PHP
Verzweigungen

Verzweigungen: if-then-else

```php
$wert = -5.3;

if($wert<0) {
    $farbe='#FF0000';
} else {
    if($wert>0) {
        $farbe='#00FF00';
    } else {
        $farbe='#000000';
    }
}

echo '<span style="color: '.$farbe.'">Test</span>';
```

Test

Verzweigungen: if-elseif-else

```php
$wert = -5.3;

if($wert<0) {
    $farbe='#FF0000';
} else if($wert>0) {
    $farbe='#00FF00';
} else {
    $farbe='#000000';
}

echo '<span style="color: '.$farbe.'">Test</span>';
```

```
Test
```

Verzweigungen: Das Fragezeichen

- Ähnlich wie Java besitzt auch PHP den Trinitätsoperator „?“.
- Programmier-Anfänger halten den Fragezeichen-Operator für kryptisch und den erzeugten Quellcode für schlecht lesbar.
- Profis verwenden gerade diesen Operator jedoch gern, um kompakten Quellcode zu verfassen.
- Überlegen Sie am Besten, wer in Zukunft Einblick in Ihren Quellcode bekommt...

```
$wert = -5.3;
($wert<0) ? $farbe='#FF0000' : $farbe='#000000';

echo '<span style="color: '.$farbe.'">Test</span>';
```

```
Test
```

Verzweigungen: switch

```php
$note = 2;

switch($note) {
    case 1:
    $noteText="Sehr gut"; break;
    case 2:
    $noteText="Gut"; break;
    case 3:
    $noteText="Befriedigend"; break;
    case 4:
    $noteText="Ausreichend"; break;
    case 5:
    $noteText="Mangelhaft"; break;
    case 6:
    $noteText="Ungenügend"; break;
    default:
    $noteText="FEHLER"; break;
}

echo $noteText;
```

Verzweigungen: switch - fall through

```php
$note = 2;

switch($note) {
    case 1: // fall through
    case 2:
    case 3:
    case 4:
    $noteText="Bestanden"; break;
    case 5:
    case 6:
    $noteText="Durchgefallen"; break;
    default:
    $noteText="FEHLER"; break;
}

echo $noteText;
```

PHP
Schleifen

Schleifen: Die for-Schleife

- Beispiel mit einer dynamisch aufgebauten Tabelle:

```
echo "<table><tbody>";

for($i=0; $i < 10; $i++) {
  echo "<tr><td>Zeile</td><td>" . $i . "</td></tr>";
}

echo "</tbody></table>";
```

Zeile	0
Zeile	1
Zeile	2
Zeile	3
Zeile	4
Zeile	5
Zeile	6
Zeile	7
Zeile	8
Zeile	9

Schleifen: Die for-each-Schleife

- Durchläuft ein Array elementweise:

```php
$a = array("A", "B", "C", "D", "E");

foreach ($a as $element) {
    echo $element . " ";
}

echo "<br>";

foreach ($a as $key => $element) {
    echo $key . ": " . $element . ", ";
}
```

```
A B C D E
0: A, 1: B, 2: C, 3: D, 4: E,
```

Schleifen: Die while-Schleife

- Wird so lange ausgeführt, bis die Bedingungen false ergibt:

```php
$a=12;
$b=23;

while ($a<$b){
    echo 'Der Wert '.$a.' ist kleiner als '.$b.'<br>';
    $a++;
}
```

```
Der Wert 12 ist kleiner als 23
Der Wert 13 ist kleiner als 23
Der Wert 14 ist kleiner als 23
Der Wert 15 ist kleiner als 23
Der Wert 16 ist kleiner als 23
Der Wert 17 ist kleiner als 23
Der Wert 18 ist kleiner als 23
Der Wert 19 ist kleiner als 23
Der Wert 20 ist kleiner als 23
Der Wert 21 ist kleiner als 23
Der Wert 22 ist kleiner als 23
```

Schleifen:
Die do-while-Schleife und Zufallszahlen

- Erst ausführen, dann prüfen:

```php
$a = array(1, 2, 3);

do {
    // Hole Zufallszahl zwischen 1 und 5
    $number = rand(1, 5);
    echo "Teste " . $number . "<br>";
} while(in_array($number, $a));
    // Wiederhole so oft, bis Zufallszahl nicht mehr im Array enthalten ist

echo $number;
```

```
Teste 2
Teste 2
Teste 2
Teste 5
5
```

Schleifen: break & continue und Zufallszahlen

```php
$stop = rand(1, 100);

for($i=0; $i < 100; $i++) {
    if($i % 2 == 0) {
        // Alle geraden Zahlen überspringen
        continue;
    } else {
        // Wenn Zufallszahl erreicht ist, Schleife stoppen
        // Geht nur, wenn Zufallszahl ungerade ist
        if($i == $stop) {
            break;
        }
    }
}

echo "Bei $i gestoppt.";
```

```
Bei 93 gestoppt.
```

PHP
Funktionen

eigene Funktionen

- Eigene Funktionen werden mit function definiert:
  - `function greet1($vorname, $nachname) { ... }`
- Optionale Parameter sind am Ende der Parameterliste möglich:
  - `function greet2($name, $greeting = "Hallo") { ... }`
- PHP unterstützt in Funktionen per Default Call-by-Value, jedoch auch Call-by-Reference in einer C-ähnlichen Syntax:
  - `function promotion(&$nombre) { ... }`

eigene Funktionen und globale Variablen

- Globale Variablen sind nicht automatisch in Funktionen sichtbar!
- Sie müssen mit Schlüsselwort **`global`** bekannt gemacht werden!

```php
function tuwas() {
    echo $a;
}
$a = "Hallo";
tuwas(); // -> keine Ausgabe


function tuwas() {
    global $a;
    echo $a;
}
$a = "Hallo";
tuwas(); // -> Hallo
```

eigene Funktionen: Beispiel einer Wertübergabe

```php
function greet1($vorname, $nachname) {
    echo "Hallo $vorname $nachname";
    return true; // Rückgabewert
}

greet1("Frank", "Dopatka"); // -> Hello Frank Dopatka

function greet2($name, $greeting = "Hallo") {
    echo $greeting, " ", $name;
}

greet2("Frank", "Gruezi"); // -> Gruezi Frank
greet2("Frank");           // -> Hallo Frank
```

eigene Funktionen: Beispiel einer Referenzübergabe

```php
function promotion(&$nombre) {
    $nombre = "Dr. " . $nombre;
}

$name = "Meier";
promotion($name);
echo $name; // -> Dr. Meier
```

Funktionen als Übergabe-Parameter u.a. für Arrays

```php
$data = Array();
$data[] = 6;
$data[] = 675;
$data[] = 46;
$data[] = 235;

function addiere(&$x){
    $x=$x+11;
}

array_walk($data, 'addiere');

var_dump($data);
```

```
array(4) { [0]=> int(17) [1]=> int(686) [2]=> int(57) [3]=> int(246) }
```