1

VARIABLEN, AUSDRÜCKE & OPERATOREN

Dieses Kapitel zeigt, wie Daten, die sich bei jeder Anforderung einer PHP-Seite ändern können, in Variablen gespeichert werden und wie Ausdrücke und Operatoren mit Variablenwerten umgehen.

Variablen stellen mit einem Namen einen Wert dar, der sich bei jedem Aufruf einer PHP-Seite ändern kann:

- Der **Name** beschreibt die Art der Daten, die die Variable enthält.
- Der **Wert** ist der Wert, den die Variable beim Aufruf der Seite enthalten soll.

Sobald die Seite fertig abgearbeitet und der HTML-Code an den Browser zurückgesendet wurde, vergisst der PHP-Interpreter die Variable wieder (und sie kann dann beim nächsten Aufruf der Seite einen anderen Wert enthalten).

PHP unterscheidet zwischen verschiedenen Arten von Werten, die Sie in einer Variablen speichern können (z. B. Text und Zahlen); diese werden als **Datentypen** bezeichnet:

- Ein Text wird als **Zeichenkette (String)** bezeichnet.
- Eine ganze Zahl heißt **Integer.**
- Ein Dezimalbruch entspricht dem Datentyp **Float.**
- Ein Wert, der wahr oder falsch (`true` oder `false`) sein kann, ist ein **boolescher Wert (Boolean)**.
- Eine Reihe zusammenhängender Namen und Werte lässt sich in einem **Array** speichern.

Wenn Sie sich erst einmal mit Variablen vertraut gemacht haben, werden Sie sehen, dass Ausdrücke aus mehreren Werten einen einzigen Wert bilden können. So lässt sich z. B. Text aus zwei Variablen zu einem Satz zusammenfügen oder eine in einer Variablen gespeicherte Zahl mit einer Zahl aus einer anderen Variablen multiplizieren.

Ausdrücke verwenden **Operatoren**, um einen einzelnen Wert zu erzeugen. Der Operator + wird beispielsweise zur Addition zweier Werte verwendet, der Operator – zur Subtraktion eines Werts von einem anderen.


Safari File Edit View History Bookmarks Develop Window Help
Thu 6 Jun 9:02 am
phpandmysql.com
The Candy Store
The Candy Store
MULTI-BUY OFFER
Hello, Ivy.
Buy 5 packs of Chocolate for $20
(usual price $25)
Save $5

VARIABLEN

Variablen speichern Daten, die sich bei jedem Aufruf einer PHP-Seite verändern (variieren) können. Sie stellen also einen veränderlichen Wert mit einem feststehenden Namen dar.

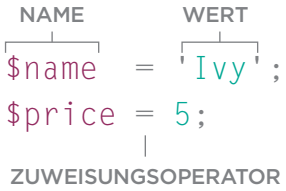
Um eine Variable zu erstellen und einen Wert darin zu speichern, benötigen Sie:

- einen **Variablennamen**, der mit einem Dollarzeichen beginnen muss, gefolgt von einem oder mehreren Wörtern, die die Art der von der Variablen zu speichernden Informationen beschreiben.
- ein **Gleichheitszeichen**, das auch als Zuweisungsoperator bezeichnet wird, weil es dem Variablennamen einen Wert zuweist.
- den **Wert**, der in der Variablen abgelegt werden soll.

Wenn die Variable Text enthält, muss dieser in Anführungszeichen gesetzt werden. Sie können einfache oder doppelte Anführungszeichen verwenden, aber sie müssen zusammenpassen. (Beginnen Sie zum Beispiel nicht mit einem einfachen Anführungszeichen und schließen dann mit einem doppelten Anführungszeichen ab.)

Wenn die Variable eine Zahl oder einen booleschen Wert (`true` oder `false`) enthält, wird dieser nicht in Anführungszeichen gesetzt.

Die Erstellung einer Variablen bezeichnen Programmierer als **Variablendeklaration**. Nach der Deklaration erfolgt dann üblicherweise die **Zuweisung** eines Werts.



Sobald eine Variable deklariert und ihr ein Wert zugewiesen wurde, können Sie den Variablennamen im PHP-Code überall dort nutzen, wo Sie den aktuellen Wert der Variablen benötigen.

Wenn der PHP-Interpreter auf einen Variablennamen stößt, ersetzt er den Namen durch den darin enthaltenen Wert. Im Folgenden zeigen wir den in der oben gezeigten Variablen `$name` gespeicherten Wert mit dem Befehl `echo` an.

VARIABLEN ERSTELLEN UND DARAUF ZUGREIFEN

PHP section_a/c01/variables.php

```
<?php
① $name  = 'Ivy';
② $price = 5;
?>
<!DOCTYPE html>
<html>
  <head>
    <title>Variables</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
③   <h2>Welcome <?php echo $name; ?></h2>
    <p>The cost of your candy is
④      $<?php echo $price; ?> per pack.</p>
  </body>
</html>
```

ERGEBNIS

In diesem Kapitel weisen wir den Variablen direkt im PHP-Code Werte zu. In künftigen Kapiteln stammen die Werte, die in die Variablen geschrieben werden, aus von den Besuchern abgeschickten HTML-Formularen, aus URL-Daten und aus Datenbanken.

In diesem Beispiel werden am Anfang der Seite zwei Variablen erstellt und mit Werten gefüllt:

1. `$name` enthält den Namen des aktuellen Website-Besuchers. Da es sich um Text handelt, steht dieser in Anführungszeichen.

2. `$price` enthält den Preis für eine einzelne Süßigkeit. Da es sich um eine Zahl handelt, steht der Wert nicht in Anführungszeichen.

Als Nächstes sehen Sie den HTML-Code, der an den Browser zurückgesendet wird. Darin wird:

3. der Benutzername mit dem Befehl `echo` in die Seite geschrieben

4. der Süßwarenpreis in die Seite geschrieben

Probieren Sie es: Ändern Sie in Schritt 1 den Wert der Variablen `$name` so ab, dass er Ihren Namen enthält. Speichern Sie die Datei und aktualisieren Sie dann die Seite in Ihrem Browser. Sie sehen daraufhin Ihren Namen.

Probieren Sie es: Ändern Sie in Schritt 2 den Preis auf 2. Speichern Sie die Datei und aktualisieren Sie dann die Seite. Nun wird der neue Preis angezeigt.

ZUR BENENNUNG VON VARIABLEN

Ein Variablename sollte die in der Variablen gespeicherten Daten beschreiben. Halten Sie folgende Regeln ein, um einen Variablennamen zu erstellen.

1

Beginnen Sie mit einem Dollarzeichen (`$`).

✓ `$greeting`

✗ `greeting`

Wenn Sie Variablennamen mit beschreibendem Charakter verwenden, ist Ihr Code leichter zu verstehen und nachzuvollziehen.

Wenn Sie mehrere Wörter verwenden, um die in einer Variablen enthaltenen Daten zu beschreiben, werden diese üblicherweise durch einen Unterstrich getrennt.

2

Darauf folgt ein Buchstabe oder ein Unterstrich (keine Zahl).

✓ `$greeting`

✗ `$2_greeting`

Die Groß- und Kleinschreibung ist bei der Namensgebung zu beachten, sodass `$Score` und `$score` zwei verschiedene Namen bilden würden. Sie sollten es jedoch generell vermeiden, zwei Variablen zu erstellen, die dasselbe Wort in unterschiedlicher Schreibweise verwenden, da Sie damit andere Personen, die Ihren Code lesen, verwirren könnten.

3

Nutzen Sie dann eine beliebige Kombination aus Buchstaben von A bis Z (Groß- und Kleinbuchstaben), Zahlen und Unterstrichen. (Bindestriche oder Punkte sind nicht erlaubt.)

✓ `$greeting_2`

✗ `$greeting-2`

✗ `$greeting.2`

Hinweis: `$this` hat eine besondere Bedeutung. Verwenden Sie es nicht als Variablenname.

✗ `$this`

Aus technischen Gesichtspunkten können Sie Zeichen aus verschiedenen Zeichensätzen verwenden (z. B. chinesische oder kyrillische Zeichen), aber es wird häufig als gängige Praxis angesehen, nur die Buchstaben A–z, Zahlen und Unterstriche zu verwenden (da die Unterstützung anderer Zeichen mit einigen Schwierigkeiten verbunden ist).

SKALARE (EINFACHE) DATENTYPEN

PHP unterscheidet zwischen drei skalaren Datentypen, die Text, Zahlen und boolesche Werte enthalten.

STRING-DATENTYP

Programmierer bezeichnen ein Textstück als Zeichenkette oder **String**. Der String-Datentyp kann aus Buchstaben, Zahlen und anderen Zeichen bestehen, die jedoch zur Darstellung von Text verwendet werden.

```
$name = 'Ivy';
```

Zeichenketten werden immer von einfachen oder doppelten Anführungszeichen umgeben. Das öffnende Anführungszeichen muss mit dem schließenden Anführungszeichen übereinstimmen.

```
✓ $name = 'Ivy';
✓ $name = "Ivy";
✗ $name = "Ivy';
✗ $name = 'Ivy";
```

NUMERISCHE DATENTYPEN

Numerische Datentypen ermöglichen es, mathematische Operationen mit den darin gespeicherten Werten durchzuführen, wie etwa Additionen oder Multiplikationen.

```
$price = 5;
```

Zahlen werden nicht in Anführungszeichen gesetzt. Wenn Sie Zahlen in Anführungszeichen setzen, werden sie statt als Zahlen als Zeichenketten behandelt.

PHP kennt zwei numerische Datentypen:

`int` steht für Integer, also für ganze Zahlen (z. B. 275).

`float` enthält Fließkommazahlen, die Dezimalbrüche darstellen (z. B. 2,75).

BOOLESCHE DATENTYPEN

Der boolesche Datentyp kann nur einen von zwei Werten annehmen: `true` oder `false`. Diese Werte sind in den meisten Programmiersprachen üblich.

```
$logged_in = true;
```

`true` und `false` sollten in Kleinbuchstaben geschrieben und nicht in Anführungszeichen gesetzt werden. Auf den ersten Blick wirken boolesche Werte vielleicht abstrakt, aber mit `true` oder `false` lassen sich viele Dinge ausdrücken, wie zum Beispiel:

- Ist jemand eingeloggt?
- Hat die Person den allgemeinen Geschäftsbedingungen zugestimmt?
- Ist ein Produkt für den kostenlosen Versand qualifiziert?

NULL-DATENTYP

PHP kennt auch den Datentyp `null`. Dieser kann nur den Wert `null` haben. Er zeigt an, dass für eine Variable noch kein Wert definiert wurde.

MIT DATENTYPEN JONGLIEREN

Auf den Seiten 60 und 61 sehen Sie, wie der PHP-Interpreter verschiedene Datentypen ineinander umwandeln kann (z. B. einen String in einen Zahlenwert).

EINEN VARIABLENWERT AKTUALISIEREN

Sie können den in einer Variablen abgelegten Wert ändern oder überschreiben, indem Sie ihr einen neuen Wert zuweisen. Das erfolgt auf die gleiche Weise, wie Sie der Variablen bei der Erstellung einen Wert zuweisen.

1. Die Variable `$name` wird **initialisiert**. Das bedeutet, dass sie deklariert und ihr ein Anfangswert zugewiesen wird. Dieser wird verwendet, sofern die Variable später auf der Seite nicht aktualisiert wird.

Der Anfangswert ist `Guest`; da es sich um Text handelt, wird er in Anführungszeichen geschrieben.

2. Der Variablen `$name` wird nachfolgend der neue Wert `Ivy` zugewiesen.

3. Die Variable `$price` enthält den Preis für eine einzelne Packung Süßigkeiten.

Als Nächstes sehen Sie den HTML-Code, der an den Browser zurückgesendet wird. Darin wird:

4. der Name mit dem Befehl `echo` in die Seite geschrieben.

Angezeigt wird der aktualisierte Wert, der der Variablen `$name` in Schritt 2 zugewiesen wurde.

5. der Süßigkeitenpreis auf die Seite geschrieben.

section_a/c01/updating-variables.php **PHP**

```
<?php
① $name  = 'Guest';
② $name  = 'Ivy';
③ $price = 5;
?>
<!DOCTYPE html>
<html>
  <head>
    <title>Updating Variables</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
④   <h2>Welcome <?php echo $name; ?></h2>
    <p>The cost of your candy is
⑤      $<?php echo $price; ?> per pack.</p>
  </body>
</html>
```

ERGEBNIS

Probieren Sie aus: Ändern Sie in Schritt 2 den Wert der Variablen `$name` so, dass er Ihren Namen enthält. Speichern Sie die Datei und aktualisieren Sie dann die Seite in Ihrem Browser. Nun wird Ihr Name angezeigt.

Probieren Sie aus: Fügen Sie nach Schritt 2 eine neue Zeile ein und geben Sie der Variablen `$name` einen anderen Namen. Speichern Sie die Datei und aktualisieren Sie die Seite. Sie sehen nun den neuen Namen.

ARRAYS

Eine Variable kann auch ein **Array** oder **Datenfeld** enthalten, in dem eine Reihe zusammengehöriger Werte gespeichert ist. Arrays werden auch zu den **zusammengesetzten Datentypen** gezählt, da sie mehr als einen Wert speichern können.

Ein Array ist wie ein Behälter, der eine Reihe zusammengehöriger Variablen enthält. Jeder Eintrag in einem Array wird als **Element** bezeichnet. Ebenso wie eine Variable einen Namen für die Darstellung eines Werts verwendet, besitzt auch jedes Element in einem Array:

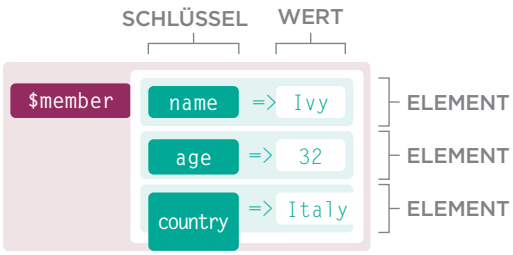
- einen **Schlüssel**, der sich genau wie ein Variablenname verhält
- einen **Wert**, also die zum Namen gehörigen Daten

PHP kennt zwei Arten von Arrays:

- In **assoziativen Arrays** bildet ein Name, der die darunter abgelegten Daten beschreibt, den Schlüssel für jedes Element.
- In **indizierten Arrays** bestehen die Elementschlüssel aus fortlaufenden Zahlen, den sogenannten **Indexwerten**.

ASSOZIATIVES ARRAY

Das nachstehende Array dient der Speicherung von Daten, die für ein Mitglied einer Website stehen. Bei jeder Verwendung des Arrays bleiben die Schlüsselnamen (die die in den einzelnen Elementen des Arrays gespeicherten Daten beschreiben) gleich.

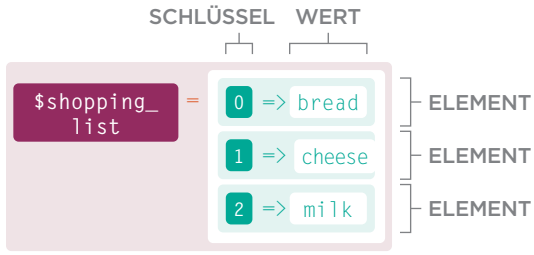


In diesen beiden Beispielen ist jeder im Array gespeicherte Wert ein skalarer Datentyp (ein einzelnes Datenelement).

Auf Seite 44 finden Sie Beispiele für Arrays, deren Elemente selbst auch wieder ein weiteres Array enthalten.

INDIZIERTES ARRAY (STANDARD-ARRAY)

Das nachstehende Array dient zur Erfassung einer Einkaufsliste. Derartige Listen können bei jeder Nutzung eine andere Anzahl von Elementen enthalten. Der Schlüssel nutzt keinen Namen zur Beschreibung der einzelnen Listenelemente, sondern einen aufsteigenden Indexwert (ganzzahlig und bei 0 beginnend).



Hinweis: Die Indexwerte beginnen bei 0, nicht bei 1. Das erste Listenelement hat den Index 0, das zweite Element hat den Index 1 und so weiter. Mit der Indexnummer wird häufig die Reihenfolge der Listenelemente beschrieben.

ASSOZIATIVE ARRAYS

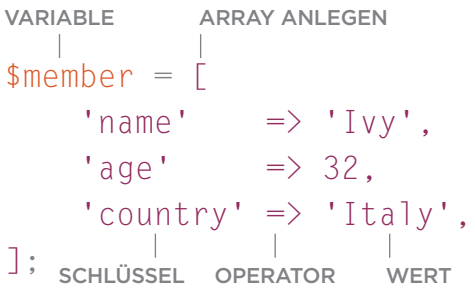
Um ein assoziatives Array zu erstellen, weisen Sie jedem Element (oder Eintrag) im Array einen **Schlüssel** zu, der die darin enthaltenen Daten beschreibt.

Um ein assoziatives Array in einer Variablen zu speichern, verwenden Sie:

- einen Variablenamen, der alle im Array enthaltenen Werte beschreibt
- den Zuweisungsoperator
- eckige Klammern zur Erstellung des Arrays

Innerhalb der eckigen Klammern verwenden Sie:

- den Schlüsselnamen in Anführungszeichen
- den Doppelpfeil-Operator =>
- den Wert für dieses Element (Zeichenketten stehen in Anführungszeichen, Zahlen und boolesche Werte nicht)
- ein Komma nach jedem Element



Ein assoziatives Array kann auch mit der unten dargestellten Syntax erstellt werden, wobei auf das Wort array dann runde (anstelle von eckigen) Klammern folgen.

```
$member = array(
  'name'    => 'Ivy',
  'age'     => 32,
  'country' => 'Italy',
);
```

Um auf ein Element eines assoziativen Arrays zuzugreifen, verwenden Sie:

- den Namen der Variablen, die das Array enthält
- gefolgt von eckigen Klammern und Anführungszeichen
- den Schlüssel des abzurufenden Elements

ASSOZIATIVE ARRAYS ANLEGEN UND DARAUF ZUGREIFEN

PHP section_a/c01/associative-arrays.php

```php
<?php
$nutrition = [
    'fat'   => 16,
    'sugar' => 51,
    'salt'  => 6.3,
];
?>
<!DOCTYPE html>
<html>
  <head> ... </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Nutrition (per 100g)</h2>
    <p>Fat:   <?php echo $nutrition['fat']; ?>%</p>
    <p>Sugar: <?php echo $nutrition['sugar']; ?>%</p>
    <p>Salt:  <?php echo $nutrition['salt']; ?>%</p>
  </body>
</html>
```

ERGEBNIS

Probieren Sie aus: Fügen Sie in Schritt 1 ein weiteres Element in das Array ein. Verwenden Sie den Schlüssel `protein` und weisen Sie ihm einen Wert von 2.6 zu. Lassen Sie dann in Schritt 2 den Proteinwert auf der Seite anzeigen.

1. In diesem Beispiel erstellen wir ein assoziatives Array und speichern es in der Variablen `$nutrition`. Das Array wird in eckigen Klammern erstellt. Es besteht aus drei Elementen (jedes Element umfasst ein Schlüssel/Wert-Paar). Der Operator => ordnet den Schlüsseln die einzelnen Werte zu.

2. Um im Array gespeicherte Daten anzuzeigen, verwenden Sie:

- den `echo`-Befehl, der dafür sorgt, dass der nachfolgende Wert auf der Webseite ausgegeben wird
- gefolgt von dem Namen der Variablen, die das Array enthält
- gefolgt von eckigen Klammern und Anführungszeichen, die den Schlüsselnamen enthalten, auf dessen Wert Sie zugreifen möchten

Um beispielsweise den Zuckergehalt auf der Seite auszugeben, verwenden Sie:

```
echo $nutrition
['sugar'];
```

Probieren Sie aus: Ändern Sie in Schritt 1 die Werte des Arrays. Geben Sie dem Schlüssel:

- `fat` einen Wert von 42
- `sugar` einen Wert von 60
- `salt` einen Wert von 3.5

Speichern und aktualisieren Sie die Seite, um die aktualisierten Werte zu sehen.

INDIZIERTE ARRAYS

Wenn bei der Deklaration eines Arrays nicht für jedes Element ein Schlüssel angegeben wird, weist der PHP-Interpreter ihm eine Zahl zu, die Indexnummer. Indexnummern beginnen bei null (0), nicht bei eins (1).

Um ein indiziertes Array in einer Variablen zu speichern, verwenden Sie:

- einen Variablenamen, der alle im Array enthaltenen Werte beschreibt
- den Zuweisungsoperator
- eckige Klammern zur Erstellung des Arrays

Innerhalb der eckigen Klammern verwenden Sie:

- die Liste der Werte, die im Array gespeichert werden sollen (Zeichenketten stehen in Anführungszeichen, Zahlen und boolesche Werte nicht)
- ein Komma nach jedem Wert

Jedes Element erhält dabei eine Indexnummer.

Oben würde `bread` die Indexnummer 0, `cheese` die 1 und `milk` die 2 erhalten. Mit Indexnummern wird oft die Reihenfolge der im Datenfeld enthaltenen Elemente angegeben.

Ein indiziertes Array kann auch mit der unten dargestellten Syntax erstellt werden, wobei auf das Wort array dann runde (anstelle von eckigen) Klammern folgen.

```
$shopping_list = array('bread',
                       'cheese',
                       'milk');
```

Neue Werte können dem Array in derselben oder in einer neuen Zeile (so wie oben gezeigt) zugewiesen werden.

Um auf die Daten eines indizierten Arrays zuzugreifen, verwenden Sie:

- den Namen der Variablen, die das Array enthält
- gefolgt von eckigen Klammern (keine Anführungszeichen)
- dem Indexwert des Elements, auf das Sie zugreifen möchten (in den eckigen Klammern)

Der nachfolgende Code greift auf das dritte Array-Element zu, in diesem Beispiel als auf den Wert `milk`.

INDIZIERTE ARRAYS ANLEGEN UND DARAUF ZUGREIFEN

PHP section_a/c01/indexed-arrays.php

```php
<?php
$best_sellers = ['Chocolate', 'Mints', 'Fudge',
    'Bubble gum', 'Toffee', 'Jelly beans',];
?>
<!DOCTYPE html>
<html>
  <head> ... </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Best Sellers</h2>
    <ul>
      <li><?php echo $best_sellers[0]; ?></li>
      <li><?php echo $best_sellers[1]; ?></li>
      <li><?php echo $best_sellers[2]; ?></li>
    </ul>
  </body>
</html>
```

ERGEBNIS

1. In diesem Beispiel erstellen wir zunächst die Variable `$best_sellers`. Als Variablenwert enthält sie ein Array mit einer Liste der meistverkauften Artikel auf der Website.

Dieses Array wird mittels eckiger Klammern erstellt, und seine Elemente stehen innerhalb dieser Klammern. Da es sich bei den Elementen im Array um Text handelt, stehen sie zudem in Anführungszeichen. (Zahlen und boolesche Werte gehören nicht in Anführungszeichen.) Auf jedes Element folgt ein Komma.

2. Hier werden die drei meistverkauften Artikel auf die Seite geschrieben:

- der `echo`-Befehl sorgt für die Ausgabe des nachfolgenden Werts
- gefolgt wird er vom Namen der Variablen, die das Array enthält
- danach steht die Indexnummer des abzurufenden Elements in eckigen Klammern. Denken Sie daran, dass Indexnummern bei 0 beginnen, nicht bei 1.

Probieren Sie aus: Fügen Sie in Schritt 1 hinter `Fudge` noch `Licorice` in die Auflistung ein. In Schritt 2 lassen Sie auch noch den 4. und 5. Artikel aus dem Array anzeigen.

ARRAYS AKTUALISIEREN

Nachdem Sie ein Array erstellt haben, können Sie ihm neue Elemente hinzufügen oder den Wert eines bestehenden Elements aktualisieren.

Um einen in einem assoziativen Array abgelegten Wert zu aktualisieren, verwenden Sie:

- den Namen der Variablen, die das Array enthält
- gefolgt von eckigen Klammern
- nun den Schlüsselnamen in Anführungszeichen
- einen Zuweisungsoperator
- den neuen Wert, den es enthalten soll

```
$member['name'] = 'Tom';
```

VARIABLE SCHLÜSSEL NEUER WERT

Um einem assoziativen Array ein neues Element hinzuzufügen, gehen Sie genauso vor wie oben beschrieben, nutzen dabei aber einen neuen Schlüsselnamen (keinen, der bereits im Array verwendet wurde).

Die Anführungszeichen umschließen den Schlüsselnamen, wenn es sich dabei um eine Zeichenkette handelt, da Anführungszeichen einen String-Datentyp anzeigen.

Um einen in einem indizierten Array abgelegten Wert zu aktualisieren, verwenden Sie:

- den Namen der Variablen, die das Array enthält
- gefolgt von eckigen Klammern
- die Indexnummer (ohne Anführungszeichen)
- einen Zuweisungsoperator
- den neuen Wert, den es enthalten soll

```
$shopping_list[2] = 'butter';
```

VARIABLE INDEX-NUMMER NEUER WERT

Wie man Elemente zu indizierten Arrays hinzufügt, erfahren Sie auf Seite 220. Der Ablauf ist etwas anders, da Sie die Position des neuen Elements im Array bestimmen können.

Indexnummern stehen nicht in Anführungszeichen, da numerische Datentypen nicht in Anführungszeichen geschrieben werden.

WELCHER ARRAY-TYP FÜR WELCHE ANWENDUNG?

Assoziative Arrays sind am besten geeignet, wenn Sie:

- genau wissen, welche Informationen das Array enthalten soll. Dies ist erforderlich, um für jedes Element einen Schlüsselnamen zu vergeben.
- einzelne Daten unter Verwendung eines Schlüsselnamens abrufen müssen.

Indizierte Arrays sind hilfreich, wenn Sie:

- nicht wissen, wie viele Daten in dem Array gespeichert werden sollen. (Der Index wird stetig erweitert, je mehr Elemente Sie der Liste hinzufügen.)
- eine Reihe von Werten in einer bestimmten Reihenfolge abspeichern möchten.

IN ARRAYS GESPEICHERTE WERTE VERÄNDERN

PHP section_a/c01/updating-arrays.php

```php
<?php
$nutrition = [
    'fat'   => 38,
    'sugar' => 51,
    'salt'  => 0.25,
];
$nutrition['fat']   = 36;
$nutrition['fiber'] = 2.1;
?>
<!DOCTYPE html>
<html>
  <head> ... </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Nutrition (per 100g)</h2>
    <p>Fat:   <?php echo $nutrition['fat']; ?>%</p>
    <p>Sugar: <?php echo $nutrition['sugar']; ?>%</p>
    <p>Salt:  <?php echo $nutrition['salt']; ?>%</p>
    <p>Fiber: <?php echo $nutrition['fiber']; ?>%</p>
  </body>
</html>
```

ERGEBNIS

1. In diesem Beispiel wird zunächst ein Array in der Variablen `$nutrition` abgelegt.

Die Schlüssel und Werte, die die einzelnen Array-Elemente bilden, müssen nicht unbedingt auf einer neuen Zeile stehen (so wie hier gezeigt), allerdings erhöht diese Praxis die Lesbarkeit.

2. Der Wert für den Fettgehalt wird von 38 auf 36 aktualisiert.

3. Ein neues Element wird zum Array hinzugefügt. Der Schlüssel heißt `fiber` und sein Wert ist `2.1`.

4. Die Werte im Array werden auf der Seite ausgegeben.

Probieren Sie aus: Fügen Sie nach Schritt 3 einen weiteren Schlüssel für Eiweiß (`protein`) hinzu und weisen Sie ihm den Wert `7.3` zu.

ARRAYS IN EINEM ARRAY SPEICHERN

Der Wert eines jeden Elements innerhalb eines Arrays kann ein weiteres Array sein. Enthält jedes Array-Element ein weiteres Array, spricht man von einem **mehrdimensionalen Array**. Dieses eignet sich für die Darstellung von Daten, wie sie etwa in Tabellen vorkommen.

In einigen Fällen müssen Sie einen zusammenhängenden Wertesatz in einem Element eines Arrays speichern (z. B. bei der Darstellung von Daten, die sonst üblicherweise in Tabellen zu finden sind). Sehen Sie sich die rechte Tabelle mit drei Personen, ihrem Alter und ihren jeweiligen Aufenthaltsorten an.

NAME	ALTER	LAND
Ivy	32	UK
Emi	24	Japan
Luke	47	USA

Jede Zeile dieser Tabelle (jede Person) lässt sich durch ein Element eines indizierten Arrays darstellen. Jedes dieser Elemente kann dann wiederum ein assoziatives Array enthalten, in dem der Name, das Alter und das Land der jeweiligen Person gespeichert sind.

Die Indexnummern für das indizierte Array werden automatisch vom PHP-Interpreter zugewiesen. Das Komma nach jedem assoziativen Array zeigt das Ende des Werts für dieses Element an.

```
$members = [
    ['name' => 'Ivy',  'age' => 32, 'country' => 'UK',],
    ['name' => 'Emi',  'age' => 24, 'country' => 'Japan',],
    ['name' => 'Luke', 'age' => 47, 'country' => 'USA',],
];
```

Um das Array mit den Daten über Emi zu erhalten, verwenden Sie:

- den Namen der Variablen, die das indizierte Array enthält
- die Indexnummer des Elements, auf das Sie zugreifen möchten in eckigen Klammern (denken Sie daran, dass die Zählung bei 0 beginnt und dass Zahlen nicht in Anführungszeichen gesetzt werden).

```
$members[1];
```

Um das Alter von Luke abzurufen, verwenden Sie:

- den Namen der Variablen, die das indizierte Array enthält
- die Indexnummer des Elements, das die Informationen über Luke enthält in eckigen Klammern
- den Schlüssel des Elements, auf das Sie im Array über Luke zugreifen möchten in einem zweiten Satz eckiger Klammern (der Schlüssel ist eine Zeichenkette und gehört daher in Anführungszeichen gesetzt)

```
$members[2]['age'];
```

MEHRDIMENSIONALE ARRAYS

PHP section_a/c01/multidimensional-arrays.php

```php
<?php
$offers = [
    ['name' => 'Toffee', 'price' => 5, 'stock' => 120,],
    ['name' => 'Mints',  'price' => 3, 'stock' => 66,],
    ['name' => 'Fudge',  'price' => 4, 'stock' => 97,],
];
?>
<!DOCTYPE html>
<html>
  <head> ... </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Offers</h2>
    <p><?php echo $offers[0]['name']; ?> -
      $<?php echo $offers[0]['price']; ?> </p>
    <p><?php echo $offers[1]['name']; ?> -
      $<?php echo $offers[1]['price']; ?> </p>
    <p><?php echo $offers[2]['name']; ?> -
      $<?php echo $offers[2]['price']; ?> </p>
  </body>
</html>
```

ERGEBNIS

1. Dieses Beispiel beginnt mit der Hinterlegung eines indizierten Arrays in der Variablen `$offers`.

Jedes Element im Array enthält ein assoziatives Array mit dem Namen, Preis und Lagerbestand eines angebotenen Artikels.

2. Der Name des ersten Produkts wird ausgegeben. (Die Indexnummer des ersten Produkts ist 0.)

3. Der Preis des ersten Produkts wird ausgegeben.

4. Der Name und der Preis des zweiten Produkts werden ausgegeben.

5. Der Name und der Preis des dritten Produkts werden ausgegeben.

Probieren Sie es: Fügen Sie in Schritt 1 ein weiteres Produkt mit dem Namen `Chocolate` zum Array hinzu. Setzen Sie den Preis auf 2 und den Lagerbestand auf 83. Geben Sie dann nach Schritt 5 den Namen und den Preis des neu hinzugefügten Produkts aus.

Im nächsten Kapitel lernen Sie, wie Sie mithilfe einer Schleife den Namen und den Preis aller im Array `$offers` enthaltenen Produkte ausgeben können, und zwar unabhängig von der Anzahl der darin enthaltenen Produkte.

KURZFORM FÜR ECHO

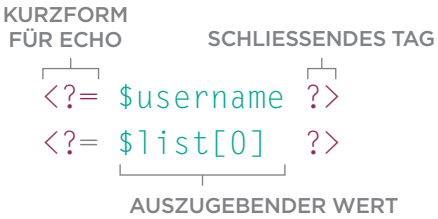
Wenn ein PHP-Block nur dazu dient, einen Wert in den Browser zu schreiben, können Sie anstelle von `<?php echo ?>` eine Abkürzung verwenden.

Anstatt `<?php echo $name; ?>` zu schreiben, können Sie die Abkürzung `<?= $name ?>` verwenden.

Dies ist die einzige Situation, in der Sie nicht das vollständige öffnende `<?php`-Tag brauchen.

Das können Sie weglassen:

- Die Buchstaben `php` im öffnenden Tag
- Den `echo`-Befehl
- Den Strichpunkt vor dem schließenden Tag



In vielen der Beispiele in den ersten Buchkapiteln wird deutlich, dass jede PHP-Datei aus zwei Teilen besteht:

- Zuerst speichert der PHP-Code Werte in Variablen oder Arrays. (Er kann auch bestimmte Aktionen mit den darin enthaltenen Daten durchführen.)
- Danach folgt der HTML-Code, der an den Browser zurückgesendet wird. Dieser zweite Teil der Seite gibt mit der oben gezeigten Kurzsyntax Werte aus, die in Variablen gespeichert wurden.

Wenn Sie zu Beginn jeder Seite die auf der Seite anzuzeigenden Werte erzeugen und in Variablen speichern, können Sie eine klare Trennung zwischen dem auf dem Server ausgeführten PHP-Code und dem HTML-Code, der letztlich im Browser dargestellt wird, herstellen.

Im zweiten Teil der Datei, in dem die HTML-Seite erstellt wird, sollte so wenig PHP-Code wie möglich vorkommen. In unseren ersten Beispielen schreibt der PHP-Code in diesem Teil der Seite lediglich in Variablen gespeicherte Werte in die HTML-Seite.

DIE KURZFORM FÜR ECHO VERWENDEN

PHP section_a/c01/echo-shorthand.php

```
<?php
① $name      = 'Ivy';
② $favorites = ['Chocolate', 'Toffee', 'Fudge',];
?>
<!DOCTYPE html>
<html>
  <head>
    <title>Echo Shorthand</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
③   <h2>Welcome <?= $name ?></h2>
    <p>Your favorite type of candy is:
④      <?= $favorites[0] ?>.</p>
  </body>
</html>
```

ERGEBNIS

In diesem Beispiel werden zwei verschiedene Variablen erstellt und am Anfang der Seite mit Werten gefüllt, bevor der HTML-Code beginnt:

1. `$name` enthält den Namen eines Website-Nutzers. Da es sich um Text handelt, wird er in Anführungszeichen gesetzt.

2. `$favorites` enthält eine Reihe von Lieblingssüßigkeiten dieses Mitglieds.

3. Der Name wird mithilfe der Kurzschreibweise des `echo`-Befehls in die Seite geschrieben.

4. Die Lieblingssüßigkeit des Mitglieds wird mit der Kurzschreibweise des `echo`-Befehls in die Seite geschrieben.

Probieren Sie es: Ändern Sie in Schritt 1 den in der Variablen `$name` gespeicherten Wert in Ihren eigenen Namen. In Schritt 2 setzen Sie Ihre Lieblingssüßigkeit an den Anfang des Arrays. Speichern Sie die Datei und aktualisieren Sie die Seite in Ihrem Browser. Daraufhin sehen Sie, wie sich der Inhalt ändert.

AUSDRÜCKE & OPERATOREN

Häufig wird aus zwei (oder mehr) Werten ein neuer Wert erzeugt. **Ausdrücke** bestehen aus einem oder mehreren Konstrukten, die zur Ausgabe eines einzelnen Werts führen. Ausdrücke nutzen hierfür **Operatoren**.

Bei den Grundrechenarten (Addition, Subtraktion, Multiplikation und Division) werden zwei Werte zu einem neuen Wert verrechnet. Der folgende Ausdruck multipliziert die Zahl 3 mit der Zahl 5 und erzeugt dabei den Wert 15:

```
3 * 5
```

Ausdrücke ergeben also einen einzelnen Wert. Nachfolgend wird der neu erzeugte Wert in der Variablen `$total` gespeichert:

```
$total = 3 * 5;
```

Die Zeichen `+ - * / =` werden als **Operatoren** bezeichnet.

Mit dem String-Operator (**Verknüpfungsoperator**) können Sie mindestens zwei Strings zu einem einzelnen längeren Text zusammenfügen. Der folgende Ausdruck verknüpft die Werte »Hi« und »Ivy« zu einer einzelnen Zeichenkette.

```
$greeting = 'Hi ' . 'Ivy';
```

Die Verknüpfung dieser beiden Zeichenketten ergibt den einzelnen Wert `Hi Ivy`, der in der Variablen `$greeting` gespeichert wird.

Im weiteren Verlauf dieses Kapitels lernen Sie die auf der rechten Seite aufgeführten Operatoren kennen.

ARITHMETISCHE OPERATOREN

Seite 50 bis 51

Mit arithmetischen Operatoren können Sie Zahlen verarbeiten und beispielsweise Additionen, Subtraktionen, Multiplikationen und Divisionen durchführen.

Wenn zum Beispiel jemand 3 Päckchen Süßigkeiten kauft und jedes Päckchen 5 $ kostet, können Sie mit einem Multiplikationsoperator den Gesamtpreis für diese drei Süßigkeitenpakete berechnen.

STRING-OPERATOREN

Seite 52 bis 53

Mit String-Operatoren können Sie Texte verarbeiten. Es gibt zwei String-Operatoren, mit denen Sie verschiedene Textstücke zu einer einzigen Zeichenkette zusammenfügen können.

Wenn Sie zum Beispiel den Vornamen eines Mitglieds in einer Variablen und den Nachnamen in einer zweiten Variablen gespeichert haben, können Sie diese beiden Variablen miteinander kombinieren, um den vollständigen Namen zu erhalten.

VERGLEICHSOPERATOREN

Seite 54 bis 55 und 58

Wie der Name schon sagt, *vergleichen* Vergleichsoperatoren zwei Werte und geben einen booleschen Wert (entweder `true` oder `false`) zurück.

Im Falle der beiden Zahlen 3 und 5 könnten Sie mit einem Vergleich beispielsweise feststellen, ob:

- 3 größer als 5 ist (`false`)
- 3 gleich 5 ist (`false`)
- 3 kleiner als 5 ist (`true`)

Auch Zeichenketten können Sie vergleichen, um festzustellen, ob ein Wert größer oder kleiner als ein anderer ist:

- `'Apple'` ist größer als `'Banana'` (`false`)
- `'A'` ist gleich `'B'` (`false`)
- `'A'` ist kleiner als `'B'` (`true`)

LOGISCHE OPERATOREN

Seite 56 bis 57 und 59

Die drei logischen Operatoren `and`, `or` und `not` beziehen sich immer auf zwei Eingabewerte, entweder `true` oder `false`. Um ihre Funktionsweise zu verstehen, betrachten Sie die beiden folgenden Fragen; beide können mit wahr oder falsch beantwortet werden:

Ist es heiß? Ist es sonnig?

- Der `and`-Operator prüft, ob es zugleich heiß **und** sonnig ist.
- Der `or`-Operator prüft, ob es entweder heiß **oder** sonnig ist.
- Mit dem `not`-Operator können Sie prüfen, ob jeweils nur die Antwort auf eine dieser Fragen nicht zutrifft.
- Zum Beispiel: Ist es nicht sonnig?

Als Ergebnis erhalten Sie jeweils entweder den Wert `true` oder `false`.

ARITHMETISCHE OPERATOREN

In PHP können Sie die folgenden mathematischen Operatoren mit Zahlen und Variablen, in denen Zahlen gespeichert sind, verwenden.

NAME	OPERATOR	ZWECK	BEISPIEL	ERGEBNIS
Addition	`+`	Einen Wert zu einem anderen hinzuzählen	`10 + 5`	`15`
Subtraktion	`-`	Einen Wert von einem anderen abziehen	`10 - 5`	`5`
Multiplikation	`*`	Zwei Werte miteinander multiplizieren	`10 * 5`	`50`
Division	`/`	Einen Wert durch einen anderen teilen	`10 / 5`	`2`
Modulo	`%`	Einen Wert durch einen anderen teilen und den Rest ausgeben	`10 % 3`	`1`
Potenzierung	`**`	Einen Wert mit einem anderen potenzieren	`10 ** 5`	`100000`
Erhöhung	`++`	Den Zahlenwert um eins erhöhen	`$i = 10;` `$i++;`	`11`
Verringerung	`--`	Den Zahlenwert um eins absenken	`$i = 10;` `$i--;`	`9`

AUSFÜHRUNGSREIHENFOLGE

Sie können innerhalb eines Ausdrucks mehrere arithmetische Operationen durchführen, aber dabei müssen Sie die Reihenfolge beachten, in der das Ergebnis berechnet wird: Multiplikationen und Divisionen werden vor Additionen und Subtraktionen ausgeführt.

Dies kann Auswirkungen auf das von Ihnen erwartete Ergebnis haben. Die Zahlen hier werden beispielsweise von links nach rechts berechnet. Das Ergebnis ist 16:

```
$total = 2 + 4 + 10;
```

Im folgenden Beispiel ist das Ergebnis jedoch 42 (nicht 60):

```
$total = 2 + 4 * 10;
```

Mithilfe von Klammern können Sie bestimmen, welche Operation zuerst durchgeführt werden soll. Die folgende Berechnung liefert also ein Ergebnis von 60:

```
$total = (2 + 4) * 10;
```

Die Klammern zeigen an, dass zunächst 2 und 4 zusammengezählt werden, *bevor* das Ergebnis mit 10 multipliziert wird.

ARITHMETISCHE OPERATOREN VERWENDEN

PHP section_a/c01/arithmetic-operators.php

```php
<?php
① $items    = 3;
② $cost     = 5;
③ $subtotal = $cost * $items;
④ $tax      = ($subtotal / 100) * 20;
⑤ $total    = $subtotal + $tax;
?>
<!DOCTYPE html>
  <html>
  <head> ... </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Shopping Cart</h2>
    <p>Items: <?= $items ?></p>
    <p>Cost per pack: $<?= $cost ?></p>
⑥   <p>Subtotal: $<?= $subtotal ?></p>
    <p>Tax: $<?= $tax ?></p>
    <p>Total: $<?= $total ?></p>
  </body>
</html>
```

ERGEBNIS

Dieses Beispiel zeigt, wie mathematische Operatoren in Verbindung mit Zahlen eingesetzt werden, um den Gesamtpreis einer Bestellung zu berechnen. Zunächst benötigen Sie zwei Variablen zur Speicherung der:

1. Gesamtzahl der bestellten Artikel (`$items`)

2. Kosten für eine einzelne Packung Süßigkeiten (`$cost`)

Anschließend erfolgen die Berechnungen, und die Ergebnisse werden in Variablen gespeichert. Erst danach wird der HTML-Code erstellt. Dies hilft, den PHP-Code vom HTML-Inhalt zu trennen.

3. Zur Berechnung der Auftragskosten wird die Anzahl der Artikel mit dem Preis für eine Packung Süßigkeiten multipliziert.

4. Nachfolgend muss die Steuer in Höhe von 20 % hinzugerechnet werden. Dazu wird die Zwischensumme durch 100 geteilt. (Dies geschieht in Klammern, um sicherzustellen, dass sie zuerst berechnet wird.) Anschließend wird das Ergebnis mit 20 multipliziert.

5. Schließlich wird die Steuer zur Zwischensumme hinzuaddiert, um den Gesamtbetrag zu ermitteln.

6. Die in Variablen abgelegten Ergebnisse werden dann auf der HTML-Seite ausgegeben.

Probieren Sie es: Verändern Sie die Stückpreise in Schritt 1 und die Menge in Schritt 2.

ZEICHENKETTEN-OPERATOREN

Möglicherweise müssen Sie zwei oder mehrere Zeichenketten zusammenfügen, um daraus einen einzigen Wert zu erzeugen.

Eine solche Operation wird als Verknüpfung oder Verkettung von Strings bezeichnet.

VERKKNÜPFUNGSOPERATOR

Der Verknüpfungsoperator ist ein Punktsymbol. Er verknüpft den Wert in einer Zeichenkette mit dem Wert in einer weiteren. Im folgenden Beispiel würde der Variablen `$name` die Zeichenfolge `'Ivy Stone'` zugewiesen:

```
$forename = 'Ivy';
$surname  = 'Stone';
$name     = $forename . ' ' . $surname;
```

Beachten Sie, dass zwischen die Variablen `$forename` und `$surname` noch ein separates Leerzeichen eingefügt wird; ohne dieses Leerzeichen würde die Variable `$name` den Wert `IvyStone` enthalten.

Sie können beliebig viele Zeichenketten in einer Anweisung verknüpfen, sofern Sie zwischen den einzelnen Zeichenketten konsequent den Verknüpfungsoperator anwenden.

Sie können in Variablen gespeicherte Zeichenketten auch ganz ohne Verknüpfungsoperator zusammenfügen. Wenn ein Wert mit doppelten (statt einfachen) Anführungszeichen zugewiesen wird, ersetzt der PHP-Interpreter die Variablennamen in doppelten Anführungszeichen durch die darin enthaltenen Werte. Im folgenden Fall würde `$name` daher den Wert `Ivy Stone` enthalten:

```
$name = "$forename $surname";
```

VERKNÜPFENDER ZUWEISUNGSOPERATOR

Wenn Sie einen Text an eine bereits vorhandene Variable anhängen möchten, können Sie den zuweisenden Verknüpfungsoperator verwenden. Diesen können Sie sich als Kurzform zur Erzeugung eines aktualisierten Strings vorstellen:

```
$greeting = 'Hello ';
$greeting .= 'Ivy';
```

Hier wird die Zeichenkette `'Hello '` in der Variablen `$greeting` gespeichert. In der nächsten Zeile fügt der verknüpfende Zuweisungsoperator die Zeichenkette `'Ivy'` hinten an den bestehenden Wert der Variablen `$greeting` an.

Jetzt lautet der Wert der Variablen `$greeting` `'Hello Ivy'`. Wie Sie sehen, spart dies gegenüber dem links gezeigten Beispiel eine Zeile Code ein.

ZEICHENKETTEN ZUSAMMENFÜGEN

PHP section_a/c01/string-operator.php

```
<?php
① $prefix  = 'Thank you';
② $name    = 'Ivy';
③ $message = $prefix . ', ' . $name;
?>
<!DOCTYPE html>
<html>
  <head>
    <title>String Operator</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
    <h2><?= $name ?>'s Order</h2>
    <p><?= $message ?></p>
  </body>
</html>
```

ERGEBNIS

Dieses Beispiel zeigt eine personalisierte Nachricht an.

1. Zunächst wird die Variable `$prefix` erstellt, um den Anfang der Nachricht zu speichern. Sie enthält die Worte `'Thank you'`.

2. Eine zweite neu erstellte Variable nimmt den Namen des Besuchers oder der Besucherin auf. Die Variable heißt `$name` und die Besucherin heißt `Ivy`.

3. Die persönliche Nachricht wird durch Verknüpfung (oder Verkettung) dreier Werte erzeugt und der neue Wert in der Variablen `$message` gespeichert:

- Zunächst wird der in `$prefix` gespeicherte Wert zu `$message` hinzugefügt
- Dann werden ein Komma und ein Leerzeichen angehängt
- Schließlich wird der in `$name` gespeicherte Wert angefügt

Probieren Sie es: Ändern Sie in Schritt 2 den in `$name` gespeicherten Wert in Ihren Namen um.

Probieren Sie es: Weisen Sie der Variablen `$message` in Schritt 3 ihren Wert mithilfe doppelter Anführungszeichen (und ohne Verknüpfungsoperator) zu:

```
$message = "$prefix $name";
```

VERGLEICHSOPERATOREN VERWENDEN

Mithilfe von Vergleichsoperatoren können Sie zwei oder mehr Werte miteinander vergleichen. Das Ergebnis ist ein boolescher Wert, der entweder wahr oder falsch (`true` oder `false`) ist.

IST GLEICH

Dieser Operator vergleicht zwei Werte, um festzustellen, ob sie gleich sind.

`'Hello' == 'Hello'` ergibt `true`,

da es sich um denselben String handelt.

`'Hello' == 'Goodbye'` ergibt `false`,

da die beiden Strings unterschiedlich sind.

IST NICHT GLEICH

Diese Operatoren vergleichen zwei Werte, um festzustellen, ob sie ungleich sind.

`'Hello' != 'Hello'` ergibt `false`

da es sich um denselben String handelt.

`'Hello' != 'Goodbye'` ergibt `true`

da die beiden Strings unterschiedlich sind.

Mit den oben genannten Operatoren kann der PHP-Interpreter feststellen, ob die beiden Werte einander gleich sind oder nicht. Die nachfolgenden Operatoren sind strikter, da sie sowohl den Wert als auch den Datentyp überprüfen.

Die oben genannten Operatoren würden 3 (eine ganze Zahl) und 3.0 (eine Fließkommazahl) als gleich betrachten. Für die nachfolgenden Operatoren gilt dies nicht. (Auf den Seiten 60 und 61 sehen Sie, wie 0 auch als boolescher Wert `false` und 1 als `true` interpretiert werden kann).

IST IDENTISCH MIT

Dieser Operator vergleicht zwei Werte, um festzustellen, ob sowohl deren Wert als auch deren Datentyp übereinstimmen.

`'3' === 3` ergibt `false`,

weil es sich nicht um denselben Datentyp handelt.

`'3' === '3'` ergibt `true`

weil sowohl Datentyp als auch Wert übereinstimmen.

IST NICHT IDENTISCH MIT

Dieser Operator vergleicht zwei Werte, um zu prüfen, ob sie im Wert und/oder Datentyp voneinander abweichen.

`3.0 !== 3` ergibt `true`

weil es sich nicht um denselben Datentyp handelt.

`3.0 !== 3.0` ergibt `false`

weil sowohl Datentyp als auch Wert übereinstimmen.

Wenn Sie einen booleschen Wert mittels `echo` auf der Seite ausgeben, wird bei `true` eine 1 angezeigt und bei `false` gar nichts.

KLEINER ALS und GRÖSSER ALS

`<` prüft, ob der Wert auf der linken Seite kleiner ist als der Wert auf der rechten Seite.

`4 < 3` ergibt `false` `3 < 4` ergibt `true`

`>` prüft, ob der Wert auf der linken Seite größer ist als der Wert auf der rechten Seite.

`z > a` ergibt `true` `a > z` ergibt `false`

KLEINER ODER GLEICH und GRÖSSER ODER GLEICH

`<=` prüft, ob der Wert auf der linken Seite kleiner oder gleich dem Wert auf der rechten Seite ist.

`4 <= 3` ergibt `false` `3 <= 4` ergibt `true`

`>=` prüft, ob der Wert auf der linken Seite größer oder gleich dem Wert auf der rechten Seite ist.

`z >= a` ergibt `true` `z >= z` ergibt `true`

RAUMSCHIFF-OPERATOR

Der Raumschiff-Operator vergleicht die links und rechts von ihm stehenden Werte und liefert:

`0` für zwei gleiche Werte

`1` für einen größeren Wert auf der linken Seite

`-1` für einen größeren Wert auf der rechten Seite

Dieser Operator wurde erst mit PHP 7 eingeführt (und funktioniert mit früheren PHP-Versionen nicht).

`1 <=> 1` ergibt: `0`

`2 <=> 1` ergibt: `1`

`2 <=> 3` ergibt: `-1`

LOGISCHE OPERATOREN

Vergleichsoperatoren liefern einen einzelnen Wert, der entweder `true` oder `false` ist. Mehrere Vergleichsoperatoren können zusammen mit logischen Operatoren eingesetzt werden, um die Ergebnisse mehrerer Ausdrücke zu vergleichen.

In dieser einen Codezeile stehen drei Ausdrücke, von denen jeder einen einzigen Wert (entweder `true` oder `false`) liefert.

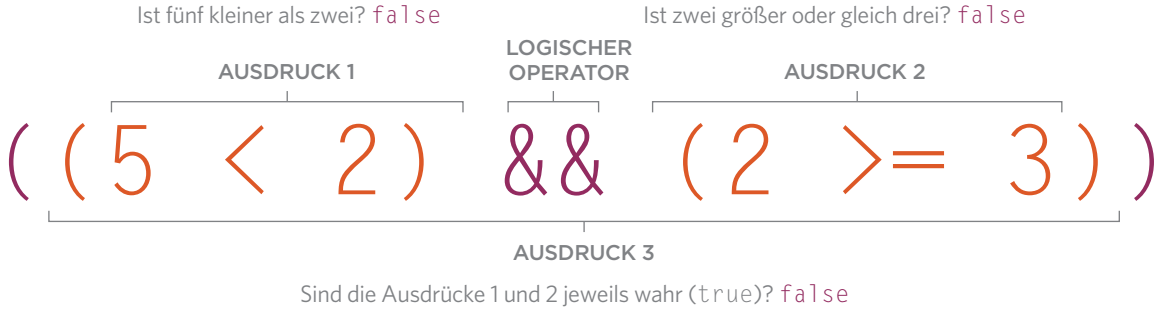
Ausdruck 1 (links) und Ausdruck 2 (rechts) enthalten beide einen Vergleichsoperator, und beide Ausdrücke liefern den Wert `false`.

Ausdruck 3 nutzt einen logischen Operator (und keinen Vergleichsoperator).

Der logische Und-Operator (`&&`) prüft, ob die Ausdrücke auf *beiden* Seiten den Wert `true` liefern. In diesem Fall trifft das nicht zu, sodass der gesamte Ausdruck den Wert `false` ergibt.

Die Ausdrücke 1 und 2 werden vor Ausdruck 3 ausgewertet.

Jeder Ausdruck steht für sich in Klammern. Auf diese Weise wird besser deutlich, dass der Code in jedem einzelnen Klammernpaar einen einzigen Wert ergeben sollte. Es funktioniert auch ohne die Klammern, ist dann aber viel schlechter lesbar.



&&

LOGISCHES UND

Dieser Operator überprüft mehrere Bedingungen.

```
((2 < 5) && (3 >= 2))
```

Dies ergibt den Wert `true`.

Wenn beide Ausdrücke den Wert `true` haben, ergibt der Ausdruck `true`.

Wenn einer der Ausdrücke den Wert `false` hat, liefert der Ausdruck ebenfalls den Wert `false`.

`true && true` ergibt `true`

`true && false` ergibt `false`

`false && true` ergibt `false`

`false && false` ergibt `false`

Statt der beiden kaufmännischen Und-Zeichen können Sie auch das Wort `and` verwenden.

||

LOGISCHES ODER

Dieser Operator überprüft mindestens eine Bedingung.

```
((2 < 5) || (2 < 1))
```

Dies ergibt den Wert `true`.

Wenn einer der beiden Ausdrücke den Wert `true` hat, liefert der Ausdruck `true` zurück.

Wenn beide Ausdrücke `false` sind, ergibt der Ausdruck auch `false`.

`true || true` ergibt `true`

`true || false` ergibt `true`

`false || true` ergibt `true`

`false || false` ergibt `false`

Statt der beiden Pipe-Symbole können sie auch das Wort `or` verwenden.

!

LOGISCHES NICHT

Dieser Operator negiert einen einzelnen booleschen Wert.

```
!(2 < 1)
```

Dies ergibt den Wert `true`.

Das ! negiert einen Ausdruck. Wenn er `false` ist (ohne das ! davor), ergibt sich dadurch `true`. Wenn die Aussage `true` ist, ergibt sich `false`.

`!true` ergibt `false`

`!false` ergibt `true`

Sie können statt des Ausrufezeichen nicht das Wort `not` verwenden.

VERKÜRZTE AUSWERTUNG

Logische Ausdrücke werden von links nach rechts ausgewertet. Sobald der erste Ausdruck ausgewertet wurde und der PHP-Interpreter den logischen Operator kennt, kann er möglicherweise auf die Auswertung der zweiten Bedingung verzichten. Warum, sehen Sie in den Beispielen rechts.

```
((5 < 2) && (2 >= 2))
```

Ein falscher Wert (`false`) wurde gefunden.

Es hat keinen Sinn, die zweite Bedingung zu überprüfen, da die beiden Bedingungen zusammen nicht mehr den Wert `true` ergeben können.

```
((2 < 5) || (2 >= 2))
```

Ein wahrer Wert (`true`) wurde gefunden.

Es hat keinen Sinn, die zweite Bedingung zu überprüfen, da mindestens einer der Werte `true` lautet.

VERGLEICHSOPERATOREN VERWENDEN

1. Es werden drei Variablen angelegt:

- Die erste enthält die Art der gewünschten Süßigkeiten.
- Die zweite zeigt, dass 5 Päckchen auf Lager sind.
- Die dritte zeigt, dass 8 Päckchen gewünscht sind.

2. Ein Vergleichsoperator überprüft, ob die gewünschte Menge kleiner oder gleich der verfügbaren Menge ist. Das Ergebnis wird in der Variablen `$can_buy` gespeichert.

3. Sie werden kaum jemals einen booleschen Wert in eine Seite schreiben (so wie es hier gezeigt wird). Viel wahrscheinlicher werden Sie den Wert in einer bedingten Logik einsetzen, wie Sie sie im nächsten Kapitel kennenlernen. Aber es ist wichtig zu sehen, was Sie beim Versuch, einen solchen booleschen Wert auszuschreiben, erhalten:

- für `true` zeigt die Seite `1` an
- für `false` zeigt die Seite nichts an

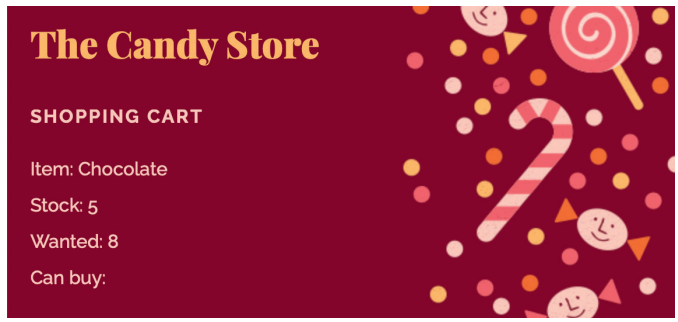
Probieren Sie es: Vertauschen Sie in Schritt 1 die Werte in `$stock` und `$wanted`. Dadurch ändert sich der Wert in `$can_buy`.

Auf Seite 75 erfahren Sie, wie Sie verschiedene Meldungen anzeigen können, wenn ein Ausdruck mit Vergleichsoperator `true` oder `false` zurückliefert.

section_a/c01/comparison-operators.php **PHP**

```
<?php
$item    = 'Chocolate';
$stock   = 5;                                    (1)
$wanted  = 8;
$can_buy = ($wanted <= $stock);                  (2)
?>
<!DOCTYPE html>
<html>
  <head> ... </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Shopping Cart</h2>
    <p>Item:    <?= $item ?></p>
    <p>Stock:   <?= $stock ?></p>
    <p>Wanted:  <?= $wanted ?></p>
    <p>Can buy: <?= $can_buy ?></p>              (3)
  </body>
</html>
```

ERGEBNIS



LOGISCHE OPERATOREN VERWENDEN

PHP section_a/c01/logical-operators.php

```php
<?php
$item    = 'Chocolate';
$stock   = 5;
$wanted  = 3;
$deliver = true;
$can_buy = (($wanted <= $stock) && ($deliver == true));
?>
<!DOCTYPE html>
<html>
  <head> ... </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Shopping Cart</h2>
    <p>Item:    <?= $item ?></p>
    <p>Stock:   <?= $stock ?></p>
    <p>Ordered: <?= $wanted ?></p>
    <p>Can buy: <?= $can_buy ?></p>
  </body>
</html>
```

ERGEBNIS

Dieses Beispiel baut auf dem links gezeigten Beispiel auf.

1. Die Kundin möchte hier nur 3 Päckchen Süßwaren kaufen.

2. Die Variable `$deliver` wird angelegt; sie speichert einen booleschen Wert, der angibt, ob eine Lieferung möglich ist oder nicht.

3. Dieser Ausdruck verwendet zwei Vergleichsoperatoren:

- Die erste überprüft, ob genügend Artikel auf Lager sind
- Die zweite überprüft, ob der Artikel geliefert werden kann

Mit dem logischen Operator `&&` wird geprüft, ob beide Bedingungen den Wert `true` ergeben. Wenn dies der Fall ist, ist der Wert von `$can_buy` `true`, und auf der Seite wird die Zahl 1 ausgegeben.

Sind nicht beide Bedingungen erfüllt, enthält `$can_buy` den Wert `false` und es wird gar nichts angezeigt.

Probieren Sie es: Vertauschen Sie in Schritt 1 die Werte in `$stock` und `$wanted`. Dadurch ändert sich der Wert in `$can_buy`.

Auf Seite 75 erfahren Sie, wie Sie verschiedene Meldungen anzeigen können, wenn ein Ausdrucks `true` oder `false` zurückliefert.

TYPE JUGGLING: AUTOMATISCHE TYPUMWANDLUNG

Der PHP-Interpreter kann einen Wert von einem Datentyp in einen anderen umwandeln. Dies wird auch als »Type Juggling« bezeichnet und kann zu unerwarteten Ergebnissen führen.

PHP ist als Sprache mit **schwacher Typisierung** bekannt, weil Sie bei der Variablendeklaration keinen Datentyp für den zu speichernden Wert angeben müssen. Im Folgenden enthält die Variable `$title` zunächst eine Zeichenkette und dann eine ganze Zahl:

```
$title = 'Ten';  // String
$title = 10;     // Integer
```

Im Unterschied dazu müssen Programmierer in Programmiersprachen mit **strengerer Typisierung** (wie z. B. C++ oder C#) bei der Deklaration jeder Variablen den Datentyp angeben, den sie enthalten soll.

Wenn der PHP-Interpreter auf einen Wert stößt, der nicht den erwarteten Datentyp enthält, kann er versuchen, den Wert in den erwarteten Datentyp zu konvertieren. Dieser Prozess wird auch als **Type Juggling** oder Type Casting bezeichnet.

Die automatische Typumwandlung kann zu Problemen führen, da der PHP-Interpreter mitunter überraschende Ergebnisse oder Fehler erzeugt. Der unten stehende Additionsoperator addiert zum Beispiel zwei Werte zusammen. Die Zahl 1 ist eine Ganzzahl, die 2 ist allerdings ein String, weil sie in Anführungszeichen steht.

```
$total =  1 + '2';
```

In diesem Fall versucht der PHP-Interpreter automatisch, die Zeichenkette in eine Zahl umzuwandeln, damit er die Berechnung durchführen kann. Im Ergebnis enthält die Variable `$total` daher die Zahl 3.

Auf der rechten Seite sehen Sie, nach welchen Regeln ein Wert von einem Datentyp in einen anderen umgewandelt wird. Beispiele für das Type Juggling finden Sie unter: `http://notes.re/php/type-juggling`.

Wenn der Datentyp eines Werts in einen anderen Datentyp umgewandelt wird, sprechen Programmierer davon, dass der Datentyp des Werts von einem Typ in einen anderen **gecastet** wird. Das Type Juggling wird als implizites Casting oder **implizite Typumwandlung** bezeichnet, da es vom PHP-Interpreter selbstständig durchgeführt wird.

Wenn Sie den Datentyp eines Werts explizit mittels Code verändern, wird dies als explizites Casting oder **explizite Typumwandlung** bezeichnet, da der PHP-Interpreter explizit angewiesen wurde, den Datentyp zu ändern.

ZAHLEN

Wenn der PHP-Interpreter zwei Zahlen erwartet, kann er eine mathematische Operation mit diesen durchführen.

Nachfolgend sehen Sie, was passiert, wenn:

- eine Zeichenkette zu einer Zahl addiert wird
- ein boolescher Wert zu einer Zahl addiert wird

ZAHL + STRING	BEHANDELT ALS	ERGEBNIS	BESCHREIBUNG
`1 + '1'`	`1 + 1`	`2` (int)	String enthält eine gültige Ganzzahl. Sie wird als Ganzzahl behandelt.
`1 + '1.2'`	`1 + 1.2`	`2.2` (float)	String enthält einen Fließkommawert. Er wird als Fließkommawert behandelt.
`1 + '1.2e+3'`	`1 + 1200`	`1201` (float)	String enthält einen Fließkommawert mit einem e (Zehnerpotenz). Er wird als Fließkommawert behandelt.
`1 + '5star'`	`1 + 5`	`6` (int)	String enthält eine Ganzzahl, gefolgt von weiteren Zeichen. Die Zahl wird als Ganzzahl behandelt. Nachfolgende Zeichen werden ignoriert.
`1 + '3.5star'`	`1 + 3.5`	`4.5` (float)	String enthält einen Fließkommawert, gefolgt von weiteren Zeichen. Die Zahl wird als Fließkommawert behandelt. Nachfolgende Zeichen werden ignoriert.
`1 + 'star9'`	`1 + 0`	`1` (int)	String beginnt mit etwas anderem als einer Ganzzahl oder einem Fließkommawert. Er wird wie die Zahl 0 behandelt.

ZAHL + BOOLESCHER WERT	BEHANDELT ALS	ERGEBNIS	BESCHREIBUNG
`1 + true`	`1 + 1`	`2` (int)	Boolescher Wert `true` wird als Ganzzahl `1` behandelt.
`1 + false`	`1 + 0`	`1` (int)	Boolescher Wert `false` wird als Ganzzahl `0` behandelt.

ZEICHENKETTEN

Wenn der PHP-Interpreter versucht, zwei Zeichenketten zu verknüpfen, folgt er diesen Regeln.

Nachfolgend sehen Sie, was passiert, wenn PHP:

- eine Zeichenkette mit einer Zahl verknüpft
- eine Zeichenkette mit einem booleschen Wert verknüpft

STRING . ZAHL	BEHANDELT ALS	ERGEBNIS	BESCHREIBUNG
`'Hi ' . 1`	`'Hi ' . '1'`	`Hi 1` (string)	Ganzzahl wird als String behandelt.
`'Hi ' . 1.23`	`'Hi ' . '1.23'`	`Hi 1.23` (string)	Fließkommazahl wird als String behandelt.

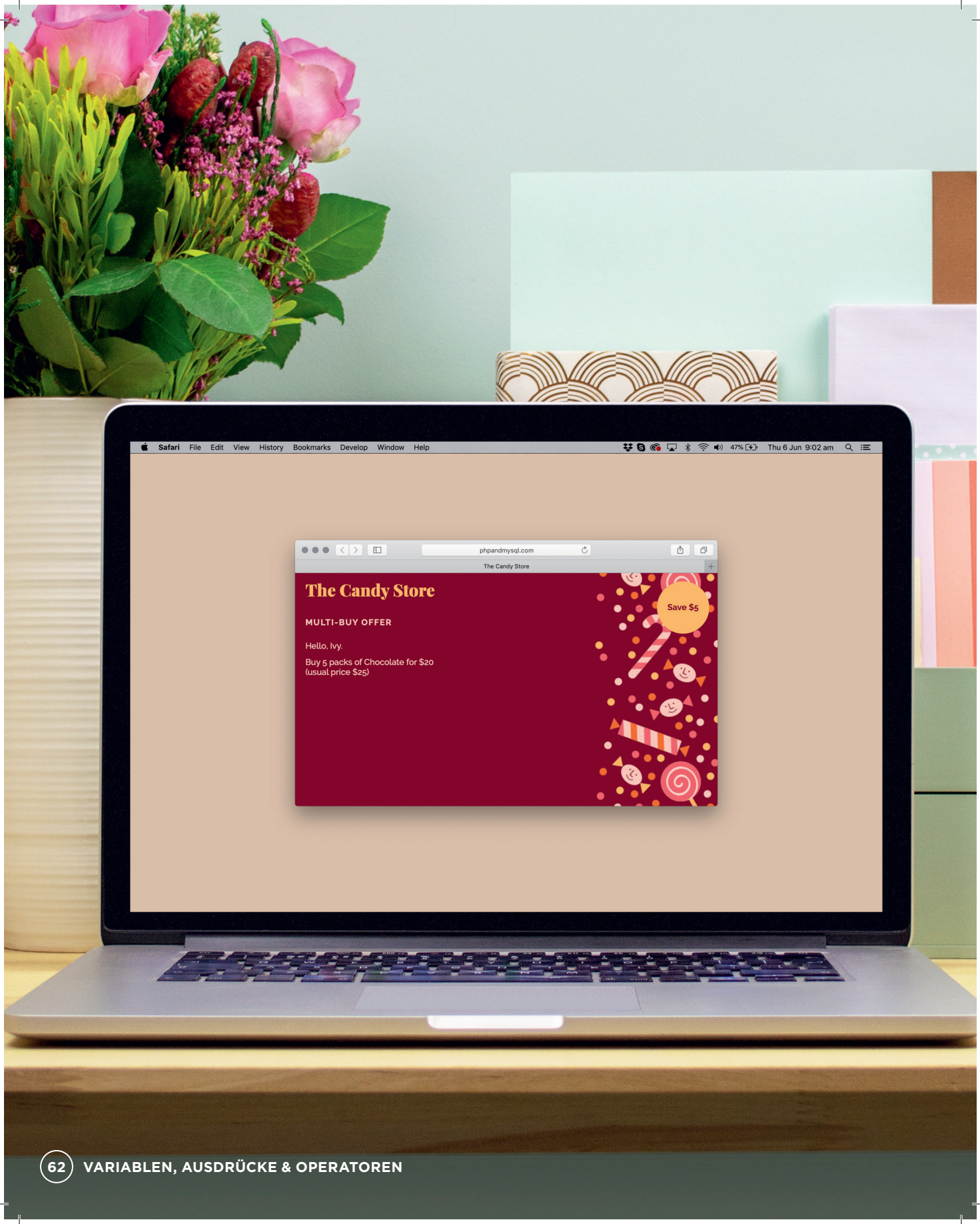
STRING . BOOLESCHER WERT	BEHANDELT ALS	ERGEBNIS	BESCHREIBUNG
`'Hi ' . true`	`'Hi ' . '1'`	`Hi 1` (string)	Boolescher Wert `true` wird als Integer-Wert 1 behandelt.
`'Hi ' . false`	`'Hi ' . ''`	`Hi` (string)	Boolescher Wert `false` wird als leerer String behandelt.

WERT	DATENTYP	BEHANDELT ALS
`false`	Boolean	`false`
`0`	Integer	`false`
`0.0`	Float	`false`
`'0'`	String mit Wert 0	`false`
`''`	Leerer String	`false`
`array[]`	Leeres Array	`false`
`null`	Null	`false`

BOOLESCHE WERTE

Wenn der PHP-Interpreter einen booleschen Wert erwartet, werden alle in der Tabelle links aufgeführten Werte als `false` behandelt.

Sämtliche anderen Werte werden als `true` behandelt (jeglicher Text, jede Zahl ungleich 0 oder der boolesche Wert `true`).


Safari File Edit View History Bookmarks Develop Window Help
47% Thu 6 Jun 9:02 am
phpandmysql.com
The Candy Store
The Candy Store
MULTI-BUY OFFER
Hello, Ivy.
Buy 5 packs of Chocolate for $20
(usual price $25)
Save $5

EINE EINFACHE PHP-SEITE

Dieses Beispiel fasst mehrere der Techniken zusammen, die Sie in diesem Kapitel kennengelernt haben.
Die PHP-Datei erzeugt eine HTML-Seite, die über einen Rabatt beim Kauf mehrerer Päckchen Süßigkeiten informiert.

Sie werden sehen, wie Sie:

- Informationen in Variablen und Arrays speichern.
- den Verknüpfungsoperator verwenden, um Texte in Variablen zu verbinden und so eine personalisierte Begrüßung zu erstellen.
- arithmetische Operatoren einsetzen, um die auf der Seite angezeigten Preise zu berechnen.
- die vom PHP-Interpreter neu erstellten Werte in den HTML-Bereich der Seite schreiben.

Zudem gibt die Seite automatisch die neuen Produkte und Preise wieder, wenn die in den Variablen gespeicherten Werte aktualisiert werden.

DATEN VERARBEITEN UND ANZEIGEN

Wenn Sie anfangen, PHP-Dateien zu schreiben, enthalten diese oft eine Mischung aus HTML- und PHP-Code. Es empfiehlt sich dabei, diesen Code so weit wie möglich zu trennen:

- Fangen Sie mit PHP an, um die Werte zu erzeugen, die auf der HTML-Seite angezeigt werden sollen, und speichern Sie diese in Variablen. (Rechts entspricht dies dem Bereich oberhalb der gestrichelten Linie.)
- Der untere Teil der Seite kann sich dann auf den HTML-Inhalt konzentrieren. PHP-Code sollte hier lediglich zur Anzeige der in den Variablen abgelegten Werte verwendet werden. (Rechts entspricht dies dem Bereich unterhalb der gestrichelten Linie.)

Sehen wir uns den PHP-Code am Anfang der Seite an:

1. Dieses Beispiel beginnt mit der Deklaration einer Variablen zur Speicherung des Benutzernamens. Sie heißt `$username`, weil ein Variablenname immer mit einem Dollarzeichen beginnen sollte, gefolgt von einem aussagekräftigen Namen, der die Art der darin enthaltenen Daten beschreibt.

2. Hier wird die Variable `$greeting` deklariert, in der eine Begrüßungsfloskel hinterlegt wird. Diese wird mithilfe des String-Operators aus der Zeichenkette `Hello` und dem Benutzernamen zusammengesetzt.

3. Die Variable `$offer` wird angelegt, um die Details eines im Sonderangebot befindlichen Artikels zu speichern. Als Wert enthält sie ein Array mit vier Elementen:

- der angebotene Artikel
- die erforderliche Abnahmemenge
- der normale Packungspreis (ohne Rabatt)
- der ermäßigte Packungspreis

Das erste Element, das den angebotenen Artikel beschreibt, ist vom Datentyp String. Die anderen Werte sind ganzzahlige Integer.

4. Die Variable `$usual_price` wird angelegt.

Ihr Wert ist der Artikelpreis ohne Rabatt.

Er wird durch die Multiplikation zweier im Array gespeicherter Werte berechnet: der Menge und dem Preis.

5. Die Variable `$offer_price` wird angelegt.

Ihr Wert ist der rabattierte Preis der Artikel. Dieser wird durch Multiplikation der Menge und des ermäßigten Packungspreises (beides im Array gespeichert) berechnet.

6. Die Variable `$saving` wird angelegt, um die Gesamtersparnis zu speichern. Diese berechnet sich durch Subtraktion des in der Variablen `$offer_price` (erstellt in Schritt 5) gespeicherten Werts von dem in `$usual_price` (erstellt in Schritt 4) gespeicherten Wert.

In der zweiten Hälfte der Seite (unterhalb der gestrichelten Linie) wird das HTML generiert, das an den Browser zurückgesendet wird. Sie beginnt mit der Deklaration des HTML-DOCTYPE. PHP dient hier nur noch dazu, Werte auszugeben, die in den vorangegangenen Schritten in Variablen gespeichert wurden:

7. Die Begrüßung, also das Wort `Hello`, gefolgt vom Benutzernamen, wird mit der Kurzschreibweise des `echo`-Befehls auf der Seite ausgegeben.

8. Die Gesamtersparnis, die in der (in Schritt 6 erstellten) Variablen `$saving` gespeichert ist, wird in einem gelben Kreis angezeigt. Mithilfe von CSS wird dieser Kreis rechts oben im Browserfenster platziert.

9. Ein neuer Absatz erläutert die Einzelheiten des Angebots.

Er zeigt die Menge und den Namen der Süßigkeit, die der Besucher kaufen muss.

10. Darauf folgen der in `$offer_price` gespeicherte ermäßigte Preis und der in `$usual_price` abgelegte Normalpreis.

PHP section_a/c01/example.php

```php
<?php
$username = 'Ivy';                              // Variable für Benutzernamen

$greeting = 'Hello, ' . $username . '.';        // Begrüßung lautet 'Hello, ' + Benutzername

$offer = [                                      // Array für Angebotsdaten anlegen
    'item'     => 'Chocolate',                  // angebotener Artikel
    'qty'      => 5,                            // Mindestmenge
    'price'    => 5,                            // Normalpreis pro Stück
    'discount' => 4,                            // Angebotspreis pro Stück
];

$usual_price = $offer['qty'] * $offer['price'];         // Normalpreis gesamt
$offer_price = $offer['qty'] * $offer['discount'];      // Angebotspreis gesamt
$saving      = $usual_price - $offer_price;             // Ersparnis gesamt
?>
<!DOCTYPE html>
<html>
  <head>
    <title>The Candy Store</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>

    <h2>Multi-buy Offer</h2>

    <p><?= $greeting ?></p>

    <p class="sticker">Save $<?= $saving ?></p>

    <p>Buy <?= $offer['qty'] ?> packs of <?= $offer['item'] ?>
       for $<?= $offer_price ?><br>(usual price $<?= $usual_price ?>)</p>
  </body>
</html>
```

Probieren Sie es: Ändern Sie in Schritt 1 den Benutzernamen in Ihren Namen um.

Ändern Sie in Schritt 2 die Begrüßung in `Hi` (statt `Hello`).

Ändern Sie in Schritt 3 die Päckchenanzahl im `qty`-Schlüssel des Arrays `$offer` auf 3.

Ändern Sie in Schritt 3 den Süßwarenpreis auf 6.

ZUSAMMENFASSUNG

VARIABLEN, AUSDRÜCKE & OPERATOREN

- Variablen speichern Daten, die sich bei jeder Ausführung eines Skripts verändern können.
- Skalare Datentypen speichern Text, ganze Zahlen, Fließkommazahlen und die booleschen Werte `true` oder `false`.
- Ein Array ist ein zusammengesetzter Datentyp zum Speichern einer Reihe von zusammengehörigen Werten.
- Die einzelnen Einträge in einem Array werden als Elemente bezeichnet. Elemente in assoziativen Arrays verfügen über einen Schlüssel und einen Wert. Elemente in indizierten Arrays haben eine Indexnummer und einen Wert.
- String-Operatoren verknüpfen (verketten) Texte in Zeichenketten.
- Mathematische Operatoren sind für mathematische Berechnungen mit Zahlen zuständig.
- Vergleichsoperatoren vergleichen zwei Werte, um festzustellen, ob sie beide gleich oder einer größer oder kleiner als der andere ist.
- Logische Operatoren verknüpfen die Ergebnisse mehrerer Ausdrücke mit UND, ODER und NICHT.