenmodellierung

**Unit 12 - 13**  
Datenbanken und SQL  
Vom Datenmodell zur Datenbank

**Unit 14**  
Datenbank-gestützte Web-Anwendungen

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

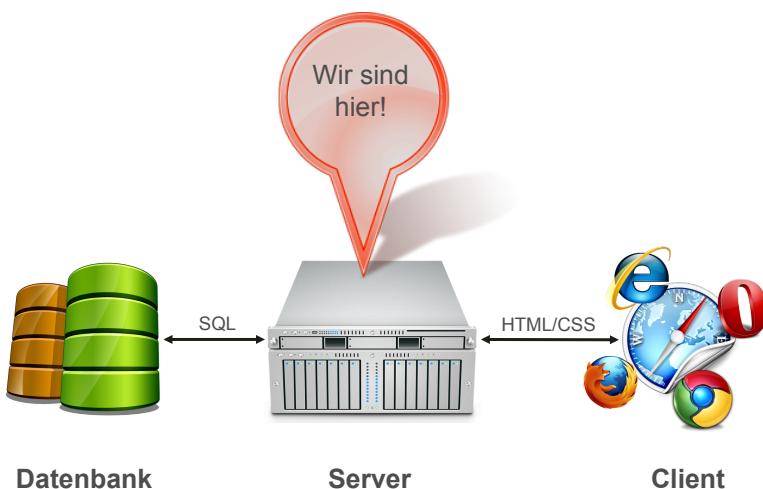
4

## Datenbank – Server – Client

Wo waren wir nochmal?



University of Applied Sciences



Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

5

## Zum Nachlesen ...



University of Applied Sciences



### Kapitel 4 – PHP Basics

Kapitel 4.5 bis 4.7

### Kapitel 5 – More Basics

Kapitel 5.1 bis 5.4

### Kapitel 6 – Funktionen für ...

Kapitel 6.2 und 6.3

**Kapitel 3**  
Variablen u. Datentypen  
Kapitel 3.8 bis 3.21

**Kapitel 4**  
Programmstrukturen  
Kapitel 4.1 bis 4.3

**Kapitel 5**  
Arrays  
Kapitel 5.1 bis 5.3



Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

6

## Zum Nachlesen ...



University of Applied Sciences



### Bereitgestellte Skripte:

#### **PHP Programmierung**

- **Kapitel 5:** Kontrollstrukturen
- **Kapitel 6:** Datentypen und Datenstrukturen
- **Kapitel 7:** Operatoren

<http://praktische-informatik.fh-luebeck.de/node/39>

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

7

## Inhalte der PHP Units



University of Applied Sciences

1

- Laufzeitmodelle und Programmierparadigma

2

- Elemente eines PHP-Programms

3

- Datentypen

4

- Operatoren

5

- Kontrollstrukturen

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

8

## Die gängigsten Operatorarten



University of Applied Sciences

### Klassische Operatoren

- Arithmetische Operatoren
- Relationale Operatoren
- Logische Operatoren
- Bedingte Auswertung
- Zuweisungsoperatoren

### Spezielle Operatoren

- String-Verkettung
- Array Operatoren

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

9

## Arithmetische Operatoren



University of Applied Sciences

Erwarten numerische Operanden



Liefern numerische Operanden



Dienen numerischen Berechnungen

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

10

## Liste arithmetischer Operatoren



University of Applied Sciences

Operator	Bezeichnung	Bedeutung
++	Pre-Inkrement Post-Inkrement	$\$a++$ = Erhöhung von $\$a$ um 1 nach Auswertung eines Ausdrucks. $++\$a$ = Erhöhung von $\$a$ um 1 vor Auswertung eines Ausdrucks
--	Pre-Dekrement Post-Dekrement	$\$a--$ = Reduktion von $\$a$ um 1 nach Auswertung eines Ausdrucks. $--\$a$ = Reduktion von $\$a$ um 1 vor Auswertung eines Ausdrucks
+	Summe	$\$a + \$b$ = Summe von $\$a$ und $\$b$
-	Subtraktion	$\$a - \$b$ = Differenz von $\$a$ und $\$b$
*	Multiplikation	$\$a * \$b$ = Produkt von $\$a$ und $\$b$
/	Division	$\$a / \$b$ = Quotient von $\$a$ und $\$b$ .
%	Restwert (Modulo)	$\$a \% \$b$ = Rest der ganzzahligen Division von $\$a$ durch $\$b$ .

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

11

## Miniübung zu arithmetischen Operatoren



University of Applied Sciences

```
$ai = 5; $bi = 2;  
$af = 5.0; $bf = 2.0;
```

```
echo $ai + $bi;
```

7

```
echo $ai / $bi;
```

2.5 (bei JAVA wäre das Ergebnis 2!)

```
echo $af / $bi;
```

2.5

```
echo $ai % $bi;
```

1

```
echo $ai--;
```

5

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

12

## Relationale Operatoren



University of Applied Sciences

Erwarten beliebige Ausdrücke

Liefern boolsche Werte

Dienen dem Vergleich von Ausdrücken

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

13

## Liste relationaler Operatoren



University of Applied Sciences

Operator	Bezeichnung	Bedeutung
<code>==</code>	gleich	<code>\$a == \$b</code> ergibt true wenn \$a und \$b gleich sind.
<code>===</code>	Identisch	<code>\$a === \$b</code> ergibt true, wenn \$a und \$b gleich sind und beide vom gleichen Typ sind.
<code>!=</code> <code>&lt;&gt;</code>	ungleich	<code>\$a != \$b</code> ergibt true, wenn \$a und \$b nicht gleich sind.
<code>!==</code>	Nicht identisch	<code>\$a !== \$b</code> ergibt true, wenn \$a und \$b nicht gleich sind oder nicht vom gleichen Typ sind.
<code>&lt;</code>	kleiner	<code>\$a &lt; \$b</code> ergibt true wenn \$a kleiner als \$b ist.
<code>&gt;</code>	größer	<code>\$a &gt; \$b</code> ergibt true wenn \$a größer als \$b ist.
<code>&lt;=</code>	Kleiner gleich	<code>\$a &lt;= \$b</code> ergibt true wenn \$a kleiner als \$b oder gleich \$b ist.
<code>&gt;=</code>	größer gleich	<code>\$a &gt;= \$b</code> ergibt true wenn \$a größer als \$b oder gleich \$b ist.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

14

## Miniübung zu relationalen Operatoren



University of Applied Sciences

```
$ai = 5; $bi = 2;  
$af = 5.0; $bf = 2.0;
```

```
var_dump($bi == $bf);
```

true

```
var_dump($ai !== $af);
```

true

```
var_dump($ai === $bi);
```

false

```
var_dump($ai === $af);
```

false

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

15

## Logische Operatoren



University of Applied Sciences

Erwarten boolesche Werte

Liefert boolesche Werte

Dienen der Formulierung logischer Bedingungen

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

16

## Liste logischer Operatoren



University of Applied Sciences

Operator	Bezeichnung	Bedeutung
!	Logisches NICHT	<code>!\$a</code> ergibt true wenn <code>\$a</code> false ist und true, wenn <code>\$a</code> true ist.
&& and	UND	<code>\$a &amp;&amp; \$b</code> ergibt true, wenn <code>\$a</code> und <code>\$b</code> true sind.
 or	ODER	<code>\$a    \$b</code> ergibt true, wenn mindestens einer der beiden Ausdrücke <code>\$a</code> oder <code>\$b</code> wahr ist. Ist bereits <code>\$a</code> wahr wird <code>\$b</code> nicht mehr ausgewertet. (logisches Oder)
xor	Exklusiv-ODER	<code>\$a xor \$b</code> ergibt true wenn <code>\$a</code> und <code>\$b</code> einen unterschiedlichen Wahrheitswert haben (sprachliches Oder)

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

17

## Miniübung zu logischen Operatoren



University of Applied Sciences

```
$a = true;  
$b = false;
```

```
var_dump( !$a );
```

false

```
var_dump( !$b );
```

true

```
var_dump($a and $b);
```

false

```
var_dump($b || $a);
```

true

```
var_dump($b xor $a);
```

true

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

18

## Miniübung zu logischen Operatoren (Short Circuit Verhalten)



University of Applied Sciences

```
$a = false;  
$c = true;  
$x = 1;
```

```
var_dump($a && 5 / --$x == 0);
```

false

```
var_dump($a & 5 / --$x == 0);
```

Division by Zero!

```
var_dump($c || 5 / --$x == 0);
```

true

```
var_dump($c | 5 / --$x == 0);
```

Division by Zero!

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

19

## Bedingte Auswertung **\$a ? \$b : \$c**



University of Applied Sciences

- Der Fragezeichen Operator ?: ist der einzige dreiwertige Operator
- Kann häufig eingesetzt werden, um if-Abfragen zu vermeiden.
- \$a ? \$b : \$c
  - Ist \$a true wird \$b zurückgeliefert
  - Ist \$a false wird \$c zurückgeliefert

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

20

## Miniübung: Bedingte Auswertung



```
$a = 6;  
$b = 2;  
echo ($a % 2 == 0 ? "Hello" : "World");
```

Hello

```
$hw = "Hello World";  
$h = "Hello";  
$ew = function($s,$n) { return strpos($s, $n) ==  
           strlen($s) - strlen($n); };  
echo ($ew($hw, $h) ? $h : $hw);
```

Hello World

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

21

## Zuweisungs-Operatoren



Erwarten Ausdrücke

Liefern boolesche oder numerische Werte

Dienen der Zuweisung von ausgewerteten  
Ausdrücken an Variablen

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

22

## Liste der Zuweisungsoperatoren



University of Applied Sciences

Operator	Bezeichnung	Bedeutung
=	Einfache Zuweisung	$\$a = \$b$ weist $\$a$ den Wert von $\$b$ .
+=	Additionszuweisung	$\$a += \$b$ weist $\$a$ den Wert von $\$a + \$b$ zu. $\$a += \$b$ identisch zu $\$a = \$a + \$b$
-=	Subtraktionszuweisung	Analog $+=$ Operator mit $-$
*=	Multiplikationszuweisung	Analog $+=$ Operator mit $*$
/=	Divisionszuweisung	Analog $+=$ Operator mit $/$
%=	Modulozuweisung	Analog $+=$ Operator mit $\%$

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

23

## Miniübung zu Zuweisungs-Operatoren



University of Applied Sciences

`$a = 5;`

5

`$a += 2;`

7

`$a %= 2;`

1

`$a -= -2;`

3

`$a /= 3;`

1

`$a *= 2;`

2

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

24

## Die gängigsten Operatorarten



University of Applied Sciences

### Klassische Operatoren

- Arithmetische Operatoren
- Relationale Operatoren
- Logische Operatoren
- Bedingte Auswertung
- Zuweisungsoperatoren

### Spezielle Operatoren

- String-Verkettung
- Array Operatoren

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

25

## String Verkettung (Konkatenation)



University of Applied Sciences

- Zwei Strings in PHP können mittels des .-Operators verknüpft werden.

```
$a = "Nice ";
$b = 2;
$c = " meet you";
echo $a.$b.$c;
```

Nice 2 meet you

- Bei der Operation wird ggf. ein Nichtstring Operand in einen String gewandelt.
- Der Operator existiert auch in Kombination mit dem Zuweisungsoperator als .=-Operator.

```
$ausgabe = $a.$b.$c;
$ausgabe .= " again";
echo $ausgabe;
```

Nice 2 meet you
again

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

26

## Liste der Array Operatoren



University of Applied Sciences

Operator	Bezeichnung	Bedeutung
+	„Vereinigung“	$\$a + \$b$ Vereinigung von $\$a$ und $\$b$ .
==	Gleichheit	$\$a == \$b$ wenn $\$a$ und $\$b$ dieselben key/value Paare haben
====	Identität	$\$a === \$b$ , wenn $\$a$ und $\$b$ dieselben key/value Paare in derselben Reihenfolge haben
!= <>	Ungleichheit	Es gilt die Invariante $(\$a != \$b) == !(\$a == \$b)$
!==	Nicht identisch	Es gilt die Invariante $(\$a !== \$b) == !(\$a === \$b)$
[]	Anhängoperator	$\$a[] = \$v$ hängt $\$v$ an der letzten Position des Arrays von $\$a$ an.

### Vorsicht:

Der Vereinigungsoperator + bei Arrays hat eine wenig intuitive Semantik.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

27

## Semantik des +-Operators bei Arrays an einem Beispiel



University of Applied Sciences

```
$a = array("a"=>"apple", "b"=>"banana");
$b = array("a"=>"pear", "b"=>"strawberry", "c"=>"cherry");
```

```
var_dump($a + $b);
```

```
Array("a"=>"apple"
      "b"=>"banana"
      "c"=>"cherry")
```

```
var_dump($b + $a);
```

```
Array("a"=>"pear"
      "b"=>"strawberry"
      "c"=>"cherry")
```

Der + Operator hängt verbleibende Key/Value Paare des rechts vom Operator stehenden Arrays an das links vom Operator stehende Array.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

28

## Mini-Übung: Array Operatoren



University of Applied Sciences

```
$a = array(1, 2, 3);  
$b = array(2, 3, 4, 5);
```

```
$a + $b;
```

array(1,2,3,5)

```
$a[] = 6;
```

array(1,2,3,6)

```
$b[] = 7;
```

array(2,3,4,5,7)

```
$b + $a;
```

array(2,3,4,5,7)

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

29

## Vorrangregeln



University of Applied Sciences

Kennen Sie noch die Regeln:  
Punktrechnung vor  
Strichrechnung?

PHP kennt dasselbe mit ein paar mehr zu berücksichtigenden Operatoren.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

30

## Vorrangregeln Operatorgruppen



University of Applied Sciences

Assoziativität	Operatoren	Bezeichnung
rechts	<code>++</code> , <code>--</code> , <code>!</code>	Inkrement, Dekrement, Nicht
links	<code>*</code> , <code>/</code> , <code>%</code>	Multiplikation, Division, Modulo
links	<code>+</code> , <code>-</code> , <code>.</code>	Addition, Subtraktion, String-Konkatenation
-	<code>&lt;</code> , <code>&lt;=</code> , <code>&gt;</code> , <code>&gt;=</code>	Kleiner, Kleiner-Gleich, Größer, Größer-Gleich
-	<code>==</code> , <code>!=</code> , <code>====</code> , <code>!</code> <code>==</code>	Gleich, Ungleich, Identität, Nicht-Identität
links	<code>&amp;&amp;</code>	Logisches Und
links	<code>  </code>	Logisches Oder
links	<code>?:</code>	Bedingte Auswertung
rechts	<code>=</code> , <code>+=</code> , <code>-=</code> , <code>*=</code> , ...	Zuweisungen
links	<code>and</code>	Logisches Und
links	<code>or</code>	Logisches Oder
links	<code>xor</code>	Exklusiv Oder

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

31

## Inhalte der PHP Units



University of Applied Sciences

- 1** • Laufzeitmodelle und Programmierparadigma
- 2** • Elemente eines PHP-Programms
- 3** • Datentypen
- 4** • Operatoren
- 5** • Kontrollstrukturen

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

32

## Kontrollstrukturen



University of Applied Sciences

### Verzweigungen

- if
- switch

### Schleifen

- while
- do
- for
- foreach

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

33

### Verzweigungen if-Abfrage



University of Applied Sciences

- Verzweigungen dienen dazu bestimmte Programmteile nur beim Eintreten vorgegebener Bedingungen auszuführen.

If Variante

```
if ($ausdruck)  
    anweisung;
```

If Else Variante

```
if ($ausdruck)  
    anweisung;  
else  
    anweisung;
```

If Block Variante

```
if ($ausdruck) {  
    Block von Anweisungen;  
}
```

If Else Block Variante

```
if ($ausdruck) {  
    Block von Anweisungen;  
} else {  
    Block von Anweisungen;  
}
```

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

34

## Verzweigungen Switch-Anweisung



University of Applied Sciences

- Switch Anweisung ist eine Mehrfachverzweigung.
- sie wertet einen im Ergebnis ganzzahligen Ausdruck aus
- und springt einen case Zweig oder den default Zweig an.

### Syntax:

```
switch ($ausdruck) {  
    case Konstante: anweisung;  
    ...  
    default: anweisung;  
}
```

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

35

## Verzweigungen Switch-Anweisung (Beispiel)



University of Applied Sciences

Welche Ausgabe erzeugt dieser Code?

```
$i = 4;  
  
switch (i % 3) {  
    case 1: echo "Rest 1";  
    case 2: echo "Rest 2";  
    case 3: echo "Rest 3";  
    default: echo "Rest 0";  
}
```

- |        |   |
|--------|---|
| Rest 1 | Achtung: Nachdem ein case- oder default-Label angesprungen wurde, werden alle dahinter stehenden Anweisungen ausgeführt.  |
| Rest 2 |   |
| Rest 3 |   |
| Rest 0 | Will man das nicht, muss man das Label mit einer break Anweisung dazu zwingen, am Ende der switch-Anweisung fortzusetzen. |

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

36

## Verzweigungen Switch-Anweisung (Beispiel)



University of Applied Sciences

Welche Ausgabe erzeugt dieser Code?

```
$i = 4;

switch ($i % 3) {
    case 1: echo "Rest 1"; break;
    case 2: echo "Rest 2"; break;
    case 3: echo "Rest 3"; break;
    default: echo "Rest 0";
}
```

Rest 1

Die ergänzende break Anweisung realisiert die Semantik der switch Anweisung, wie man sie intuitiv erwarten würde.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

37

## Kontrollstrukturen



University of Applied Sciences

### Verzweigungen

- if
- switch

### Schleifen

- while
- do
- for
- foreach

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

38

## Abweisende Schleife



University of Applied Sciences

### Syntax:

```
while ($ausdruck) {  
    anweisung;  
}
```

- Prüfen des Ausdrucks
- Solange dieser True ist, wird der Anweisungsblock oder Einzelanweisung ausgeführt
- Ist der Ausdruck bereits zu Beginn false wird der Anweisungsblock nicht ausgeführt, daher **abweisende Schleife**.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

39

## Abweisende Schleife Mini-Übung: while Schleifen



University of Applied Sciences

```
$i = 0;  
while ($i < 4) { echo $i++; }
```

0  
1  
2

```
$i = 0;  
while ($i < 0) { echo $i++; }
```

Keine Ausgabe

```
$i = 0;  
while ($i <= 0) { echo --$i; }
```

-1  
-2  
-3  
... endet nie!

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

40

## Nicht abweisende Schleife



University of Applied Sciences

### Syntax:

```
do {  
    anweisung;  
} while ($ausdruck);
```

- Der Anweisungsblock wird mindestens einmal ausgeführt, daher **nicht abweisende Schleife**.
- Prüfen des Ausdrucks
  - Ist dieser True, wird der Anweisungsblock oder Einzelanweisung wieder ausgeführt
  - Ist dieser false wird die Programmausführung hinter dem while Ausdruck fortgesetzt.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

41

## Nicht abweisende Schleife Mini-Übung: do while Schleifen



University of Applied Sciences

```
$i = 0;  
do {  
    echo $i++;  
} while ($i < 4);
```

0  
1  
2  
3

```
$i = 0;  
do {  
    echo $i++;  
    echo -$i;  
} while ($i < 4);
```

0  
0  
0  
... endet nie!

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

42

## Zählschleife for – klassische Variante



University of Applied Sciences

### Syntax:

```
for (init; test; update) {  
    anweisung;  
}
```

#### Init Ausdruck

- Aufruf einmalig **vor Start** der Schleife
- Optional
- mehrere kommasseparierte Ausdrücke mögl.
- Variablen Deklaration möglich

#### Test Ausdruck

- Aufruf **jew. am Anfang** einer Schleife
- Optional, wenn nicht angegeben wird true gesetzt
- Schleife wird nur ausgeführt, wenn Ausdruck true

#### Update Ausdruck

- Aufruf **jew. am Ende** der Schleife
- Optional
- Mehrere kommasseparierte Ausdrücke möglich
- Dient dazu den Schleifenzähler zu ändern

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

43

## Zählschleife for – klassische Variante (Beispiele)



University of Applied Sciences

```
for ($i = 1; $i <= 3; $i++) { echo $i };
```

123

```
$i = 1;  
while (TRUE) {  
    if (!$i <= 3) break;  
    echo $i;  
    $i++;  
}
```

```
1 //init-Ausdruck – einmalig  
2 //test-Ausdruck – vor Schleifenlauf  
3 //Update-Ausdruck – am Ende
```

Jede for-Schleife kann nach diesem Muster umformuliert werden.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

44

## Zählschleife foreach Variante



University of Applied Sciences

### Syntax:

```
foreach ($array as [$key =>] $value) {  
    anweisung;  
}
```

- **\$key** und **\$value** sind „Laufvariablen“
- **\$array** ist ein Array (d.h. eine Menge von Key/Value Paaren).
- Zu lesen ist dieser Ausdruck als **for each \$value in \$array**, d.h.
  - es werden alle Key/Value Paare aus einem Array
  - elementweise durchlaufen
- **\$key** muss nicht angegeben werden, dann werden nur alle Werte eines Arrays durchlaufen

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

45

## Foreach Schleife Mini-Übung: foreach Schleifen



University of Applied Sciences

```
$as = array(7, 5, 3, 1);  
foreach ($as as $a) { echo $a; }
```

7 5 3 1

```
$as = array('a' => "Hello",  
           'b' => "",  
           'c' => "World!");  
  
foreach ($as as $k => $v) {  
    echo "$k => $v";  
}
```

a => Hello  
b =>  
c => World

```
$as = array(7, 5, 3, 1);  
foreach ($as as $k => $v) {  
    echo "$k => $v";  
}
```

0 => 7  
1 => 5  
2 => 3  
3 => 1

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

46

## Zusammenfassung



University of Applied Sciences

- **Klassische Operatoren**
  - Arithmetische Operatoren
  - Relationale Operatoren
  - Logische Operatoren
  - Bedingte Auswertung
  - Zuweisungsoperatoren
- **Spezielle Operatoren**
  - String-Verkettung
  - Array Operatoren
- **Kontrollstrukturen**
  - while
  - do
  - for
  - foreach

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

47