PHP
Arrays

Arrays

- Arrays in PHP sind dynamisch; es ist keine Deklaration erforderlich.
- Die Elemente werden automatisch an das Ende angehängt:
  - `$a[] = 17;`
- Arrays in PHP sind wahlweise assoziativ oder indiziert.
  - Assoziativ: `$a["key"] = 5;`
  - Indiziert: `$b[0] = 5;`
- Der Zugriff auf Array-Element erfolgt über
  - den Schlüssel: `$a["key"]`
  - oder den Index mit Index (0..count-1): `$b[0]`

Beispiel eines indizierten Arrays

```php
// Erzeugen eines indizierten Arrays
$a[] = 1;
$a[] = 2;
$a[] = 3;

echo $a[1]; // -> 2
echo count($a); // -> 3

$a[1] = "ersetzt";
echo $a[1]; // -> ersetzt

// iterieren über das Array mittels for-Schleife
for($i=0; $i < count($a); $i++) {
    echo ($a[$i]);
}
```

Beispiel eines indizierten Arrays: Keine Grenzen!

```php
$data = Array();
$data[2] = 0;
$data[-5] = 0.0;
$data[8] = "Hallo";

echo(var_dump($data));
```

```
array(3) { [2]=> int(0) [-5]=> float(0) [8]=> string(5) "Hallo" }
```

die for-each Schleife

- sie durchläuft elementweise das Array
- die Elemente dürfen von beliebigem Typ sein
- sie funktioniert sowohl für indizierte, als auch für assoziative Arrays
- Syntax: **`foreach ($array as $element) {...}`**

```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,
```

Beispiel eines assoziativen Arrays

```php
// assoziatives Array
$b["gehen"] = "go";
$b["schlafen"] = "sleep";
echo $b["gehen"];

echo "<br>";

// assoziatives Array
$c = array('gehen' => 'go', 'schlafen' => 'sleep');
echo $c["gehen"];

echo "<br>";

// iterieren über assoziatives Array mittels for-each Schleife
foreach ($c as $key => $value) {
    echo $key . "->" . $value . '<br>';
}
```

```
go
go
gehen->go
schlafen->sleep
```

Mehrdimensionale Arrays

- Mehrdimensionale Arrays sind in PHP Arrays in Arrays:

```php
// 2-dimensionales Array erzeugen
$professoren = array(
  array("Frank", "Dopatka"),
  array("Peter", "Knauber"),
  array("Wolfgang", "Schramm")
);

// Professoren ausgeben
echo "Professor " . $professoren[0][0] . " " . $professoren[0][1] . "<br>";
echo "Professor " . $professoren[1][0] . " " . $professoren[1][1] . "<br>";
echo "Professor " . $professoren[2][0] . " " . $professoren[2][1] . "<br>";
```

```
Professor Frank Dopatka
Professor Peter Knauber
Professor Wolfgang Schramm
```

Werte und Referenzen

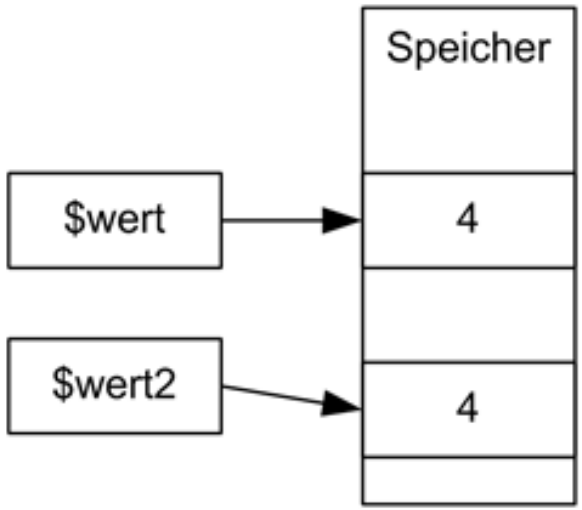
```php
$x="name";
$$x="daten";
echo($x.'<br/>');
echo($name.'<br/>');
```

```
name
daten
```

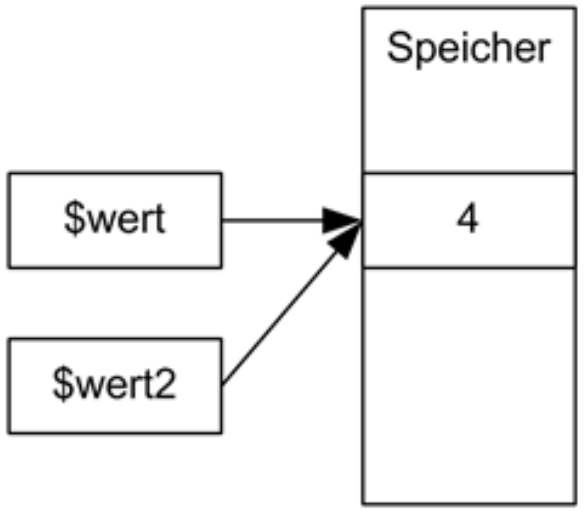
Werte und Referenzen

```php
$wert = 4;
echo($wert.'<br/>');
$wert2 = &$wert;
echo($wert2.'<br/>');
$wert2 = 99;
echo($wert.'<br/>');
echo($wert2.'<br/>');
```

```
4
4
99
99
```



$wert=$wert2;



$wert=&$wert2;

Array-Funktionen: Sortieren

Funktion	Bedeutung
shuffle($arr)	ordnet alle Elemente des Datenfeldes zufällig neu an
sort($arr)	sortiert ein eindimensionales Datenfeld vorwärts; war es ein assoziatives Feld, so wird es in ein nummerisches Feld umgewandelt
rsort($arr)	sortiert ein eindimensionales Datenfeld rückwärts; war es ein assoziatives Feld, so wird es in ein nummerisches Feld umgewandelt
asort($arr)	sortiert ein eindimensionales Datenfeld vorwärts und behält die Beziehungen in einem assoziativen Feld bei
arsort($arr)	sortiert ein eindimensionales Datenfeld rückwärts und behält die Beziehungen in einem assoziativen Feld bei

Funktion	Bedeutung
usort($arr, func)	sortiert ein eindimensionales Datenfeld nach einer eigenen Funktion
uasort($arr, func)	sortiert ein assoziatives Datenfeld nach einer eigenen Funktion

Array-Funktionen: Durchlaufen & Zugreifen

Funktion	Bedeutung
count($arr) oder size_of($arr)	gibt die Anzahl der Elemente im Datenfeld zurück
reset($arr)	setzt den internen Zeiger im Datenfeld auf das erste Element
end($arr)	setzt den internen Zeiger im Datenfeld auf das letzte Element
current($arr) oder pos($arr)	gibt den Inhalt des Elementes zurück, auf dem der Zeiger steht
key($arr)	gibt den Index des Elementes zurück, auf dem der Zeiger steht
next($arr)	setzt den internen Zeiger im Datenfeld um 1 nach vorne
prev($arr)	setzt den internen Zeiger im Datenfeld um 1 zurück
array_walk($arr, func)	wendet eine selbst definierte Funktion auf jedes Element des Datenfeldes an

Array-Funktionen: weitere nützliche Funktionen

Funktion	Bedeutung
array_diff($arr1, $arr2,...)	ermittelt Unterschiede in Datenfeldern und gibt diese als neues Datenfeld zurück
array_merge($arr1, $arr2)	verbindet zwei Datenfelder zu einem neuen Feld
array_pad($arr, $len, $wert)	verkürzt (bei $len<0) oder verlängert ein nummerisches Feld um $len Elemente und ersetzt leere Elemente
in_array($wert, $arr)	gibt TRUE zurück, wenn ein Wert in einem Datenfeld vorhanden ist
array_shift($arr)	liefert den Wert des ersten Elementes eines Datenfeldes und löscht das Element dann im Feld
array_pop($arr)	gibt den Wert des ersten Elementes eines Datenfeldes und löscht das Element dann im Feld
array_sum($arr)	summiert die Werte aller Ganz- und Fließkommazahlen aus einem Feld
array_unique($arr)	entfernt mehrfache Einträge aus einem Datenfeld