

PHP-Befehlsübersicht

PHP
Hypertext
Preprocessor



Personal
Home
Page

Variablen

Einfache Beispiele

Notation	Beschreibung
<code>\$a = "String";</code>	String
<code>\$b = "4";</code>	String
<code>\$c = 4;</code>	Integer
<code>\$d = 4.837;</code>	Fließkommazahl

Stringsteuerzeichen

Notation	Beschreibung
<code>"String"</code>	Stringbegrenzer (Variablen im String werden ausgewertet)
<code>'String'</code>	Stringbegrenzer (Variablen im String werden nicht ausgewertet)
<code>\$</code>	Variablenzeichen (Variablen werden im "String" ausgewertet, in 'String' aber nicht!)
<code>\</code>	Maskierungszeichen (z.B. um Zeichen wie " oder \$ im String zu verwenden)
<code><<<EOQ String EOQ;</code>	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: <code>\$a = <<<Ende <Dies "ist" /ein\$Text. Ende; echo \$a; -> <Dies"ist"/ein\$Text.</code>

Typkonvertierung

Automatische Typkonvertierung

Notation	Beschreibung
<code>\$a = 2; \$b = "4"; \$c = \$a + \$b; -> \$c = 6</code>	Ergebnis vom Typ Integer
<code>\$a = 2; \$b = "4 Autos"; \$c = \$a + \$b; -> \$c = 6</code>	Ergebnis vom Typ Integer
<code>\$a = 2; \$b = 4.123; \$a = \$a + \$b; -> \$a = 6.123</code>	Die Integervariable wird durch die Berechnung zur Fließkommazahl.

Explizite Typkonvertierung

Beispiel	Bedeutung
<code>\$a = 1; \$b = (String) \$a;</code>	Umwandlung durch Typcastoperator (Datentyp in Klammern)
<code>\$a = 1; settype (\$a, "string");</code>	Umwandlung durch settype -Funktion
<code>intval(), doubleval(), stringval()</code>	Wandelt Variable in gewünschten Datentyp um

Referenz

Beispiel	Bedeutung
<code>\$a = 'b'; \$\$a = 'c' -> \$b = 'c'</code>	Variablen für Variablen (Referenzen) können angelegt werden mit <code>\$\$</code>

Array

Beispiel

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

Definition

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

Ausgabe

Definition	Ausgabe	Anmerkung
<code>\$a = array ("Kreis", "Rot");</code>	<code>for (\$i = 0; \$i < count(\$a), \$i++) { echo "\$a[\$i]"; }</code>	Ausgabe eines indizierten Arrays
<code>\$a = array ("form" => "Kreis", "farbe" => "Rot");</code>	<code>foreach (\$a as \$key => \$value) { echo "Schlüssel : \$key, Wert: \$value" ; }</code>	Schlüssel und Werte eines assoziativen Arrays werden ausgegeben
	<code>while (list (\$key, \$value) = each(\$a)) { echo "Schlüssel : \$key, Wert: \$value" ; }</code>	Schlüssel und Werte eines assoziativen Arrays werden ausgegeben
	<code>while (list (, \$value) = each(\$a)) { echo "Wert: \$value" ; }</code>	Nur die Werte eines assoziativen Arrays werden ausgegeben
	<code>while (list (\$key) = each(\$a)) { echo "Schlüssel : \$key" ; }</code>	Nur die Schlüssel eines assoziativen Arrays werden ausgegeben

Mehrdimensionale Arrays

Definition	Ausgabe
<pre>\$a = array ("kreis" => array ("farbe" = "Rot", "groesse" = 15), "quadrat" = array ("farbe" = "Blau", "groesse" = 14));</pre>	<pre>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"; } }</pre>

Prüfen von Variablen

Beispiel	Bedeutung
<code>isset()</code>	true, wenn Variable gesetzt
<code>empty()</code>	true, wenn Variablenwert = 0 bei Integer true, wenn Variablenwert = "" bei String
<code>is_int()</code> <code>is_long()</code>	true, wenn Variable einen ganzzahlig Wert enthält
<code>is_double()</code> <code>is_float()</code>	true, wenn Variable eine Fließkommazahl enthält.
<code>is_array()</code>	true, wenn Variable ein Array ist.
<code>is_bool()</code>	true, wenn Variable boole'schen Wert enthält.
<code>is_object()</code>	true, wenn Variable ein Objekt ist
<code>gettype()</code>	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
<code>\$_SERVER</code>	Serverinformationen
<code>HTTP_HOST</code> <code>HTTP_USER_AGENT</code> <code>HTTP_COOKIE</code> <code>REMOTE_ADDR</code> <code>DOCUMENT_ROOT</code> <code>PHP_SELF</code>	IP-Adresse eigener Rechner Info aufrufender Browser Session-ID IP-Adr. aufrufender Rechner Öffentliches Webverzeichnis Aktives Dokument
<code>\$_POST</code>	Übergabeparameter
<code>\$_COOKIE</code>	Cookie-Informationen
<code>\$_REQUEST</code>	Anfrageinfo
<code>\$_SESSION</code>	Sitzungsvariablen

Operatoren

Vergleichsoperatoren

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

Logische Operatoren

Operator	Bezeichnung	Bedeutung
<code>and</code> , <code>&&</code>	und	beide Bedingungen wahr (<code>&&</code> bindet stärker)
<code>or</code> , <code> </code>	oder	mindestens eine Bedingung wahr (<code> </code> bindet stärker)
<code>xor</code>	entweder oder	genau eine der Bedingungen wahr
<code>!</code>	nicht	logische Negation

Anweisungen

Elementare Anweisungen

Leere Anweisung	<code>;</code>	keine Auswirkung
Block	<code>{</code> <code>Anweisung1;</code> <code>Anweisung2;</code> <code>...</code> <code>}</code>	Zusammenfassung von Anweisungen
Ausdruck	<code>Ausdruck;</code>	Ausdrucksanweisungen dienen dazu, Ausdrücke in einem Anweisungskontext auszuführen. Ausdrücke können sein: - Zuweisung - Inkrement und Dekrement - Methodenaufruf - Instanzerzeugung

Verzweigungen

if-Verzweigungen

Syntax	Bedeutung
<code>if (Ausdruck) {Anweisung;}</code>	Wenn der Ausdruck = true ist, dann führe die Anweisung aus.
<code>if (Ausdruck) {Anweisung1;} else {Anweisung2;}</code>	Wenn der Ausdruck = true ist, dann führe die Anweisung1 aus. Sonst führe die Anweisung2 aus.
<code>(Ausdruck) ? Anweisung1: Anweisung2;</code>	
<code>if (Ausdruck1) {Anweisung1;} elseif (Ausdruck2) {Anweisung2;} else {Anweisung3;}</code>	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
<code>switch (Ausdruck) { case Konstante1: Anweisung1; break; case Konstante2: Anweisung2; break; }</code>	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
<code>while (Ausdruck) {Anweisung;}</code>	Solange der Ausdruck = true ist, führe die Anweisung aus.

do-Schleife

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

for-Schleife

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

foreach-Schleife

Syntax	Bedeutung
<code>foreach (array as Einzelwert) {Anweisung;}</code>	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
<pre>\$i = 0; while (\$i <= 2) { echo "Ausgabe: \$a[\$i]"; i++; }</pre>
do-while-Schleife
<pre>\$i = 0; do { echo "Ausgabe: \$a[\$i]"; \$i++; } while (\$i <= 2)</pre>
for-Schleife
<pre>for (\$i = 0; \$i <= 2; \$i++) { echo "Ausgabe: \$i"; }</pre>
foreach-Schleife
<pre>foreach (\$a as \$i) { echo "Ausgabe: \$a[\$i]"; }</pre>

Beispiele

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

while-Schleife
<pre>while (list(\$schluessel, \$wert)=each(\$a)) { echo "Ausgabe: \$schluessel \$wert"; }</pre>
foreach-Schleife
<pre>foreach (\$a as \$schluessel => \$wert) { echo "Ausgabe: \$schluessel \$wert"; }</pre>

MySQL-Funktionen

Funktion / und deren Bedeutung	
int mysqli_connect (string host, string benutzer, string passwort)	Baut Verbindung zum DB-Server auf
int mysqli_select_db (int dbh, str db_name)	Auswählen der Datenbank
int mysqli_query (int dbh, string sql)	Sendet eine SQL-Anfrage (sql) an einen Datenbankserver (Verbindungskennung dbh). Wird der optionale Parameter der Verbindungskennung nicht angegeben, so wird versucht, eine Verbindung ohne Angaben von Argumenten aufzubauen.
array mysqli_fetch_array (int result [, int typ])	Liefert anhand einer Ergebniskennung (result) 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 (typ) 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 mysqli_fetch_row (int result)	Wie mysqli_fetch_array , aber das Ergebnis ist immer ein numerisches Array.
int mysqli_insert_id ([int dbh])	Liefert anhand einer Verbindungskennung (dbh) die ID des Datensatzes zurück, der bei einer vorangegangenen INSERT -Operation angelegt wurde.
int mysqli_num_rows (int result)	Anzahl der Zeilen einer Ergebnismenge (result), die durch eine Abfrage zurückgeliefert wurden.
int mysqli_affected_rows ([int dbh])	Liefert anhand einer Verbindungskennung (dbh) die Anzahl der Zeilen, die durch eine Abfrage (auch INSERT , DELETE , ...) betroffen sind.
int mysqli_close ([int dbh])	Schließt eine Verbindung (dbh) zu einer MySQL-Datenbank. Kann in der Regel weggelassen werden, da Verbindungen automatisch beendet werden.
int mysqli_errno (int dbh)	Liefert die Fehlernummer einer zuvor ausgeführten Operation zurück.
string mysqli_error (int dbh)	Liefert den Fehlertext einer zuvor ausgeführten Operation zurück.

Mit **or die** können Anweisungen den Funktionen hinzugefügt werden, die im Fehlerfall ausgeführt werden sollen. Die **mysqli** Befehle sind eigentlich Objektorientiert konzipiert. Da in diesem Buch (aus Platzgründen) nicht weiter auf die Objektorientierung in PHP eingegangen werden kann, werden hier lediglich die Aliasbefehle verwendet.

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";
  $dbpassword = "12345";
  $dbname = "Gaestebuch";
  //Verbindung zum DB-Server aufbauen
  $dbh = mysqli_connect($dbserver,
  $dbuser, $dbpassword)
  or die ("Fehler bei CONNECT");
  //Verbindung zur Datenbank aufbauen
  mysqli_select_db ($dbh, $dbname)
  or die ("Fehler bei SELECT_DB");
  //SQL-Abfrage an die Datenbank senden
  $sql = "SELECT * FROM Eintrag";
  $result = mysqli_query ($dbh, $sql)
  or die ("Fehler bei QUERY");
  //Ergebnis der SQL-Abfrage verarbeiten
  while ($row=mysqli_fetch_row($result))
  {
    echo "<tr>\n";
    foreach ($row as $i)
    {
      echo "<td>$i</td>\n";
    }
    echo "</tr>\n";
  }
  //Datenbankverbindung schließen
  mysqli_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 **mysqli_fetch_array**.

```

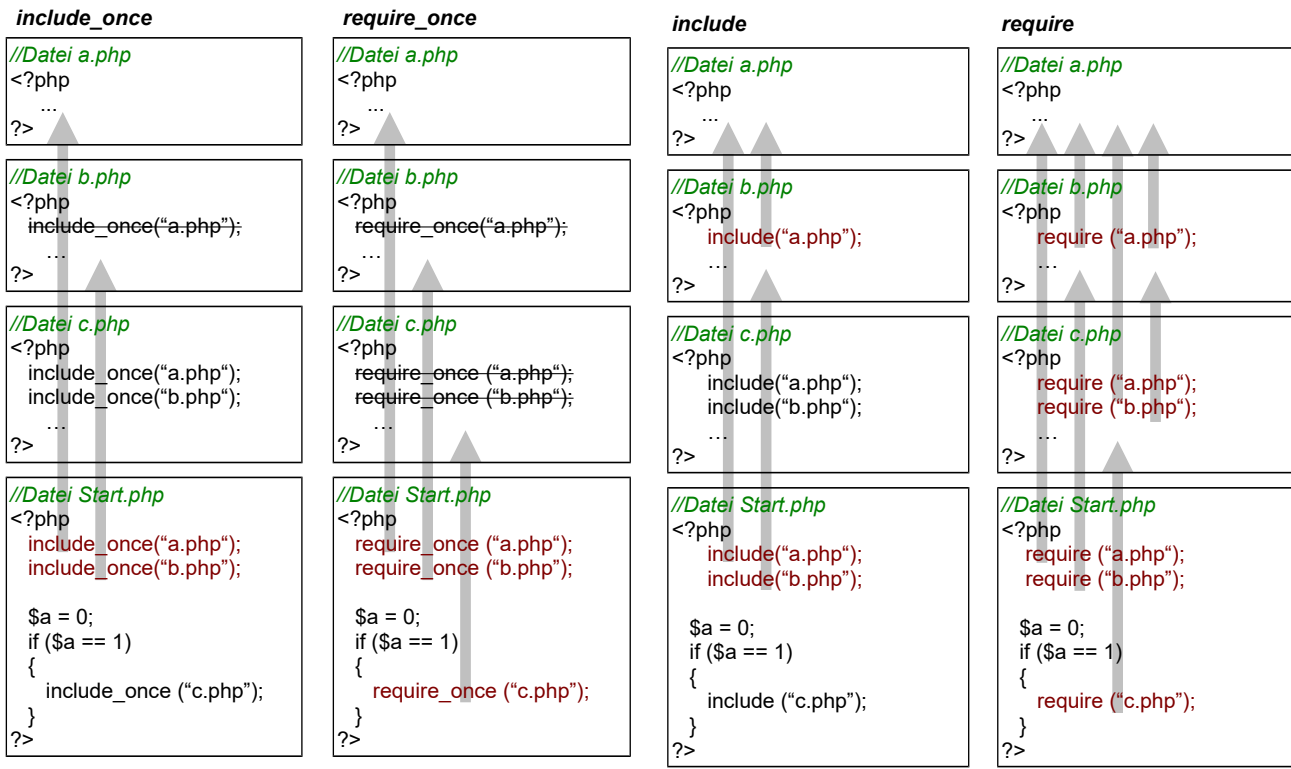
//Ergebnis der SQL-Abfrage verarbeiten
while (list($n,$e,$t)
= mysqli_fetch_array($result))
{
  echo "Name: $n, E-Mail: $e, Text: $t";
}

```

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



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.

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.

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.

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
<pre>function ausgabe() { echo "Ich gebe was aus"; }</pre>	<pre>ausgabe(); → Ich gebe was aus</pre>	parameterlose Funktion ohne Rückgabewert.
<pre>function berechne (\$a, \$b) { \$c = \$a * \$b; echo "Ergebnis: \$c"; }</pre>	<pre>berechne (2, 3); → Ergebnis: 6</pre>	Funktion mit 2 Parametern ohne Rückgabewert.
<pre>function berechne (\$a, \$b) { \$c = \$a * \$b; return \$c; }</pre>	<pre>\$d = berechne (2, 3); echo "Ergebnis: \$d"; → Ergebnis: 6 echo "Ergebnis: ", berechne (2, 3); → Ergebnis: 6</pre>	Funktion mit 2 Parametern und mit einem Rückgabewert.
<pre>function berechne (\$a = 1, \$b = 2) { \$c = \$a * \$b; return \$c; }</pre>	<pre>\$d = berechne () echo "Ergebnis: \$d"; → Ergebnis: 2</pre>	Funktion mit 2 optionalen Parametern und mit einem Rückgabewert.
	<pre>\$d = berechne (3) echo "Ergebnis: \$d"; → Ergebnis: 6</pre>	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.
	<pre>\$d = berechne (3, 4) echo "Ergebnis: \$d"; → Ergebnis: 12</pre>	
<pre>function berechne () { \$args = func_get_args(); \$serg = 0; foreach (\$args as \$i) { \$serg = \$serg + \$i; } return \$serg; }</pre>	<pre>\$d = berechne (1) echo "Ergebnis: \$d"; → Ergebnis: 1 \$d = berechne (1, 5, 7) echo "Ergebnis: \$d"; → Ergebnis: 13</pre>	<p>Funktion mit beliebig vielen Parametern und mit einem Rückgabewert.</p> <p><code>func_get_args();</code> liest Argumente in ein Array <code>func_num_args();</code> liefert die Anzahl der Parameter</p>

Grundsätzlich kann in PHP nur ein Wert mit `return` aus einer Funktion zurückgegeben werden. Da dieser Rückgabewert auch ein ganzes Array sein 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 findet sich im Internet unter:

- <https://secure.php.net/manual/de/>
- <http://www.php.net/manual/de/>

