# L1\_3.3 Variablen in PHP

Variablen dienen dazu, Daten während der Abarbeitung eines Skriptes zu speichern. Wenn Sie eine Variable deklarieren und an einen Wert binden, ist dies nichts anders, als dem Wert einen Namen zu geben. Wenn Sie genau diesen Wert zu einem späteren Zeitpunkt in ihrem Skript wieder benötigen, können Sie ihn über den vergebenen Namen ansprechen. Der Wert einer Variablen ist veränderlich. Wenn Sie Daten einer Variablen zuweisen, die bereits besteht, wird der bisherige Wert von dem neuen überschrieben.

Sie kennen Variablen und den Umgang mit Variablen bereits aus anderen Programmiersprachen. Ein großer Unterschied von PHP-Variablen zu Variablen in anderen Programmiersprachen ist die Tatsache, dass eine Variable immer mit einem Dollarzeichen ($) beginnt! Anschließend muss ein Buchstabe oder ein Unterstich folgen. Bei allen weiteren Zeichen können Sie zwischen Buchstaben, Ziffern und dem Unterstrich auswählen. Bei Variablennamen unterscheidet PHP zwischen Groß- und Kleinschreibung.

**!!!**

**!**

**!**

**$ergebnis**

Variablen werden bei der ersten Wertzuweisung (Initialisierung) implizit einem Typen zugewiesen. Der Typ legt die Operatoren fest, mit denen Sie die Daten bearbeiten können. Zum Beispiel lässt sich die Zeichenkette „Das ist ein ganz normaler Satz“ schwerlich zu der Zahl 10 addieren, weil sich mit Sätzen im Allgemeinen nicht rechnen lässt.

**Wichtige Datentypen:**

|  |  |  |
| --- | --- | --- |
| **Bezeichnung** | **Datentyp** | **Beispiele** |
| Zeichenkette | String | 'a' 'ab c' "a" "ab c" "123" |
| Ganzzahliger Wert | Integer | 1 532 -94 |
| Fließkommazahl | Float/Double | 1.5 3.14 -482.003 |
| boolescher Wert | Boolean | true false |
| Arrays | Array | mehrwertiger Datentyp |

PHP ist eine schwach getypte Sprache und nimmt Ihnen die Vergabe und Konvertierung von Datentypen weitgehend ab. Variablen können ihren Datentyp im Verlauf eines Skriptes mehrfach ändern, je nachdem, in welchem Kontext sie verarbeitet werden. Dies führt teilweisen zu verwirrenden Ergebnissen:

echo 2 + '5 Autos'

Die Ausgabe der Anweisung ist 7. Der PHP-Interpreter erkennt, dass es sich bei der Operation um eine Addition handelt und beide Operanden folglich eine Zahl sein müssen. Der linke Operand ist bereits eine Zahl, der rechte wird implizit in eine Zahl umgewandelt. Für die Typenkonvertierung gibt es in PHP mehrere Regeln. Zeichenketten werden beispielsweise in Zahlen überführt, indem von links nach rechts so viele Zeichen verwendet werden, wie sich als Zahl interpretieren lassen.

**Zuweisungsoperator (=)**

Zuweisungen erfolgen über den Zuweisungsoperator (=)

Beispiele:

$zahl1 **=** 10;

$zahl\_2 **=** 10;

$name **=** 'Max';

**Zeichenketten (String)**

Zeichenketten stehen immer in einfachen oder doppelten Anführungszeichen und können beliebige Zeichen enthalten. Der Ausdruck '1234' stellt eine Zeichenkette dar, weil er in Hochkommata eingeschlossen ist. Innerhalb von Zeichenketten kann nicht gerechnet werden!

1. Einfache Anführungsstriche  
   Eine Zeichenkette, die in einfache Anführungsstriche eingeschlossen ist, wird fast ohne Verarbeitung ausgegeben. Das bedeutet, darin enthaltene Variablen oder Sonderzeichen werden nicht ausgewertet. Die Anweisungen  
     
   $a = 'Test';  
   echo 'Ausgabe einer $a-Variablen‘;  
     
   führt nicht dazu, dass der Satz „Ausgabe einer Test-Variablen“ auf dem Bildschirm erscheint. Stattdessen ist „Ausgabe einer $a-Variablen“ zu lesen.
2. Doppelte Anführungsstriche  
   Zeichenketten in doppelten Anführungsstrichen werden vom PHP-Interpreter bei ihrer Verwendung verarbeitet. Es werden beispielsweise darin enthaltene Variablen aufgelöst. Die Anweisungen  
     
   $a = 'Test';  
   echo "Ausgabe einer $a-Variablen";  
     
   führt dazu, dass der Satz „Ausgabe einer Test-Variablen“ lautet. Doppelte Anführungszeichen sind damit weitaus flexibler. Möchten Sie Sonderzeichen in einem String ausgeben, ohne dass sie interpretiert werden, müssen Sie diese mit einem Backslash „maskieren“.  
   echo "Ausgabe von \$b"; 🡪 „Ausgabe von $b“

**Punkt-Operator (.)**

Der Punktoperator ermöglicht das Zusammenfügen mehrerer Komponenten zu einer Zeichenkette. Dabei werden Komponenten, die nicht vom Typ String sind, implizit in eine Zeichenkette verwandelt. Die Anweisungen

$a = 'Test';  
echo "Ausgabe einer " . $a . '-Variablen';

führen zum gleichen Ergebnis wie oben: „Ausgabe einer Test-Variablen“

**Ganzzahlige Werte (Integer)**

Ganzzahlige Werte sind Zahlen ohne Nachkommastellen. Dazu gehören die positiven und negativen Zahlen sowie die Null.

**Fließkommazahlen (Float, Double)**

Bei Berechnungen mit Integer-Zahlen verlassen Sie schnell den Bereich ganzzahliger Werte. Das Ergebnis einer Division 3 / 2 lässt sich nicht als Integer abbilden. Zahlen mit Nachkommastellen heißen in PHP-Fließkommazahlen. Achtung: Das Trennzeichen in Fließkommazahlen ist der Punkt, nicht das für uns vertraute Komma!

**Boolescher Wahrheitswert (Boolean)**

Bei booleschen Wahrheitswerten gibt es nur zwei unterschiedliche Ausprägungen: wahr und falsch, in PHP werden sie mit den Schlüsselwörtern true und false gekennzeichnet. Anwendung finden sie vor allem bei der Überprüfung der Gleichheit zweier oder mehrerer Werte.

**Vergleichsoperatoren**

Mittels Vergleichsoperatoren lassen sich Variablen als auch Werte in PHP vergleichen. Das Ergebnis des Vergleichs ist stets entweder true oder false. Das Ergebnis des Vergleichs lässt sich entweder in einer Variablen abspeichern oder direkt in der Bedingung von Kontrollstrukturen nutzen.

|  |  |  |
| --- | --- | --- |
| $a == $b | Gleich | Dieser Vergleich ist erfüllt, falls $a und $b denselben Wert beinhaltet. Sind die Typen der Variablen verschieden, so werden die konvertiert. |
| $a === $b | Identisch | Dieser Vergleich ist erfüllt, falls $a und $b denselben Typ und den Inhalt besitzen. Wäre ein Wert vom Typ Integer und der andere vom Typ String, so würde false zurückgegeben werden. |
| $a != $b | Ungleich | Dieser Vergleich ist erfüllt, falls $a und $b nicht denselben Wert beinhaltet. Sind die Typen der Variablen verschieden, so werden die konvertiert. |
| $a !== $b | Nicht identisch | Dieser Vergleich ist erfüllt, falls $a und $b einen unterschiedlichen Typ haben oder einen unterschiedlichen Wert. |
| $a < $b | Kleiner | $a muss kleiner als $b sein. |
| $a <= $b | Kleiner gleich | $a muss kleiner oder gleich $b sein. |
| $a > $b | Größer | $a muss größer als $b sein. |
| $a >= $b | Größer gleich | $a muss größer oder gleich $b sein. |