PHP-Befehlsübersicht

PHP
Hypertext
Preprocessor

Personal
Home
Page

Variablen

Einfache Beispiele

Notation	Beschreibung
$a = “String“;	String
$b = “4“;	String
$c = 4;	Integer
$d = 4.837;	Fließkommazahl

Stringsteuerzeichen

Notation	Beschreibung
“String“	Stringbegrenzer (Variablen im String werden ausgewertet)
‘String’	Stringbegrenzer (Variablen im String werden nicht ausgewertet)
$	Variablenzeichen (Variablen werden im “String“ ausgewertet, in ’String’ aber nicht!)
\	Maskierungszeichen (z.B. um Zeichen wie “ oder $ im String zu verwenden)
<<<EOQ String EOQ;	Hinter <<< wird eine beliebige Endemarke festgelegt, bei der der String endet. Der String kann beliebige Zeichen enthalten. Lediglich die Endemarke darf im String nicht vorkommen. ***Beispiel:*** $a = <<<Ende <Dies “ist“ /ein$Text. Ende; echo $a; → <Dies“ist“/ein$Text.

Typkonvertierung

Automatische Typkonvertierung

Notation	Beschreibung
$a = 2; $b =”4”; $c = $a + $b; →$c = 6	Ergebnis vom Typ Integer
$a = 2; $b =”4 Autos”; $c = $a + $b; →$c = 6	Ergebnis vom Typ Integer
$a = 2; $b = 4.123; $a = $a + $b; →$a = 6.123	Die Integervariable wird durch die Berechnung zur Fließkommazahl.

Explizite Typkonvertierung

Beispiel	Bedeutung
$a = 1; $b = (String) $a;	Umwandlung durch Typcastoperator (Datentyp in Klammern)
$a = 1; settype ($a, “string”);	Umwandlung durch settype-Funktion
intval(), doubleval(), stringval()	Wandelt Variable in gewünschten Datentyp um

Referenz

Beispiel	Bedeutung
$a = ’b’; $$a = ’c’ → $b = ‘c’	Variablen für Variablen (Referenzen) können angelegt werden mit $$

Array

Beispiel

Es soll ein eindimensionales Feld $a mit den Werten “Kreis“, “Rot“ und dem Integerwert 13 gefüllt werden.

Definition

Definition	Beschreibung
$a = array (“Kreis“, “Rot“, 13);	Array-Definition mit Indexnummerierung. Wird kein Index angegeben, wird am Ende des Arrays angefügt. (Unterschiedliche Datentypen im selben Array sind möglich.)
$a = array(); $a[0] = “Kreis”; $a[1] = “Rot”; $a[2] = 13;	
$a = array(); $a[] = “Kreis”; $a[] = “Rot”; $a[] =13;	
$a = array (“Form” => “Kreis”, “Farbe” => “Rot”, “Groesse“ => 13);	Der Index kann durch „Schlüssel“ ersetzt werden (assoziatives Array). Neue Elemente können diesem Array hinzugefügt werden durch: $a[“Groesse“] = 13;

Ausgabe

Definition	Ausgabe	Anmerkung
$a = array (“Kreis“, “Rot“);	for ($i = 0; $i<count($a), $i++) { echo ”$a[$i]”; }	Ausgabe eines indizierten Arrays

$a = array
(
“form” =>
“Kreis”,
“farbe” =>
“Rot”
);
	foreach ($a as $key
	=> $value)
	{
	echo “Schlüssel :
	$key, Wert: $value” ;
	}
		Schlüssel und Werte eines assoziativen Arrays werden ausgegeben

	while (list ($key, $value)
	= each($a))
	{
	echo “Schlüssel :
	$key, Wert: $value” ;
	}
		Schlüssel und Werte eines assoziativen Arrays werden ausgegeben

	while (list (, $value)
	= each($a))
	{
	echo “Wert: $value” ;
	}
		Nur die Werte eines assoziativen Arrays werden ausgegeben

	while (list ($key)
	= each($a))
	{
	echo “Schlüssel :
	$key ” ;
	}
		Nur die Schlüssel eines assoziativen Arrays werden ausgegeben

Mehrdimensionale Arrays

Definition	Ausgabe

$a = array
(
“kreis” => array
(
“farbe” = “Rot”,
“groesse” = 15
),
“quadrat” = array
(
“farbe” = “Blau”,
“groesse” = 14
)
);
	echo $a[“kreis“][“farbe“];
	→ „Rot“
	
	while (list ($figur, $figur_daten)
	= each($a))
	{
	while (list ($merkmal, $wert)
	= each($figur_daten))
	{
	echo “ Figur: $figur,
	Merkmal: $merkmal,
	Wert: $wert“;
	}
	}

Prüfen von Variablen

Beispiel	Bedeutung
isset()	true, wenn Variable gesetzt
empty()	true, wenn Variablenwert = 0 bei Integer true, wenn Variablenwert = “” bei String
is_int() is_long()	true, wenn Variable einen ganzzahlig Wert enthält
is_double() is_float()	true, wenn Variable eine Fließkommazahl enthält.
is_array()	true, wenn Variable ein Array ist.
is_bool()	true, wenn Variable boole'schen Wert enthält.
is_object()	true, wenn Variable ein Objekt ist
gettype()	liefert den Datentyp einer Variablen zurück.

Beispiel für die Verwendung der Funktion gettype()

Datentypbestimmung mit gettype()

```
$str = "Ich bin ein String";
$typ = gettype($str);
if ($typ == "string"){ echo "Die Variable
enthält einen String"};
```

Umgebungsvariablen

Vordefinierte Umgebungsvariablen können mit der Funktion phpinfo() abgefragt werden. Für die Verwendung in PHP gibt es einige globale Arrays, in denen die unterschiedlichsten Informationen abgelegt sind, beispielsweise liefert der Aufruf $_SERVER[PHP_SELF] den Pfad und den Dateinamen der aktiven Datei.

PHP-Variablen

Variable	Bedeutung
$_SERVER	Serverinformationen
HTTP_HOST HTTP_USER_AGENT HTTP_COOKIE REMOTE_ADDR DOCUMENT_ROOT PHP_SELF	IP-Adresse eigener Rechner Info aufrufender Browser Session-ID IP-Adr. aufrufender Rechner Öffentliches Webverzeichnis Aktives Dokument
$_POST	Übergabeparameter
$_COOKIE	Cookie-Informationen
$_REQUEST	Anfrageinfo
$_SESSION	Sitzungsvariablen

Operatoren

Vergleichsoperatoren

Operator	Bezeichnung	Bedeutung
==	gleich	gleicher Wert
===	identisch	gleicher Wert und Daten-Typ
!=	ungleich	ungleiche Werte
<, > <=, >=	kleiner, größer, kleiner oder gleich, größer oder gleich	

Logische Operatoren

Operator	Bezeich-nung	Bedeutung
and, &&	und	beide Bedingungen wahr (&& bindet stärker)
or, ||	oder	mindestens eine Bedingung wahr (|| bindet stärker)
xor	entweder oder	genau eine der Bedingungen wahr
!	nicht	logische Negation

Anweisungen

Elementare Anweisungen

Leere Anweisung	;	keine Auswirkung

Block	{
	Anweisung1;
	Anweisung2;
	...
	}
		Zusammenfassung von Anweisungen

Ausdruck	Ausdruck;	Ausdrucksanweisungen dienen dazu, Ausdrücke in einem Anweisungskontext auszuführen. Ausdrücke können sein: - Zuweisung - Inkrement und Dekrement - Methodenaufruf - Instanzerzeugung

Verzweigungen

if-Verzweigungen

Syntax	Bedeutung
if (Ausdruck) {Anweisung;}	Wenn der Ausdruck = true ist, dann führe die Anweisung aus.

if (Ausdruck)
{Anweisung1;}
else
{Anweisung2;}
	Wenn der Ausdruck = true ist, dann führe die Anweisung1 aus. Sonst führe die Anweisung2 aus.

(Ausdruck) ? Anweisung1: Anweisung2;	

if (Ausdruck1)
{Anweisung1;}
elseif (Ausdruck2)
{Anweisung2;}
else
{Anweisung3;}
	Wenn der Ausdruck1 = true ist, dann führe die Anweisung1 aus.
	Sonst prüfe den Ausdruck2.
	Wenn der Ausdruck2 = true ist, dann führe die Anweisung2 aus.
	Sonst führe die Anweisung3 aus.

switch-Verzweigung

Syntax	Bedeutung

switch (Ausdruck)
{
case Konstante1:
Anweisung1;
break;
case Konstante2:
Anweisung2;
break;
}
	Zunächst wird der Ausdruck (der vom Typ byte, short, char, int und string sein darf) ausgewertet. Ab dem entsprechenden case-Fall werden alle nachfolgenden case-Fälle ausgeführt, wenn die Ausführung nicht mit dem break-Befehl unterbrochen wird.

Schleifen

while-Schleife

Syntax	Bedeutung
while (Ausdruck) {Anweisung;}	Solange der Ausdruck = true ist, führe die Anweisung aus.

do-Schleife

Syntax	Bedeutung
do {Anweisung;} while (Ausdruck);	Führe zunächst die Anweisung aus. Wiederhole die Anweisung dann solange der Ausdruck = true ist.

for-Schleife

Syntax	Bedeutung
for (init, test, update) {Anweisung;}	Wiederhole update und die Anweisung beginnend bei init bis die Bedingung test erfüllt ist.

foreach-Schleife

Syntax	Bedeutung
foreach (array as Einzelwert) {Anweisung;}	Wiederhole die Anweisung für jedes Array-Element aus array. Dabei kann auf den jeweiligen Einzelwert zugegriffen werden.

break

Die break-Anweisung in einer Schleife bewirkt das Verlassen der Schleife. Das Programm wird mit der ersten Anweisung nach der Schleife fortgesetzt.

continue

Bei einer continue-Anweisung in einer Schleife springt das Programm an das Ende des Schleifenrumpfs und beginnt mit der nächsten Iteration.

Beispiele

Es soll das Array $a = array („Kreis“, „Rot“, 13) mit Hilfe von Schleifen ausgegeben werden.

while Schleife

```
$i = 0;
while ($i <= 2)
{
   echo "Ausgabe: $a[i]"; i++;
}
```

do-while-Schleife

```
$i = 0;
do
{
   echo "Ausgabe: $a[i]"; $i++;
} while ($i <= 2)
```

for-Schleife

```
for ($i = 0; $i <= 2; $i++)
{                    {
   echo "Ausgabe: $i";
}
```

foreach-Schleife

```
foreach ($a as $i)
{
   echo "Ausgabe: $a[i]";
}
```

Beispiele

Es soll das Array $a = array(“form” => “Kreis”, “farbe” => “Rot”) mit Hilfe von Schleifen ausgegeben werden.

while-Schleife

```
while (list($schluessel,$wert)=each($a))
{
   echo "Ausgabe: $schluessel $wert";
}
```

foreach-Schleife

```
foreach ($a as $schluessel => $wert)
{
   echo "Ausgabe: $schluessel $wert";
}
```

MySQL-Funktionen

Funktion / und deren Bedeutung
int mysql_connect (str[1] host, str benutzer, str passwort) Baut Verbindung zum DB-Server auf
int mysql_select_db(str db_name[, int verbindungsk.]) Auswählen der Datenbank
int mysql_query (string anfrage [, int verbindungsk.]) Sendet eine SQL-Anfrage an einen Datenbankserver. Wird der optionale Parameter der Verbindungskennung nicht angegeben, so wird versucht, eine Verbindung ohne Angaben von Argumenten (siehe mysql_connect) aufzubauen.
array mysql_fetch_array (int ergebnisk. [, int erg.typ]) Liefert anhand einer Ergebniskennung Datensätze in einem assoziativen Array zurück. Dabei werden die Feldnamen innerhalb der Tabelle als Schlüssel des Arrays genutzt. Im Erfolgsfall liefert diese Funktion den aktuellen Datensatz, sonst wird false zurückgegeben. Der zweite Parameter ist optional und kann folgende Konstanten als Wert enthalten: MYSQL_ASSOC: Ergebnis ist ein assoziatives Array. MYSQL_NUM: Ergebnis ist ein numerisch indiziertes Array. MYSQL_BOTH: Ergebnis ist ein Array, das die Elemente des Ergebnisdatensatzes sowohl assoziativ als auch numerisch indiziert enthält. Dies ist der Default-Wert.
array mysql_fetch_row (int ergebniskennung) Wie mysql_fetch_array, aber das Ergebnis ist immer ein numerisches Array.
int mysql_insert_id ([int verbindungskennung]) Liefert anhand einer Verbindungskennung die Kennung des Datensatzes zurück, der bei einer vorangegangenen INSERT-Operation angelegt wurde.
int mysql_num_rows (int ergebnis) Anzahl der Zeilen, die durch eine Abfrage zurückgeliefert wurden.
int mysql_affected_rows ([int link_identifier]) Anzahl der Zeilen, die durch eine Abfrage (auch INSERT, DELETE, ...) betroffen sind.
int mysql_close ([int verbindungskennung]) Schließt eine Verbindung zu einer MySQL-Datenbank. Kann in der Regel weggelassen werden, da Verbindungen automatisch beendet werden.
int mysql_errno ([int verbindungskennung]) Liefert die Fehlernummer einer zuvor ausgeführten Operation zurück.
string mysql_error ([int verbindungskennung]) Liefert den Fehlertext einer zuvor ausgeführten Operation zurück.

Mit dem Zusatz or die können Anweisungen den Funktionen hinzugefügt werden, die im Fehlerfall ausgeführt werden sollen.

Beispiel

Es werden die Werte der Tabelle 'Eintrag' der Datenbank 'Gaestebuch' ausgegeben.

PHP-Skriptdatei
Dateiname: 'anzeige.php'

```
<html>
  <body bgcolor='#cccccc'>
     <h1> Gästebuch </h1>
     <hr>
     <table border='1'>
<?php
//Beginn des php-Skriptes
//Variablendefinition
  $dbserver   = "localhost";
  $dbuser     = "gb_user";
  $dbpasswort = "12345";
  $dbname     = "Gaestebuch";
//Verbindung zum DB-Server aufbauen
  $dbh = mysql_connect($dbserver,$dbuser,
  $dbpasswort)
     or die ("Fehler bei CONNECT");
//Verbindung zur Datenbank aufbauen
  mysql_select_db ($dbname, $dbh)
     or die ("Fehler bei SELECT_DB");
//SQL-Abfrage an die Datenbank senden
  $sql = "SELECT * FROM Eintrag";
  $result = mysql_query ($sql, $dbh)
     or die ("Fehler bei QUERY");
//Ergebnis der SQL-Abfrage verarbeiten
  while ($row=mysql_fetch_row($result))
  {
     echo "<tr>\n";
     foreach ($row as $i)
     {
        echo "<td>$i</td>\n";
     }
     echo "</tr>\n";
  }
//Datenbankverbindung schließen
  mysql_close($dbh);
// Ende des PHP-Skriptes
?>
     </table>
  </body>
</html>
```

Beispiel

Alternative Ergebnisanzeige ohne HTML-Tabelle (vergleichen Sie mit dem rot hinterlegten Bereich im obigen Beispiel).

Anzeige der Ergebnismenge $result einer SQL-Abfrage mit Hilfe der Funktion mysql_fetch_array.

```
//Ergebnis der SQL-Abfrage verarbeiten
while (list($n,$e,$t)
          = mysql_fetch_array($result))
{
  echo "Name: $n, E-Mail: $e, Text: $t";
}
```

1 str steht für String (Zeichenkette)

Strukturiertes Programmieren

include / require

Funktion	Bedeutung
include_once()	bindet noch nicht eingebundene Dateien wenn notwendig ein
require_once()	bindet noch nicht eingebundene Dateien auch wenn nicht notwendig ein
include()	bindet Dateien ein (nur wenn notwendig)
require()	bindet Datei immer ein

include_once

```
//Datei a.php
<?php
   ...
?>
```

```
//Datei b.php
<?php
  include_once("a.php");
   …
?>
```

```
//Datei c.php
<?php
  include_once("a.php");
  include_once("b.php");
   …
?>
```

```
//Datei Start.php
<?php
  include_once("a.php");
  include_once("b.php");

  $a = 0;
  if ($a == 1)
  {
    include_once ("c.php");
  }
?>
```

Bereits eingebundene Dateien werden nicht noch mal eingebunden (hier durchgestrichen).

Ein include-once-Befehl in einer nicht erfüllten Bedingung wird nicht ausgeführt.

Die Datei **a.php** wird **1x** eingebunden.

require_once

```
//Datei a.php
<?php
   ...
?>
```

```
//Datei b.php
<?php
  require_once("a.php");
   …
?>
```

```
//Datei c.php
<?php
  require_once ("a.php");
  require_once ("b.php");
   …
?>
```

```
//Datei Start.php
<?php
  require_once ("a.php");
  require_once ("b.php");

  $a = 0;
  if ($a == 1)
  {
    require_once ("c.php");
  }
?>
```

Bereits eingebundene Dateien werden nicht noch mal eingebunden (hier durchgestrichen)

Ein require-once-Befehl wird in jedem Fall ausgeführt, selbst wenn er sich in einer nicht erfüllten Bedingung befindet.

Die Datei **a.php** wird **1x** eingebunden.

include

```
//Datei a.php
<?php
   ...
?>
```

```
//Datei b.php
<?php
    include("a.php");
   …
?>
```

```
//Datei c.php
<?php
    include("a.php");
    include("b.php");
   …
?>
```

```
//Datei Start.php
<?php
    include("a.php");
    include("b.php");

  $a = 0;
  if ($a == 1)
  {
    include ("c.php");
  }
?>
```

Dateien werden selbst dann eingebunden, wenn sie bereits eingebunden worden sind.

Ein include- Befehl in einer nicht erfüllten Bedingung wird nicht ausgeführt.

Die Datei **a.php** wird **2x** eingebunden.

require

```
//Datei a.php
<?php
   ...
?>
```

```
//Datei b.php
<?php
    require ("a.php");
   …
?>
```

```
//Datei c.php
<?php
    require ("a.php");
    require ("b.php");
   …
?>
```

```
//Datei Start.php
<?php
   require ("a.php");
   require ("b.php");

  $a = 0;
  if ($a == 1)
  {
    require ("c.php");
  }
?>
```

Dateien werden selbst dann eingebunden, wenn sie bereits eingebunden worden sind.

Ein require- Befehl wird in jedem Fall ausgeführt, selbst wenn er sich in einer nicht erfüllten Bedingung befindet.

Die Datei **a.php** wird **4x** eingebunden.

Funktionen

Funktion	Aufruf /Ausgabe	Beschreibung

function ausgabe()
{
echo “Ich gebe was aus“;
}
	ausgabe();
	→ Ich gebe was aus
		parameterlose Funktion
		ohne Rückgabewert.

function berechne ($a, $b)
{
$c = $a * $b;
echo “Ergebnis: $c”;
}
	berechne (2, 3);
	→ Ergebnis: 6
		Funktion mit 2 Parametern
		ohne Rückgabewert.

function berechne ($a, $b)
{
$c = $a * $b;
return $c;
}
	$d = berechne (2, 3);
	echo “Ergebnis: $d”;
	→ Ergebnis: 6
	echo “Ergebnis: ”, berechne (2, 3);
	→ Ergebnis: 6
		Funktion mit 2 Parametern und
		mit einem Rückgabewert.

function berechne ($a = 1, $b = 2)
{
$c = $a * $b;
return $c;
}
	$d = berechne ()
	echo “Ergebnis: $d”;
	→ Ergebnis: 2
	$d = berechne (3)
	echo “Ergebnis: $d”;
	→ Ergebnis: 6
	$d = berechne (3, 4)
	echo “Ergebnis: $d”;
	→ Ergebnis: 12
		Funktion mit 2 optionalen Parametern und
		mit einem Rückgabewert.
		
		Die optionalen Parameter müssen von links nach rechts „gefüllt“ werden. So ist es in dem Beispiel nicht möglich nur den Parameter $b anzugeben, ohne auch den Parameter $a zu bestimmen.

function berechne ()
{
$args = func_get_args();
 	
$erg = 0;
foreach ($args as $i)
{
$erg = $erg + $i;
}
return $erg;
}
	$d = berechne (1)
	echo “Ergebnis: $d”;
	→ Ergebnis: 1
	$d = berechne (1, 5, 7)
	echo “Ergebnis: $d”;
	→ Ergebnis: 13
		Funktion mit beliebig vielen Parametern und
		mit einem Rückgabewert.
		
		func_get_args();
		liest Argumente in ein Array
		func_num_args();
		liefert die Anzahl der Parameter

Grundsätzlich kann in PHP nur ein Wert mit return aus einer Funktion zurückgegeben werden. Da dieser Rückgabewert auch ein ganzes Array sei kann, ist somit auch die Rückgabe unterschiedlichster Werte aus einer Funktion möglich.

Anmerkung

Diese PHP-Befehlsübersicht ist keinesfalls vollständig! PHP bietet für diverse Anforderungen und Problemstellungen unterschiedlichste vordefinierte Funktionen, die hier nicht alle vorgestellt und erörtert werden können. Deshalb ist folgender Hinweis sehr nützlich.

Bevor man eigene Funktionen schreibt IMMER erst nachsehen, ob nicht bereits eine entsprechende PHP-Funktion existiert.

PHP im Internet

PHP-Befehlsreferenz

Eine sehr gute PHP-Befehlsreferenz (auch in deutscher Sprache) findet sich im Internet unter:

`http://www.php.net/`

Hinweis: Im Bereich Dokumentation kann das deutschsprachige Handbuch ausgewählt werden.