

## 1 PHP- was ist das überhaupt?

Die erste Version von PHP wurde 1995 vorgestellt. Der dänische Programmierer Rasmus Lerdorf hatte hierfür einige Skripte in der Programmiersprache Perl erstellt. Der Ausgangspunkt für diese Arbeit bestand darin, dass er die Zugriffe auf seinen Online-Lebenslauf protokollieren wollte. Daraus entstand schließlich eine umfangreiche Skriptsammlung, die Lerdorf für die Gestaltung seiner persönlichen Homepage nutzte. Daher gab er ihr den Namen "Personal Home Page Tools" -PHP.

Von diesen Ursprüngen ist außer dem Namen jedoch nicht mehr viel übrig geblieben. Bereits für die zweite Version, die nur kurze Zeit später erschien, überarbeitete Lerdorf das Konzept stark. Statt in Perl programmierte er diese Version nur in C. Bis heute wurden alle weiteren Versionen ebenfalls in C entwickelt.

1997 ändert sich das Entwicklerteam. Das neue Entwicklerteam wies auch der Abkürzung PHP eine neue Bedeutung zu. Dies steht seitdem für "PHP: Hypertext Preprocessor". Damit wollen die Programmierer verhindern, dass der Eindruck entsteht, dass sich die Programmiersprache nur für die persönliche Projekte eignet.

Der große Durchbruch gelang der Version PHP 4. Zu dieser Zeit wuchs das World Wide Web in einem beachtlichen Tempo und für die Gestaltung dynamischer Inhalte war eine geeignete Programmiersprache erforderlich. Da PHP genau auf diese Aufgabe spezialisiert ist, setzte es sich in diesem Bereich als Standard durch und ist seitdem die Grundlage unzähliger Internetauftritte.

Mit der darauffolgenden Version machte der Entwickler zu einer objektorientierten Programmiersprache.

PHP ist auf fast allen Webservern vorinstalliert. Daher eignet sich diese Sprache hervorragend für serverseitig programmierte dynamische Webseiten.

### 1.1 Statische und dynamische Webseiten

Als das World Wide Web 1989 entstand, kamen hierfür einfache HTML-Seiten zum Einsatz. Dabei handelt es sich um eine recht einfache Auszeichnungssprache, die den Inhalten einer Internetseite eine Struktur verleiht. Diese Inhalte bleiben jedoch - solange keine manuelle Anpassung vorgenommen wird - stets gleich. Auf diesen Grund werden reine HTML-Seiten als statisch bezeichnet.

Als sich das Internet jedoch weiterentwickelte, reichten diese statischen Seiten nicht mehr aus. Bei vielen Anwendungen war es notwendig, den Inhalt, der dabei ausgegeben wird, immer wieder an den Nutzer oder an die aktuellen Bedingungen anzupassen. Ein einfaches Beispiel hierfür ist die Wiedergabe von E-Mails in einem Webbrowser.

Bei der Gestaltung dynamischer Webseiten ist es möglich, diese clientseitig oder serverseitig zu programmieren. Bei clientseitigen Anwendungen liefert der Server den kompletten Programmcode aus. Anschließend führt der Browser diesen aus. Damit ist es beispielsweise möglich, die aktuelle Uhrzeit in die Seite einzufügen, dem Anwender einen Rechner für eine spezifische Aufgabe bereitzustellen oder die Eingabe in ein Formular auf ihre Richtigkeit zu überprüfen. Für diese Aufgaben eignet sich beispielsweise JavaScript. PHP ist jedoch eine serverseitige Sprache. Dabei führt der Server bereits

das Programm aus. Das fertige Ergebnis liefert er dann dem Anwender aus. Dieses ist anschließend nicht mehr veränderbar. Diese Methode ist insbesondere dann sinnvoll, wenn für die Inhalte die Informationen aus einer Datenbank notwendig sind. Da diese in der Regel ebenfalls auf dem Server gespeichert ist, bietet sich in diesen Fällen eine serverseitige Umsetzung an.

## 1.2 Anwendermöglichkeiten für PHP

Da es sich bei PHP um die Programmiersprache handelt, die am häufigsten für die serverseitige Programmierung von Internetanwendungen zum Einsatz kommt, gibt es unzählige weitere Programme, die Funktionen bereitstellen, die für viele Websites sinnvoll sind.

## 2 Vorbereitungen für das Programmieren mit PHP

### 2.1 HTML: eine wichtige Grundlage für Programme in PHP

In der Einleitung wurde dargestellt, dass PHP für die Erstellung dynamischer Internetseiten zum Einsatz kommt. Statische Seiten verwenden hingegen HTML. Der folgende Abschnitt befasst sich nun mit genau dieser Auszeichnungssprache. Das stellt jedoch nur auf den ersten Blick einen Widerspruch dar. Denn fast jede Internetseite, die mit PHP erstellt wird, verwendet auch HTML-Code.

Der Zusammenhang besteht darin, dass die Ausgabe eines PHP-Programms normalerweise in HTML erfolgt. Wenn der Server das entsprechende Programm ausführt, erzeugt er eine gewöhnliche HTML-Seite. Diese gibt er daraufhin an den Browser des Nutzers weiter, der sie dann wie eine statische Seite anzeigt. Das dynamische Element liegt hierbei in der Ausführung auf dem Server, bei der individuell angepasste Seiten entstehen. Alle nachfolgenden Schritte sind genau gleich wie bei der Anzeige herkömmlicher Internetseiten.

Eine Internetseite besteht eigentlich aus gewöhnlichen Textbausteinen. Mithilfe von Tags werden diese jedoch verschiedene Funktionen zugewiesen.

### 2.2 Webserver-Software für die Ausführung eines PHP-Programms

Bei PHP-Programme handelt es sich um Skripte. Das bedeutet, dass der Code nicht kompiliert wird und daher nicht als ausführbare Datei vorliegt. Stattdessen ist ein Interpreter notwendig. Dieser liest das Programm, das als reiner Text vorliegt, ein und setzt die Funktion um. Wie in den vorherigen Abschnitten bereits mehrfach erwähnt, werden PHP-Programme auf einem Webserver ausgeführt. Gängige Server-Software verfügt standardmäßig über einen PHP-Interpreter, sodass der Umgang mit diesen Programmen keinerlei Probleme darstellt.

Die bekannteste Software in diesem Bereich trägt den Namen "XAMPP ". Dabei handelt es sich um freie Software, sodass die Nutzung keinerlei Lizenzgebühren anfallen.

## 3 Das erste Programm mit PHP gestalten

### 3.1 PHP-Skripte im Text kenntlich machen

Wenn ein Anwender mit einem Browser ein PHP-Dokument aufruft, muss der Webserver zunächst das entsprechende PHP-Programme ausführen, um die gewünschte Ausgabe zu erstellen. Dafür ist

es jedoch notwendig, dass er den Text als PHP-Programm erkennt. Nur so ist eine sachgemäße Umsetzung möglich. Um dies dem Webserver zu vermitteln, ist zum einen eine passende Dateiendung notwendig. PHP-Programme enden normalerweise auf .php Darüber hinaus gibt es noch weitere Möglichkeiten - beispielsweise .php3 und .phtml. Zwar unterstützen viele Webserver auch diese Endungen. Da sie nicht mehr gebräuchlich sind, ist es jedoch ratsam, sie nicht mehr zu verwenden.

Um eine exakte Trennung zwischen den einzelnen Bereichen zu ermöglichen, ist es erforderlich, die PHP-Skripte im Text zu kennzeichnen. So weiß der Interpreter, an welchen Stellen er zum Einsatz kommen soll.

Für diese Aufgaben kommen Tags zum Einsatz. An der Stelle, an der der entsprechenden Abschnitte beginnt, muss der Tag `<?php` stehen. Danach folgt das Skript. Am Ende steht das Schlusstag `?>`.

### 3.2 Das erste Programm schreiben

Zu diesem Zweck ist es notwendig, zunächst den Texteditor zu öffnen, um die entsprechenden Kommandozeilen einzugeben.

Der erste Code sieht deshalb wie folgt aus:

```
1 <?php
2
3 ?>
```

In den Raum zwischen diesen beiden Tags sollen nun die eigentlichen Funktionen eingefügt werden. Für den Anfang ist es sinnvoll, einen einfachen Befehl zu wählen - beispielsweise `print`. Dieser erzeugt eine Ausgabe auf dem Bildschirm. Dabei ist es möglich beliebige Wörter zu verwenden. Für das erste Programm soll die Ausgabe "Mein erstes Programm " lauten.

Der Text, der dabei erscheinen soll, muss immer in Kodierungszeichen (") stehen. das ist insbesondere dann wichtig, wenn Zahlen ausgegeben werden sollen. Wenn beispielsweise nach dem `print`-Befehl `"2+3 "` steht, gibt das Programm diesen Wert als Zeichenkette aus. Auf dem Bildschirm erscheinen daher genau die gleichen Zahlen: `2+3`. Wenn dieser Ausdruck jedoch ohne Kodierungszeichen eingefügt wird, betrachtet ihn das Programm als eine mathematische Operation. Das bedeutet, dass es zunächst den Wert dieses Terms berechnet: `2+3=5`. Als Ausgabe erscheint lediglich die Zahl 5. Um Fehler zu vermeiden, ist es sehr wichtig, stets auf die richtige Setzung der Kodierungszeichen zu achten.

Nach jedem PHP-Befehl muss ein Semikolon stehen. Dieses beendet die Funktion und ermöglicht, dass das Programm mit der nächsten Aufgabe fortfährt.

Wenn nun der vollständige Befehl zwischen die beiden bereits vorhandenen Tags eingefügt wird, ergibt sich folgender Code:

```
1 <?php
2 print "Mein erstes Programm";
3 ?>
```

Wenn der Server diesen Code ausführt, erscheint lediglich der Titel "Mein erstes Programm " auf dem Bildschirm.

Nun ist es nur noch notwendig, den geschriebenen Code als Datei zu speichern. Der Dateiname darf dabei frei gewählt werden. Wichtig ist es lediglich, die Endung **.php** zu verwenden. Daran erkennt der Server, der das Programm ausführt, dass es sich hierbei um eine PHP-Datei handelt. Bei einigen Texteditor ist es dafür notwendig, die Standard-Optionen - die Speicherung als **txt**-Dokument - beim Abspeichern zu entfernen.

### 3.3 Das Programm zum Laufen bringen

Nun ist das erste Programm zwar bereits fertig, jedoch ist das Ergebnis bislang noch nicht auf dem Bildschirm zu erkennen. Um die Funktionsweise zu überprüfen, ist es jedoch notwendig, das Programm auszuführen. Hierfür kommt die Software XAMPP zum Einsatz.

Um die Software zu starten, ist es notwendig, die Datei xampp-control auszuführen. Diese befindet sich in dem Ordner, der bei der Installation von XAMPP gewählt wurde. Nach einem Doppelklick auf diese Datei erscheint folgendes Fenster:



Nun ist es notwendig, in der mit "Apache " gekennzeichneten Zeile auf den Button "Starten " zu klicken. Das führt dazu, dass der Webserver ausgeführt wird. Es ist nun möglich, das entsprechende Fenster zu schließen oder zu minimieren. Der Prozess findet weiterhin im Hintergrund statt.

Im nächsten Schritt ist es notwendig, das PHP-Programm in den Stammordner von XAMPP zu verschieben. Dabei handelt es sich um den Ordner htdocs. Sollten später weitere Programme hinzukommen, ist es sinnvoll, Unterverzeichnisse einzufügen. Das verbessert die Übersicht deutlich.

Wenn das Programm im richtigen Verzeichnis platziert wurde, ist es erforderlich, es über einen Webbrowser aufzurufen. Dafür ist es notwendig, folgenden Pfad in die Adressleiste einzugeben:

[http://localhost/erstes\\_programm.php](http://localhost/erstes_programm.php).

Häufig kommt es vor, dass der Programmierer einen ersten Entwurf ausführen lässt, danach das Ergebnis im Browser betrachtet und daraufhin einige Änderungen vornimmt. Wenn er nun die neue Version überprüfen will, muss er diese zunächst im Texteditor abspeichern.

### 3.4 Übung: ein einfaches Programm in PHP schreiben

1. Erstellen Sie ein Programm, das die Besucher Ihrer Homepage begrüßt.
2. Zeigen Sie drei verschiedene Möglichkeiten auf, um die Zahl 8 als Ausgabe eines PHP-Programms darzustellen und erklären Sie die Unterschiede.

```
<?php  
    print ("Hallo Besucher")  
?>
```

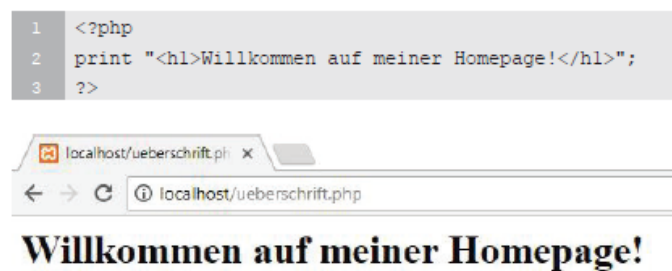
```
<?php  
    print"8";  
    print (4+4);  
    print 8;  
?>
```

## 4 PHP und HTML miteinander verbinden

Bei der Gestaltung von Webseiten mit PHP besteht eine sehr enge Verbindung zwischen der Programmiersprache und dem HTML-Code. Das Ziel besteht in diesem Anwendungsbereich stets darin, mithilfe von PHP korrekte HTML-Seiten auszugeben. Daher ist es vor der Einstellung eines Programms mit PHP stets sinnvoll, sich zu überlegen, wie die Ausgabe aussehen soll. Daher ist es wichtig, die entsprechende Seite in HTML zu gestalten. Anschließend ist es erforderlich, die Stellen, die dynamisch Inhalte verwenden, zu kennzeichnen. Anschließend müssen diese Bereiche durch ein PHP-Skript ersetzt werden. Die kleinen Programmstücke sollen nun Inhalte erzeugen, die an den jeweiligen Kontext angepasst sind. Sie sollen jedoch genau die gleiche Funktion übernehmen, wie die entsprechenden Elemente in der ursprünglichen HTML-Seite.

### 4.1 HTML-Code mit PHP ausgeben

Im ersten Schritt soll dargestellt werden, auf welche Weise es möglich ist, HTML-Code mit PHP zu erzeugen. Hierfür kommt die gewöhnliche print-Funktion zum Einsatz. Dabei kommt jedoch kein einfacher Text zum Einsatz. Dieser muss mit HTML-Tags versehen sein.



Da es sich hierbei um eine Zeichenkette handelt, ist es selbstverständlich notwendig, Kodierungszeichen zu verwenden. Das ist hierbei besonders wichtig, da die spitzen Klammern auch in PHP spezielle Funktionen haben. Sollten diese nicht eindeutig als Teil einer Zeichenkette gekennzeichnet sein, führt das zu Syntaxfehlern, sodass der Server das Programm nicht ausführen kann.

Im Prinzip ist es möglich, alle unterschiedlichen Elemente im gleichen print-Befehl unterzubringen. Auf diese Weise wird der Code jedoch sehr unübersichtlich. Deshalb ist es empfehlenswert, für jeden einzelnen Bereich einen eigenen print-Befehl zu verwenden.

Es ist empfehlenswert, den Befehl `\n` nach jedem HTML-Element einzufügen. Dieser hat keinerlei Auswirkung auf die Darstellung der Seite. Er bewirkt jedoch einen Zeilenumbruch im Quellcode.

### 4.2 PHP-Skripte in eine HTML-Seite integrieren

Eigentlich ist es möglich, auf diese Weise die gesamte Internetseite zu gestalten. Dazu wäre es notwendig, für jedes einzelne HTML-Element eine eigene print-Funktion zu gestalten. Das würde jedoch einen sehr großen Aufwand mit sich bringen, der eigentlich vollkommen unnötig ist.

Das liegt daran, dass selbst bei dynamischen Seiten viele Elemente über die gesamte Webseite hinweg gleich bleiben. Daher ist es möglich diese Bereiche statisch zu gestalten. Hierfür ist es lediglich

notwendig, sie in HTML zu codieren. PHP-Skripte kommen dann nur in den Bereichen zum Einsatz, in denen tatsächlich dynamische Inhalte erzeugt werden.

Es ist ganz einfach HTML und PHP zu mischen. Da der PHP-Code immer durch die entsprechenden Tags ausgezeichnet sein muss, erkennt der Interpreter genau, wann er ein entsprechendes Programm ausführen muss. Dabei ist es auch möglich mehrere PHP-Skripte in das Dokument zu integrieren. Das ist häufig notwendig, da sich bei umfangreichen Inhalten statische und dynamische Elemente abwechseln. Das gilt nicht nur für den body-Bereich, in dem die eigentlichen Inhalte stehen. Auch im head-Bereich gibt es viele Vorgaben, die für die gesamte Webseite gleich sind. Andere Teile wie der Titel und die Beschreibung werden hingegen für jede Seite individuell angepasst. Daher ist hierbei die Verwendung von PHP-Skripten für die dynamische Erzeugung sinnvoll.

```
1 <html>
2 <head>
3 <?php print "<title>Individueller Titel der Seite</title>\n"; ?>
4 <?php print "<meta name=\"description\" content=\"Individuelle
5     Beschreibung der Seite\"/> \n"; ?>
6 <link rel="stylesheet" href="Link zum Stylesheet ist für alle Seiten
7 gleich"/>
8 </head>
9 <body>
10 <h1>Hauptüberschrift für die Website</h1>
11 <?php
12 print "<h2>Artikelüberschrift</h2>\n";
13 print "<p>Textblock</p>\n";
14 ?>
15 <p>Fußzeile</p>
16 </body>
17 </html>
```

## 5 Variablen: ein wichtiges Element der Programmierung mit PHP

Bei Variablen handelt es sich um Platzhalter, die verschiedene Werte annehmen können. Jede Variable hat einen eindeutigen Namen, der einen Zugriff ermöglicht. Als veranschaulichendes Beispiel ist es möglich, sich die Variablen wie Schubladen in einer Kommode vorzustellen. Jede davon ist mit einem eigenen Namen beschriftet. Die Variable bleibt dabei immer gleich, der Inhalt kann sich jedoch ändern.

Es gibt unterschiedlich Typen von Variablen. Meistens handelt es sich um Zeichenketten oder Zahlen. Mit den Inhalten lassen sich - sowohl mit der Kommode als auch bei Variablen verschiedene Operationen durchführen. Mit Variablen die Zahlen enthalten, sind ganz andere Operationen möglich, als wenn sich Zeichenketten darin befinden.

Der Inhalt einer Variable kann sich zwar gelegentlich ändern. Zu einem bestimmten Zeitpunkt ist er jedoch immer genau definiert.

### 5.1 Texte mit einer Variable erfassen

Webseiten bestehen aus einem großen Teil aus einfachem Text. Bei dynamischen Webseiten kommt es in erster Linie darauf an, sich ändernde Textbausteine in die Seite einzufügen. Hinzu kommt selbstverständlich der HTML-Code.

Bei Textvariablen ist es möglich, Inhalte auszutauschen, zu ergänzen oder zu ändern.

Um eine Textvariable zu erstellen und mit Inhalt zu füllen, ist folgender Code notwendig:

```
1 <?php
2 $textbaustein = "Meine erste Variable";
3 ?>
```

Nachdem die Variable eingeführt ist, wird sie sofort mit Inhalt gefüllt. Dazu dient das Gleichheitszeichen. Anschließend folgt der Text, den die Variable enthalten soll. Im Gegensatz zu anderen Programmiersprachen ist es in PHP nicht notwendig, den Typ der Variable anzugeben. Dass es sich hierbei um eine Textvariable handelt, erkennt der Interpreter automatisch daran, dass der Inhalt in Kodierzeichen steht. Variablen können sogar innerhalb der Programms ihren Typ ändern.

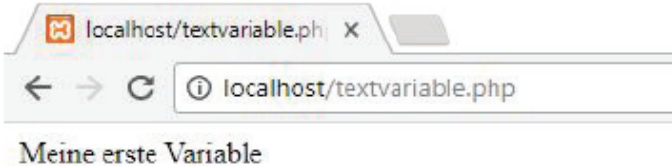
Um den Inhalt der Variable auf dem Bildschirm wiederzugeben, ist es lediglich notwendig, sie in die print-Funktion einzufügen. Dabei ist ebenfalls das Dollarzeichen vor dem Namen der Variablen erforderlich.

### 5.2 Zahlen als Variable abspeichern

Variablen können neben Text auch Zahlen aufnehmen. Es kommt in PHP-Programmen immer wieder vor, dass eine Variable einen numerischen Wert annehmen muss. Insbesondere für die Steuerung von Schleifen, ist diese Funktion sehr wichtig.



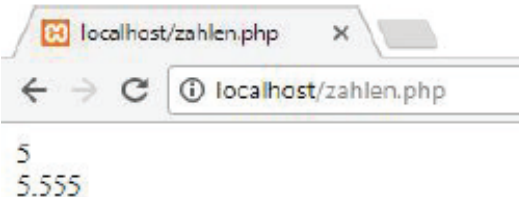
```
1 <?php
2 $textbaustein = "Meine erste Variable";
3 print $textbaustein;
4 ?>
```



Computerprogramme unterscheiden häufig zwischen vielen verschiedenen Zahlentypen. Dabei kann es sich um ganze Zahlen handeln, die als integer bezeichnet werden. Außerdem gibt es Kommazahlen. Jeder Zahlentyp ist dabei in seiner Größe begrenzt, sodass der dafür nötige Speicherplatz fest vorgegeben ist.

Um eine Variable einen numerischen Wert zuzuordnen, ist es lediglich notwendig, sie entsprechend zu deklarieren und mit einem Inhalt zu füllen.

```
1 <?php
2 $ganzezahl = 5;
3 $kommazahl = 5.555;
4 print $ganzezahl ;
5 print "<br>";
6 print $kommazahl;
7 ?>
```



### 5.3 Boolesche Variablen

Neben text und Zahlen gibt es noch eine dritte Art von Variablen. Dabei handelt es sich um sogenannte "boolesche Variablen". Für spätere Aufgaben ist es jedoch wichtig, dass der Leser weiß, dass es diesen Variablentyp gibt und wozu er dient.

Diese Variablen sind nach dem englischen Mathematiker und Logiker George Boole benannt. Eigentlich handelt es dabei um Variablen, die eine endliche Zahl vom Wert annehmen können. Die boolesche Variable kann die Werte true und false - auf Deutsch wahr und falsch - annehmen. Die boolesche Variable dient als Wahrheitswert. Sie kommen in erster Linie zum Einsatz, um bei Funktionen eine Bedingung vorzugeben.

## 5.4 Operationen mit Variablen

Die bisherigen Beispielprogramme, die Variablen verwenden, würden eigentlich auch ohne die Platzhalter auskommen. Wenn die entsprechenden Texte oder Zahlen direkt in die print-Funktion eingefügt werden, ist die Ausgabe genau die gleiche.

In etwas komplizierteren Programmen ist der Einsatz von Variablen jedoch von großer Bedeutung. Zahlreiche Funktionen sind ohne sie nicht möglich. Das liegt daran, dass sich damit viele verschiedene Operationen durchführen lassen. Diese ändern den Wert der Variablen und bieten dadurch viele neue Möglichkeiten für die Gestaltung des Programms.

```
1 <?php
2 $bestand = 5;
3 print "<p>Verfügbare Artikel: ";
4 print $bestand;
5 print "</p>\n";
6 $bestand = 4;
7 print "<p>Verfügbare Artikel: ";
8 print $bestand;
9 print "</p>\n";
10 ?>
```

Diese Programm gibt den Warenbestand für eine bestimmten Artikel an und gibt diesen auf dem Bildschirm aus. Danach wird einer dieser Artikel verkauft, sodass nur noch vier Einheiten verfügbar sind. Anschließend wird der Variablen der neue Wert zugewiesen und danach erfolgt eine erneute Anzeige, um die Ergebnisse deutlich zu machen.

Sinnvoller ist es mit mathematische Operationen durchzuführen. Diese lässt sich mit jedem beliebigen Ausgangswert anwenden und kann daher bei jedem Verkauf zum Einsatz kommen.

```
1 <?php
2 $bestand = 5;
3
4 print "<p>Verfügbare Artikel: ";
5 print $bestand;
6 print "</p>\n";
7
8 $bestand = $bestand-1;
9
10 print "<p>Verfügbare Artikel: ";
11 print $bestand;
12 print "</p>\n";
13
14 ?>
```

Nun ist es möglich, den Ausgangswert der Variablen zu verändern.

Folgende Liste zeigt Ihnen welche mathematische Operatoren es gibt:

- + Addition
- - Subtraktion
- \* Multiplikation
- / Division
- \*\* Potenzen

Da es bei vielen Programmen notwendig ist, den Wert einer Variablen um eine Einheit zu erhöhen oder zu erniedrigen, ist dafür auch eine Kurzform vorhanden. Statt der Zeile `$bestand = $bestand-1;` ist es ganz einfach möglich `$bestand--`; einzufügen. Der Befehl `$bestand++`; erhöht den Wert dieser Variable hingegen.

Für die Bearbeitung von Texten ist insbesondere der Punktoperator von großer Bedeutung. Dieser dient dazu, zwei Textvariablen miteinander zu verbinden. Wenn auf der Seite beispielsweise eine personalisierte Ansprache des Besuchers erscheinen soll, lässt sich dies aus der Anrede dem Namen zusammenfügen:

```
1  <?php
2  $anrede = "Frau";
3  $name = "Müller";
4
5  $leseransprache = $anrede." ".$name;
6
7  print "<h1>Hallo ";
8  print $leseransprache;
9  print "!</h1>\n";
10
11 ?>
```

Die Variable `$leseransprache` entsteht hierbei aus einer Verknüpfung zwischen den Variablen `$anrede` und `$name`. Wichtig ist es dabei, das Leerzeichen zu beachten, das zwischen den beiden Variablen in Kodierungszeichen steht und ebenfalls mit dem Punktoperator verbunden wird. Ohne diesen Zusatz würde das Programm die beiden Wörter miteinander verbinden und ohne das Leerzeichen ausgeben.

## 5.5 Arrays: zusammengesetzte Variablen

In vielen Fällen kommt es vor, dass Variablen in zusammengehörigen Gruppen auftreten. Ein Beispiel hierfür ist das Sortiment eines Onlineshops, das bereits bei der einführenden Erklärung der SQL-Datenbanken herangezogen wurden. Hierbei sind für jedes Produkt die Artikelnummer, der Produktname, der Preis, eine Kurzbeschreibung sowie die Anzahl der verfügbaren Einheit erforderlich.

Da sich alle diese Angaben, wie `$produktname`, `$preis` und `$anzahl`, auf den gleichen Artikel beziehen, ist es daher sinnvoll, sie zu einer Einheit zusammenzuführen. Das sorgt für eine übersichtliche Struktur. Hierfür kommen Arrays zum Einsatz.

Ein Array kann man sich wie eine Werkzeugkiste vorstellen. Diese enthält mehrere unterschiedliche Fächer. In jedem dieser Fächer lässt sich ein unterschiedliches Werkzeug - beziehungsweise im übertragenen Sinne eine Variable - ablegen. Um ein Array anzulegen, ist folgender Code notwendig:

```
1  <?php
2
3  $produkt = array();
4
5  $produkt[0] = 1;
6  $produkt[1] = "Bohrmaschine";
7  $produkt[2] = 45;
8  $produkt[3] = "Kraftvolle Bohrmaschine für Bohr- und
9      Schraubarbeiten";
10 $produkt[4] = 23;
11 ?>
```

In PHP ist es nicht notwendig, die Anzahl der einzelnen Felder anzugeben. Der Programmierer kann daher beliebig viele Einträge hinzufügen. Dies geschieht in den folgenden Zeilen. Dabei wird den einzelnen Felder ein konkreter Wert zugewiesen. PHP erlaubt es, mit jeder beliebigen Indexzahl zu beginnen. Diese müssen außerdem nicht fortlaufend sein. Es ist jedoch gängige Praxis, dem ersten Feld den Index 0 zuzuweisen und dann die einzelnen Felder fortlaufend zu nummerieren.

Um ein Array einzuführen, gibt es noch eine deutlich kürzere und praktischere Methode.

```
1  <?php
2
3  $produkt = array(1, "Bohrmaschine", 45, "Kraftvolle
4      Bohrmaschine für Bohr- und Schraubarbeiten", 23);
5
6  ?>
```

Die Indexnummer wird in diesem Fall fortlaufend vergeben und beginnt mit 0. Um die Ergebnisse zu überprüfen, ist es sinnvoll, den Array auf dem Bildschirm auszugeben.

```
1 <?php
2
3 $produkt = array(1, "Bohrmaschine", 45, "Kraftvolle
4     Bohrmaschine für Bohr- und Schraubarbeiten", 23);
5
6 print $produkt[0]."<br>\n";
7 print $produkt[1]."<br>\n";
8 print $produkt[2]."<br>\n";
9 print $produkt[3]."<br>\n";
10 print $produkt[4]."<br>\n";
11
12 ?>
```



In diesem Beispiel erfolgt der Zugriff auf die einzelnen Felder über eine Indexnummer. Das bringt zahlreiche Vorteile mit sich. Beispielsweise lassen sich auf diese Weise Funktionen erstellen, die mithilfe mathematischer Operationen auf die einzelnen Felder zugreifen. Das Problem besteht jedoch darin, dass diese Werte nicht selbsterklärend sind. Um die Probleme des Vergessens zu vermeiden, ist es auch möglich, hierfür selbsterklärende Bezeichnungen zu verwenden. Diese Form wird als assoziatives Array bezeichnet.

```
1 <?php
2
3 $produkt = array();
4 $produkt['Artikelnummer'] = 1;
5 $produkt['Produktname'] = "Bohrmaschine";
6 $produkt['Preis'] = 45;
7 $produkt['Beschreibung'] = "Kraftvolle Bohrmaschine
8     für Bohr- und Schraubarbeiten";
9 $produkt['Anzahl'] = 23;
10
11 ?>
```

```
1 $produkt = array('Artikelnummer' => 1, 'Produktname' =>
2 "Bohrmaschine", 'Preis' => 45, 'Beschreibung' => "Kraftvolle
3 Bohrmaschine für Bohr- und Schraubarbeiten", 'Anzahl' => 23);
```

Um dieses Array wiederzugeben, ist auch die Ausgabe anzupassen. Das vollständige Programm lautet daher:

```
1  <?php
2
3  $produkt = array('Artikelnummer' => 1, 'Produktname' =>
4  "Bohrmaschine", 'Preis' => 45, 'Beschreibung' => "Kraftvolle
5  Bohrmaschine für Bohr- und Schraubarbeiten", 'Anzahl' => 23);
6  print $produkt['Artikelnummer']."<br>\n";
7  print $produkt['Produktname']."<br>\n";
8  print $produkt['Preis']."<br>\n";
9  print $produkt['Beschreibung']."<br>\n";
10 print $produkt['Anzahl']."<br>\n";
11
12 ?>
```

Darüber hinaus ist es möglich, zusammengesetzte Arrays zu erzeugen. Im hier beschriebenen Beispiel wäre dies beispielsweise sinnvoll, um das gesamte Warensortiment darzustellen. Da dies jedoch sehr umfangreich wäre, soll dieses im Beispiel auf drei Produkte mit jeweils drei Eigenschaften - Artikelnummer, Produktname und Preis -beschränkt werden.

```
1  <?php
2
3  $sortiment = array();
4
5  $sortiment[0]['Artikelnummer'] = 1;
6  $sortiment[0]['Produktname'] = "Bohrmaschine";
7  $sortiment[0]['Preis'] = 45;
8
9  $sortiment[1]['Artikelnummer'] = 2;
10 $sortiment[1]['Produktname'] = "Kreissäge";
11 $sortiment[1]['Preis'] = 79;
12
13 $sortiment[2]['Artikelnummer'] = 3;
14 $sortiment[2]['Produktname'] = "Bandschleifer";
15 $sortiment[2]['Preis'] = 89;
16
17 print $sortiment[0]['Produktname']."<br>\n";
18 print $sortiment[1]['Produktname']."<br>\n";
19 print $sortiment[2]['Produktname']."<br>\n";
20
21 print $sortiment[0]['Preis']."<br>\n";
22 print $sortiment[1]['Preis']."<br>\n";
23 print $sortiment[2]['Preis']."<br>\n";
24
25 ?>
```

## 5.6 Übungen: Umgang mit Variablen

1. Schreiben Sie ein Programm, das zwei numerische Variablen verwendet. Dieses soll die beiden Werte zunächst einzeln ausgeben. Anschließend soll es beide Variablen multiplizieren und das Ergebnis ausgeben.

2. Schreiben Sie ein Programm, das folgenden Liedtext ausgibt:

"Freude, schöner Götterfunken,  
Tochter aus Elysium,  
wir betreten feuertrunken,  
Himmlische, dein Heiligtum. "

Dabei soll jede Zeile in einer Variablen erfasst werden. Verwenden Sie für die Ausgabe nur einen einzigen print-Befehl.

3. Erstellen Sie ein Array, das von drei Kollegen jeweils die Personalnummer, den Vor- und den Nachnamen enthält. Geben Sie daraufhin die entsprechenden Werte auf dem Bildschirm aus. Dabei sollten die zusammengehörigen Werte für jeden Kollegen jeweils in einer Zeile stehen.

```
<?php
$wert1 = 5;
$wert2 = 10;
$ergebnis = $wert1 * $wert2;
print $wert1 . "\n";
print $wert2 . "\n";
print $ergebnis . "\n";
?>
```

```
<?php
$zeile1 = "Freude, schöner Götterfunken,";
$zeile2 = "Tochter aus Elysium,";
$zeile3 = "wir betreten feuertrunken,";
$zeile4 = "Himmlische, dein Heiligtum.";

print $zeile1 . "\n" . $zeile2 . "\n" . $zeile3 . "\n" . $zeile4;
?>
```

```
<?php
$kollegen = [
    [123, "Max", "Muster"],
    [456, "Anna", "Beispiel"],
    [789, "Tom", "Test"]
];

foreach ($kollegen as $k) {
    print $k[0] . " " . $k[1] . " " . $k[2] . "\n";
}
?>
```



## 6 Entscheidungen durch if-Abfragen

### 6.1 Der Aufbau einer if-Abfrage

Eine if-Abfrage beginnt mit dem Schlüsselwort `if`. Daran erkennt der Interpreter, dass er den kommenden Teil nur ausführen soll, wenn eine bestimmte Bedingung erfüllt ist. Diese Bedingung folgt direkt auf den Schlüsselbegriff. Häufig handelt es sich dabei um einen mathematischen Vergleich. Es ist beispielsweise möglich, die Bedingungen vorzugeben, dass eine bestimmte Variable größer oder kleiner sein muss, als ein vorgegebener Wert. Nur falls dies zutrifft, wird der darauf folgende Teil ausgeführt. Eine andere Möglichkeit wäre die boolesche Variable dafür zu verwenden. Der gesamte Bereich, der innerhalb der geschweiften Klammern steht, wird nur ausgeführt, wenn die Bedingung erfüllt ist. Sollte dies nicht der Fall sein, fährt das Programm mit dem ersten Befehl fort.

```
1  if (Bedingung)
2  {
3      In diesem Bereich ist es möglich, beliebige Befehle einzufügen.
4  }
```

### 6.2 Verschiedene Vergleichsoperatoren verwenden

Häufig wird bei der Bedingung jedoch auch ein Vergleich angegeben. dabei kann es sich um mathematische Vergleiche handeln oder um den Inhalt einer bestimmten Zeichenkette. Wenn überprüft werden soll, ob der Wert einer Variable einem vorgegebenen Wert entspricht, ist es notwendig, ein doppeltes Gleichheitszeichen einzufügen:

```
1  if ($var == 5)
2  {
3      print "Der Wert dieser Variable beträgt 5.";
4  }
```

Das einfache Gleichheitszeichen kommt für eine Zuweisung zum Einsatz. Das bedeutet, dass das Programm der Variablen einen neuen Wert zuweist. Abfragen dürfen den Wert der Variablen hingegen nicht ändern. Daher kommt hierbei stets das doppelte Gleichheitszeichen zum Einsatz.

Hierbei ist es nicht nur möglich, einen numerischen Wert zu überprüfen. Darüber hinaus lassen sich auch Zeichenketten überprüfen:

```
1  if ($var == 5)
2  {
3      print "Der Wert dieser Variable beträgt 5.";
4  }
```

Es können auch Größer- Kleinerzeichen zum Einsatz (> und <) kommen.

```
1 if ($produktname == "Bohrmaschine")
2 {
3     print "Bei diesem Produkt handelt es sich um eine Bohrmaschine";
4 }
```

Darüber hinaus gibt es noch einige weitere Vergleichsoperatoren:

!= Führt die Funktion aus, falls die Werte ungleich sind.

<= Führt die Funktionen aus, falls der erste Wert kleiner oder gleich wie der zweite Wert ist.

>= Führt die Funktion aus, falls der erste Wert größer oder gleich wie der zweite Wert ist.

### 6.3 Logische Operatoren in die Abfrage integrieren

Es ist möglich, dass eine Funktion nur dann ausgeführt werden soll, falls gleichzeitig zwei verschiedene Bedingungen erfüllt sind. In anderen Fällen ist es ausreichend, falls eine von mehreren möglichen Bedingungen erfüllt ist. Logische Operatoren dienen dazu, derartige Ausdrücke zu formulieren.

+Wenn zwei Bedingungen gleichzeitig erfüllt sein sollen, dann kommt dafür das logische UND zum Einsatz. Hierfür ist es möglich, einfach das englische Wort "and" zu verwenden. Alternativ kann dazu kann der Programmierer ein doppeltes Ampersand (&&) einfügen

```
1 if ($bestellwert < 20)
2 {
3     print "Sie haben den Mindestbestellwert noch nicht erreicht";
4 }
```

oder

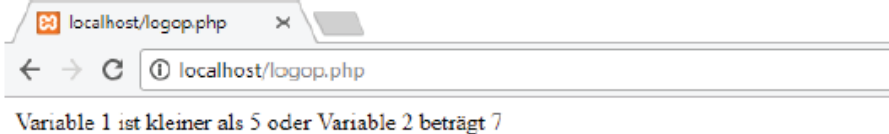
```
1 if ($var1 < 5 and $var2 == 7)
2 {
3     print "Variable 1 ist kleiner als 5 und Variable 2 beträgt 7.";
4 }
```

Dabei ist es nicht nur möglich, zwei Bedingungen vorzugeben, sondern beliebig viele:

```
1 if ($var1 < 5 && $var2 == 7 && $var3 >= 6)
2 {
3     print "Variable 1 ist kleiner als 5, Variable 2 beträgt 7 und
4     Variable 3 ist größer oder gleich 6.";
5 }
```

In anderen Fällen ist es lediglich notwendig, dass eine von zwei Bedingungen erfüllt ist. Hierbei ist es sinnvoll, das logische Oder zu verwenden. Dieses lässt sich mit dem Schlüsselwort "or" oder mit zwei senkrechten Strichen (|) in das Programm einfügen.

```
1 <?php
2
3 $var1 = 2;
4 $var2 = 7;
5
6 if ($var1 < 5 || $var2 == 7)
7 {
8     print "Variable 1 ist kleiner als 5 oder Variable 2 beträgt 7.";
9 }
10
11 ?>
```



Dabei fällt auf, dass der Browser den entsprechenden Text nicht nur anzeigt, falls genau eine der beiden Bedingungen erfüllt ist, sondern auch, wenn dies auf beide Vorgaben zutrifft. In manchen Programmen ist dies jedoch nicht erwünscht. In diesen Fällen kommt das exklusive Oder zum Einsatz. Dieses wird mit dem Schlüsselbegriff "xor " eingefügt. Wenn das Programm entsprechend abgeändert wird, zeigt der Browser den Text nur noch an, falls eine der beiden Bedingungen erfüllt ist, die andere hingegen nicht.

Ein weiterer logischer Operator ist das logische Nicht, das durch ein Ausrufezeichen dargestellt wird. Dieses kehrt den Wahrheitswert der Bedingung um. Das bedeutet, dass das Programm die Funktion nur dann ausführt, falls die entsprechende Bedingung nicht erfüllt ist:

```
1 if (!($var < 5))
2 {
3     print "Die Variable 1 ist nicht kleiner als 5.";
4 }
```

## 6.4 else und elseif

Es kommt jedoch häufig vor, dass in dem Fall, in dem die Bedingung nicht erfüllt ist, eine andere Aktion durchgeführt werden soll. Selbstverständlich ist es hierfür möglich, nach dem ersten Block eine weitere if-Abfrage einzufügen und darin die entsprechenden Bedingungen zu verneinen:

```
1  if ($geschlecht == "m" )
2  {
3  print "<hl>Hallo Herr ".$nachname."</hl>\n";
4  }
5  if (!($geschlecht == "m" ))
6  {
7  print "<hl>Hallo Frau ".$nachname."</hl>\n";
8  }
```

Deutlich einfacher ist es jedoch, den Begriff "else " zu verwenden. Folgendes Programm hat daher genau die gleiche Funktion, wie das oben beschriebene Beispiel:

```
1  if ($geschlecht == "m" )
2  {
3  print "<hl>Hallo Herr ".$nachname."</hl>\n";
4  }
5  else
6  {
7  print "<hl>Hallo Frau ".$nachname."</hl>\n";
8  }
```

Es ist möglich, dass ein Nutzer die Angabe nicht gemacht hat, sodass die Variable leer bleibt. In diesem Fall wäre es nicht empfehlenswert, den Leser automatisch mit "Frau " anzusprechen. Daher ist es sinnvoll, die Abfrage weiter zu differenzieren. Zu diesem Zweck dient eine verzweigte Abfrage:

```
1  if ($geschlecht == "m" )
2  {
3  print "<h1>Hallo Herr ".$nachname."</h1>\n";
4  }
5  else
6  {
7      if ($geschlecht == "w" )
8      {
9          print "<h1>Hallo Frau ".$nachname."</h1>\n";
10     }
11     else
12     {
13         print "<h1>Hallo!</h1>\n";
14     }
15 }
```

Sollte keine Angabe oder ein ganz anderen Wert vorhanden sein, entscheidet sich das Programm für einen neutrale Anrede mit "Hallo ".

```
1  if ($geschlecht == "m")
2  {
3  print "<h1>Hallo Herr " . $nachname."</h1>\n";
4  }
5  elseif ($geschlecht == "w")
6  {
7  print "<h1>Hallo Frau " . $nachname."</h1>\n";
8  }
9  else
10 {
11 print "<h1>Hallo!</h1>\n";
12 }
```

## 6.5 Übungen: Abfragen selbst erstellen

1. Erstellen Sie ein Programm, das ermittelt, ob die Warenbestände für ein bestimmtes Produkt auf 0 gefallen sind und gegebenenfalls eine Meldung ausgibt, dass der Artikel nicht mehr verfügbar ist.
2. Erstellen Sie zunächst ein Array, in dem sich ein Produkt mit dem zugehörigen Preis und der Anzahl der verfügbaren Artikeln befindet.

Schreiben Sie nun ein Programm:

- den Käufer darüber informiert, dass der Artikel nicht verfügbar ist, falls der Warenbestand auf 0 steht.
- ausgibt, dass das Produkte versandkostenfrei geliefert wird, falls mindestens ein Artikel vorrätig ist und falls der Preis bei mindestens 20 Euro liegt.
- anzeigt, dass für die Lieferung 5 Euro Versandkosten anfallen, falls der Artikel verfügbar ist, der Preis jedoch unter 20 Euro liegt.

```
<?php
$bestand = 0;

if ($bestand == 0) {
    print "Artikel nicht mehr verfügbar";
}
else {
    print "Artikel ist verfügbar";
}
?>
```

```
<?php
$produkt = ["Name" => "T-Shirt", "Preis" => 15, "Bestand" => 2];

if ($produkt["Bestand"] == 0 {
    print "Artikel nicht verfügbar";
}
elseif ($produkt["Preis"] >= 20) {
    echo "Produkt wird versandkostenfrei geliefert";
}
else {
    print "Lieferung kostet 5 Euro Versand";
}
?>
```

## 7 Die Funktionalität eines Programms durch Schleifen erweitern

### 7.1 Kopfgesteuerte Schleifen: while und for

Um die Funktion einer Schleife zu erklären, soll hier das Programm lediglich zehn Mal hintereinander das Wort "Hallo " auf den Bildschirm ausgeben.

Hierbei ist es zunächst notwendig, eine Laufbedingung vorzugeben. Dies wird mit dem Schlüsselbegriff `while` eingeführt. Sie steht vor dem eigentlichen Inhalt der Schleife. Aus diesem Grund werden diese als "kopfgesteuert " bezeichnet. Für die Bedingungen kommen die gleichen Operatoren wie bei den `if`-Abfragen zum Einsatz. Wenn die Schleife wie in diesem Beispiel mit einer fest vorgegebenen Anzahl durchlaufen werden soll, kommt in der Bedingung eine Index-Variable zum Einsatz. In der Informatik ist es üblich, diese als `$i` zu bezeichnen und zu Beginn der Schleife den Wert auf 0 zu setzen:

```
1  $i = 0;  
2  
3  while ($i < 10)
```

Die Laufbedingung gibt an, dass die Schleife so lange ausgeführt werden soll, wie der Wert der Index-Variablen kleiner als 10 ist. Damit die Schleife zum gewünschten Zeitpunkt beendet wird, ist es daher notwendig, den Wert dieser Index-Variablen in jedem Durchgang zu erhöhen. Dies geschieht im Schleifenkörper. Dieser schließt sich an die Laufbedingung in geschweiften Klammern an. Außerdem steht im Schleifenkörper die Funktion, die bei jedem Durchgang ausgeführt werden soll.

```
1  <?php  
2  $i = 0;  
3  
4  while ($i < 10)  
5  {  
6  print "<p>Hallo</p>\n";  
7  $i++;  
8  }  
9  
10 ?>
```

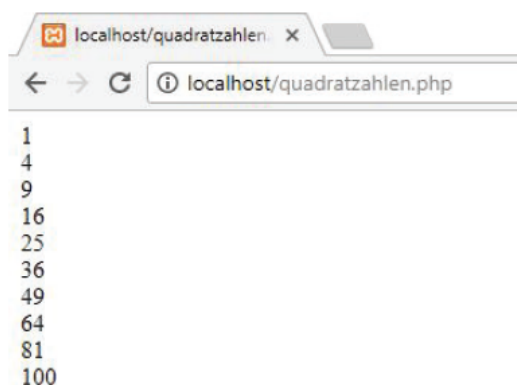
Dabei ist es auch möglich, bei jedem einzelnen Durchgang auf die Indexvariable Bezug zu nehmen. Auf diese Weise lassen sich beispielsweise die Quadratzahlen ganz einfach anzeigen:

```
1  <?php
2  $i = 0;
3  while ($i < 10)
4  {
5  $i++;
6  print $i*$i."<br>\n";
7  }
8
9  ?>
```

Dieses Programm multipliziert in jedem Durchgang die Indexvariable mit sich selbst und erzeugt auf diese Weise das Quadrat dieser Zahl. Da diese Reihe sinnvollerweise mit der Zahl 1 beginnt, muss die Erhöhung der Indexzahl vor dem Befehl für die Ausgabe erfolgen. Alternativ dazu wäre es möglich, wie im vorherigen Programm den print-Befehl voranzustellen, jedoch bei der Berechnung jeweils den Wert 1 zu addieren:

```
1  print ($i+1)*($i+1)."<br>\n";
```

Die Funktionsweise bleibt bei beiden Möglichkeiten die gleiche und erzeugt folgende Ausgabe auf dem Bildschirm:

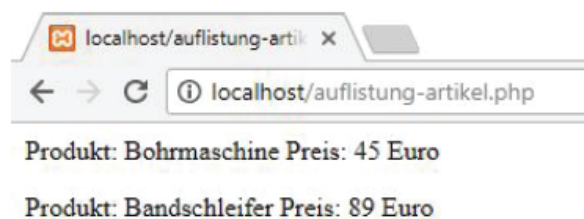


```
1
4
9
16
25
36
49
64
81
100
```

In diesem Fall ist eine if-Abfrage notwendig, in den ein Schleifenkörper integriert wird. Zur Erklärung soll das folgende Beispiel dienen. Dabei ist ein zusammengesetzter Array vorhanden, der für drei unterschiedliche Produkte den Produktnamen, den Preis und die Anzahl der verfügbaren Einheiten enthält. Die Schleife soll daraufhin alle Produkte mit dem zugehörigen Preis auflisten - aber nur, wenn mindestens ein Artikel vorrätig ist. In diesem Fall ist es möglich, auf die einzelnen Felder des Arrays mit einer Variablen zuzugreifen. Dafür bietet sich die Index-Variable an.



```
1 <?php
2
3 $sortiment = array();
4
5 $produkt[0]['Produktname'] = "Bohrmaschine";
6 $produkt[0]['Preis'] = 45;
7 $produkt[0]['Anzahl'] = 6;
8 $produkt[1]['Produktname'] = "Kreissäge";
9 $produkt[1]['Preis'] = 79;
10 $produkt[1]['Anzahl'] = 0;
11
12 $produkt[2]['Produktname'] = "Bandschleifer";
13 $produkt[2]['Preis'] = 89;
14 $produkt[2]['Anzahl'] = 11;
15
16 $i = 0;
17
18 while ($i < 3)
19 {
20     if ($produkt[$i]['Anzahl'] > 0)
21     {
22         print "<p>Produkt: " . $produkt[$i]['Produktname'] .
23             " Preis: " . $produkt[$i]['Preis'] . " Euro</p>\n";
24     }
25     $i++;
26 }
27
28 ?>
```



Die for-Schleife wird durch den Begriff for eingeführt. Daran schließt sich eine Klammer an, in der zunächst der Wert der Variablen eingeführt wird. Danach folgt die Bedingung und schließlich die Angabe, auf welche Weise die Indexvariable in jedem Durchgang verändert werden soll.

```
1 <?php
2
3 for ($i = 0; $i < 10; $i++)
4 {
5     print "<p>Hallo</p>\n";
6 }
7
8 ?>
```

## 7.2 Fußgesteuerte Schleifen: do while

Das erste Beispiel aus dem vorherigen Kapitel sieht mit einer do-while-Schleife folgendermaßen aus:

```
1  <?php
2
3  $i = 0;
4
5  do
6  {
7      print "<p>Hallo</p>\n";
8      $i++;
9  }
10 while ($i < 10);
11
12 ?>
```

In diesem Beispiel ist die Funktionsweise genau die gleiche wie bei der while- oder der for-Schleife. Das trifft auch auf alle andere Programme zu, in denen die Schleife mit einer fest vorgegebenen Anzahl an Durchläufe ausgeführt werden soll. Welche Methode zum Einsatz kommt, hängt dabei ausschließlich von den Präferenz des Programmiers ab. Allerdings gibt es einen wichtigen Unterschied, der in manchen Programmen die Funktionsweise beeinflussen kann: Bei der for- und bei der while-Schleife wird die Bedingung zu Beginn überprüft. Der Schleifenkörper wird nur dann ausgeführt, wenn die Bedingungen erfüllt ist. Bei der do-while-Schleife werden die Befehle im Schleifenkörper hingegen auf jeden Fall mindestens ein Mal ausgeführt. Erst danach erfolgt die Überprüfung der Bedingung. Sollte sie erfüllt sein, folgt ein weiterer Durchlauf. Ist sie nicht erfüllt, wird die Schleife abgebrochen.

Dieser unterschied sollen die beiden folgenden Programmbeispiele verdeutlichen:

```
1  <?php
2
3  $anzahl = 50;
4  $bedingung = true;
5
6  while ($bedingung)
7  {
8      print "<p>Aktuelle Anzahl: ".$anzahl."</p>\n";
9      $anzahl--;
10     $bedingung = ($anzahl > 20);
11 }
12
13 ?>
```

und

```
1  <?php
2
3  $anzahl = 50;
4  $bedingung = true;
5
6  do
7  {
8  print "<p>Aktuelle Anzahl:". $anzahl. "</p>\n";
9  $anzahl--;
10 $bedingung = ($anzahl > 20);
11 }
12 while ($bedingung);
13
14 ?>
```

Diese Programme geben die Anzahl eines bestimmten Artikel aus. Bei jedem Durchlauf - beispielsweise beim Verkauf einer Einheit - reduziert sich die Anzahl. Wenn die Warenbestände auf 20 sinken, hält das Programm die Schleife an, da die aktuellen Bestände zu niedrig sind. Die Funktionsweise der beiden Alternativen ist hierbei genau die gleiche.

### 7.3 foreach-Schleifen für die Arbeit mit Arrays

Unter der Verwendung der Indexzahl ist es ganz einfach, die einzelnen Felder anzusteuern. das Beispielprogramm funktioniert zwar reibungslos, doch weist es dennoch ein Problem auf. In vielen Onlineshops kommt es häufig vor, dass das Sortiment erweitert oder reduziert wird. In diesem Programm wurde die Anzahl der Durchläufe jedoch von Anfang an auf 3 eingestellt. Wenn sich nun die Anzahl der Artikel ändert, funktioniert das Programm nicht mehr. In diesem Fall wäre es notwendig, jedes Mal, wenn sich der Umfang des Warensortiments ändert, eine manuelle Anpassung vorzunehmen.

Auch bei assoziativen Arrays, die statt mit Index-Zahlen mit Begriffen für die Zuordnung arbeiten, ist der Zugriff auf diese Weise nicht möglich.

Um diese Problem zu vermeiden, ist es sinnvoll, eine foreach-Schleife zu verwenden. Diese ist speziell auf den Umgang mit Arrays ausgelegt. Sie passt die Anzahl der Durchläufe automatisch an die vorhandene Zahl der Einträge an.

```
1 <?php
2
3 $produkt = array(1, "Bohrmaschine", 45, "Kraftvolle Bohrmaschine
4 für
5 Bohr- und Schraubarbeiten", 23);
6
7 foreach ($produkt as $inhalt)
8 {
9     print $inhalt."<br>\n";
10 }
11
12 ?>
```

Dieses Programm bestimmt zunächst ein Array mit verschiedenen Angaben zu einem Produkt. Zu Beginn der Schleife muss der Schlüsselbegriff `foreach` stehen. Danach folgen eine runde Klammer und der Name des Arrays. Anschließend steht ein weiterer Schlüsselbegriff: `as`. Danach ist es notwendig, eine neue Variable zu nennen. Diese dient als Kopie der Inhalt der Arrays innerhalb der Schleife. Wenn nun im Schleifkörper die eben eingeführte Variable verwendet wird, gibt sie in jedem Durchlauf ein Feld des Arrays wieder - so lange, bis alle Einträge aufgeführt wurden.



```
1
Bohrmaschine
45
Kraftvolle Bohrmaschine für Bohr- und Schraubarbeiten
23
```

Die die Funktionsweise der `foreach`-Schleife bereits verstanden haben, dürfen zunächst auch gerne selbst versuchen, die Umwandlung vorzunehmen.

```
1 <?php
2
3 $sortiment = array();
4
5 $produkt[0]['Produktname'] = "Bohrmaschine";
6 $produkt[0]['Preis'] = 45;
7 $produkt[0]['Anzahl'] = 6;
8
9 $produkt[1]['Produktname'] = "Kreissäge";
10 $produkt[1]['Preis'] = 79;
11 $produkt[1]['Anzahl'] = 0;
12
13 $produkt[2]['Produktname'] = "Bandschleifer";
14 $produkt[2]['Preis'] = 89;
```

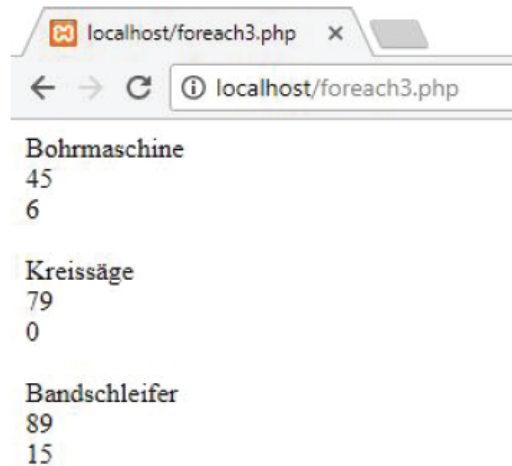
```
15 $produkt[2]['Anzahl'] = 11;
16
17
18 foreach ($produkt as $inhalt)
19 {
20     if ($inhalt['Anzahl'] > 0)
21     {
22         print "<p>Produkt: ".$inhalt['Produktname'].
23             " Preis: ".$inhalt['Preis']. "Euro</p>\n";
24     }
25 }
26
27 ?>
```

Für die Umwandlung ist es notwendig, die Einführung und die Erhöhung der Index-Variablen zu entfernen, da diese bei der Verwendung einer foreach-Schleife nicht mehr notwendig sind.

Foreach-Schleifen erlaubt es auch, ein zusammengesetztes Array auszugeben. Im Gegensatz zum vorherigen Beispiel ist es dabei nicht notwendig, die einzelnen Felder der zweiten Ebene einzeln anzusprechen. Die äußere Schleife bezieht sich dabei auf die erste Ebene. Die Einträge der zweiten Ebene

```
1 <?php
2
3 $sortiment = array();
4 $produkte[0]['Produktname'] = "Bohrmaschine";
5 $produkte[0]['Preis'] = 45;
6 $produkte[0]['Anzahl'] = 6;
7 $produkte[1]['Produktname'] = "Kreissäge";
8 $produkte[1]['Preis'] = 79;
9 $produkte[1]['Anzahl'] = 0;
10 $produkte[2]['Produktname'] = "Bandschleifer";
11 $produkte[2]['Preis'] = 89;
12 $produkte[2]['Anzahl'] = 15;
13
14 foreach ($produkt as $ebene1)
15 {
16     foreach ($ebene1 as $ebene2)
17     {
18         print $ebene2."<br>\n";
19     }
20     print "<br>";
21 }
22 ?>
```

ne sind wie ein weiteres Array für jedes einzelne Feld der ersten Ebene zu verstehen.



#### 7.4 Übung: Programme mit Schleifen gestalten

1. Erstellen Sie drei Programme, die von 1 bis 10 zählen, verwenden Sie hierfür jeweils einmal eine while-, eine for- und eine do-while-Schleife.

2. Erstellen Sie ein zusammengesetztes Array für das Sortiment eines Obst- und Gemüsehändlers mit Äpfeln, Birnen, Tomaten und Zucchini. Die erste Ebene soll einen numerischen Index haben. Die zweite soll hingegen assoziativ sein (mit den Bezeichnungen Produkte, Preis und Sonderangebote). Bei der Angabe zum Produkt handelt es sich um Zeichenkette mit der entsprechenden Obst- oder Gemüsesorte, beim Preis um eine Zahl und die Angabe Sonderangebot soll eine boolesche Variable sein.

Erstellen Sie eine foreach-Schleife, die den Produktnamen und den Preis angibt. Sollte es sich dabei um ein Sonderangebot handeln, soll vor diesen Angaben "Achtung Sonderangebot!" erscheinen.

3. Wenn Sie eine foreach-Schleife erstellen, können Sie bei assoziativen Arrays auch auf den Namen des entsprechenden Feldes zugreifen. Dazu müssen Sie in der runden Klammer nach dem Schlüsselbegriff als eine weitere Variable einführen, die diesen Wert aufnimmt. Darauf folgt das Symbol "=>" und anschließend die Variable, die die Inhalte des Arrays innerhalb der Schleife wiedergibt:

```
($ebene1 as $feldname => $ebene2)
```

Ändern Sie das letzte Programm aus dem Kapitel 7.3 so ab, dass es der Ausgabe der Inhalte den entsprechenden Feldnamen voranstellt.

## 8 Funktionen in PHP

Es kommt vor, dass das eigentliche Hauptprogramm nur aus wenigen Linien besteht. Diese rufen Funktionen auf, die die grundlegende Bestandteile des Programms beinhalten. Jeder einzelne davon verwendet wiederum viele weitere Funktionen, die der Software eine klare Struktur verleihen.

Funktionen bieten Vorteile:

- Programm ist übersichtlicher gestaltet
- Arbeit ersparen
- Fehler können besser vermieden werden
- Funktionsnamen abzurufen

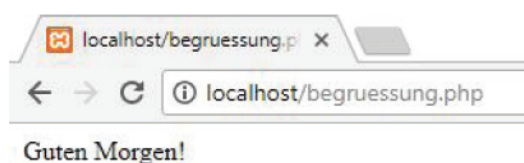
### 8.1 Der Aufbau einer Funktion in PHP

Um eine Funktion zu erstellen, ist es zunächst notwendig, sie im Programm als solche zu kennzeichnen. Hierfür dient der Schlüsselbegriff `function`. Darauf folgt der Name der Funktion. Im Prinzip kann der Programmierer den Namen dabei frei wählen.

```
1  function begruessung ()
2  {
3      print "Guten Morgen!";
4  }
```

Um diese Funktion abzurufen, ist es lediglich notwendig, den Funktionsnamen - gefolgt von der runden Klammer und von einem Semikolon - in das Programm einzufügen.

```
1  <?php
2      function begruessung ()
3      {
4          print "Guten Morgen!";
5      }
6
7      begruessung ();
8
9  ?>
```

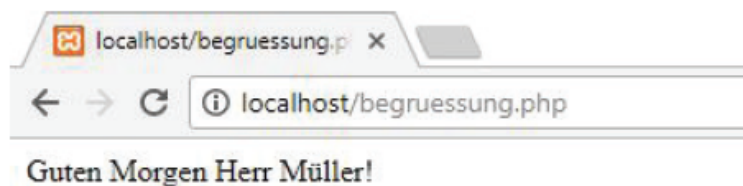


Der Inhalt der Runden Klammern dient dazu, Variablen an die Funktion zu übergeben. Lediglich wenn keine Übergabe von Werten aus dem Hauptprogramm notwendig ist, bleiben sie leer. In allen anderen Fällen muss hier ein Variablennamen stehen. Dieser nimmt den Wert an, den das Hauptprogramm übergibt. Dabei ist es wichtig, diese Variable nur innerhalb der Funktion zu verwenden.

```
1  <?php
2
3  function begruessung ($text)
4  {
5      print $text;
6  }
7
8  begruessung ("Guten Morgen!");
9
10 ?>
```

Dabei kann der Programmierer nicht nur eine einzige Variable übermitteln, sondern beliebig viele. In diesem Fall ist es notwendig, die einzelnen Werte durch ein Komma zu trennen-sowohl bei der Definition der Funktion als auch beim Aufruf.

```
1  <?php
2  function begruessung ($text, $ansprache)
3  {
4      print $text." ".$ansprache."!";
5  }
6
7  $gruss = "Guten Morgen";
8  $leser = "Herr Müller";
9
10 begruessung ($gruss, $leser);
11 ?>
```





## 8.2 Rückgabewerte der Funktion

In den bisherigen Beispielen dient die Funktion dazu, eine bestimmte Aktion durchzuführen. Nachdem sie ausgeführt wurden, war ihre Aufgabe beendet und das Programm nahm keinen weiteren Bezug auf sie. Es gibt jedoch auch viele Anwendungen, bei denen die Funktion einen bestimmten Wert berechnen und diesen an das Hauptprogramm zurückgeben soll. Dieses kann ihn dann speichern und für weitere Aufgaben nutzen. Zu diesem Zweck kommen Rückgabewerte zum Einsatz.

Der Rückgabewert wird durch den Befehl `return` bestimmt. Dieser muss in den Körper der Funktion eingebunden werden.

```
1 function verdopplung ($wert)
2 {
3
4     $wert = $wert*2;
5     return $wert;
6 }
```

Alternativ dazu kann der Programmierer die mathematische Operation direkt hinter dem `return`-Befehl einfügen.

```
1 function verdopplung ($wert)
2 {
3     return ($wert*2);
4 }
```

Um den Rückgabewert im Hauptprogramm aufzunehmen, ist es möglich, einer Variablen den Wert der entsprechenden Funktion zuzuweisen. Das geschieht genau wie bei der Zuweisung gewöhnlicher Werte durch ein Gleichheitszeichen.

```
1 <?php
2 function verdopplung ($wert)
3 {
4     return ($wert*2);
5 }
6 $ergebnis = verdopplung(5);
7 print $ergebnis;
8 ?>
```

Eine weitere Alternative besteht darin, den Rückgabewert direkt in die print-Funktion - oder auch in viele andere PHP-Befehle - zu integrieren:

```
1 <?php
2 function verdopplung ($wert)
3 {
4     return ($wert*2);
5 }
6 print verdopplung(5);
7 ?>
```

Dabei ist es wichtig, zu beachten, dass jede Funktion maximal einen Wert zurückgeben kann. Allerdings kann es sich dabei auch um ein Array handeln. Wenn es erwünscht ist, dass die Funktion mehrere Werte zurückgibt, ist es erforderlich, diese in ein Array zu verpacken. Anschließend ist es möglich, diese wieder aufzulösen und in einzelnen Variablen wiederzugeben.

```
1 <?php
2 function verdopplung_quadrat ($wert)
3 {
4     $doppelt = $wert*2;
5     $quadrat = $wert*$wert;
6     $ergebnis = array ('Verdopplung' => $doppelt, 'Quadrat' =>
7     $quadrat);
8
9     return $ergebnis;
10 }
11
12 $rueckgabewert = verdopplung_quadrat (3);
13 $verdoppelter_wert = $rueckgabewert['Verdopplung'];
14 $wert_zum_quadrat = $rueckgabewert['Quadrat'];
15
16 print "Der doppelte Wert dieser Zahl beträgt " . $verdoppelter_
17 wert."<br>\n";
18 print "Das Quadrat dieser Zahl beträgt " . $wert_zum_
19 quadrat."<br>\n";
20 ?>
```



### 8.3 Eine Funktion in das PHP-Programm einbinden

Um eine Funktion in einem PHP-Programm zu verwenden, ist es erforderlich, sie in das Hauptprogramm einzubinden. Hierfür gibt es verschiedene Möglichkeiten. Eine Alternative wurde bereits in den vorherigen Abschnitten verwendet. Hierbei wurde die Funktion dem eigentlichen Programm vorangestellt. Der Abruf erfolgt ganz einfach über den Funktionsnamen. Wenn bei einer umfangreichen Software mehr als hundert Funktionen zum Einsatz kommen, führt das zu einem sehr langen Programmcode, bei dem es schwierig ist, die einzelnen Elemente zu erkennen.

Bei umfangreicheren Anwendungen ist es üblich, die Funktion in eigene Dateien auszulagern. Das bedeutet, dass hierfür eine neue Datei erstellt werden muss. Um darauf zuzugreifen, ist es notwendig, die entsprechenden Dateien vor dem Aufruf der Funktion in das Hauptprogramm einzubinden.

Um diese auszulagern, ist es erforderlich, zunächst eine neue Datei zu erstellen. Um diese auszulagern, ist es erforderlich, zunächst eine neue Datei zu erstellen. Um die Zuordnung zu erleichtern, sollte der Dateiname dabei dem Funktionsnamen entsprechen. Auch hierbei kommt die Endung `.php` zum Einsatz.

```
1 <?php
2 function verdopplung ($wert)
3 {
4     return ($wert*2);
5 }
6 ?>
```

Bevor die Funktion im Hauptprogramm zum Einsatz kommen kann, ist es nun notwendig, die entsprechende Datei einzubinden. Dafür kommen zwei Befehle infrage: `include` und `require`. Der Unterschied zwischen beiden Befehlen ist jedoch minimal und zeigt sich lediglich, wenn eine Datei nicht geladen werden kann. Bei der Verwendung von `include` fährt das Programm in diesem Fall fort und arbeitet alle weiteren Befehle ab – auch ohne die Inhalte der Datei. Sollte `require` zum Einsatz kommen, bricht es die Ausführung hingegen sofort ab.

Bei beiden Möglichkeiten ist es möglich, den Zusatz `_once` hinzuzufügen. Das bewirkt, dass der Interpreter kontrolliert, ob die entsprechende Datei bereits in einem vorherigen Teil abgerufen wurde. Ist dies der Fall, lädt er die Datei nicht erneut. Das ist insbesondere bei umfangreicher Software sinnvoll, wenn der Programmierer nicht mehr genau weiß, ob er die entsprechende Datei bereits geladen hat. Indem die Dateien nicht doppelt geladen werden, ist es möglich, die Ausführungsgeschwindigkeit zu erhöhen.

Nach dem entsprechenden Befehl ist es notwendig, den Dateinamen einzufügen. Dieser steht in runden Klammern und die Kodierungszeichen. Sollte sich die Datei in einem anderen Ordner als das Hauptprogramm befinden, ist es wichtig, darauf zu achten, zusätzliche den Pfad anzugeben. Das Hauptprogramm sieht daher folgendermaßen aus:

```
1 <?php
2 include ("verdopplung.php");
3 print verdopplung(3);
4 ?>
```



## 9 Objektorientierte Programmierung: Klassen, Objekte und Methoden

Im Mittelpunkt der objektorientierten Programmierung steht das Objekt. Dabei handelt es sich um ein Element, das spezifische Eigenschaften aufweist. Es ist möglich, sich diese Objekte wie verschiedene Alltagsgegenstände vorzustellen. Bei Objekten eines unterschiedlichen Typs sind hingegen auch verschiedene Eigenschaften von Bedeutung. Von großer Bedeutung für die objektorientierte Programmierung sind außerdem Klassen. Diese geben die grundlegenden Eigenschaften vor, die bei der Beschreibung eines Objekts beachtet werden müssen. Darüber hinaus gibt es Methoden. Diese dienen dazu, verschiedene Aktionen mit den Objekten durchzuführen.

### 9.1 Die Klasse: Grundlage der objektorientierten Programmierung

Zu Beginn ist es daher notwendig, sich zu überlegen, welche Merkmale für eine bestimmte Anwendung von Bedeutung sind. Wenn ein Programmierer beispielsweise eine Plattform erstellt, auf der gebrauchte Autos zum Verkauf angeboten werden, muss er sich überlegen, welche Angaben dafür notwendig sind. Wenn er daraufhin ein Objekt für ein bestimmtes Auto erstellt, muss er diesem alle wichtigen Eigenschaften zuweisen.

An dieser Stelle kommen die Klassen ins Spiel. Dabei handelt es sich um ein Muster, an dem sich alle Objekte orientieren. In diesem Beispiel gibt die Vorgänge genau vor, welche Eigenschaften bei einem Auto beschrieben werden sollen. Die Variablen, die die entsprechenden Werte beinhalten, werden als "Member" bezeichnet. Wenn nun ein Objekt, das ein Auto symbolisiert, entstehen soll, muss er diese Vorgabe genau beachten. Das bedeutet, dass hierbei für alle Eigenschaften, die in der Klasse definiert sind, ein Wert vorhanden sein muss.

```
1 <?php
2 class Auto
3 {
4     private $sitzeplaetze;
5     public $geschwindigkeit;
6     public $kraftstoffverbrauch;
7 }
8 ?>
```

Bei den einzelnen Members handelt es sich um Variablen. Daher ist es auch hierbei notwendig, ein Dollarzeichen vor den entsprechenden Namen zu stellen. Vor der Angabe der entsprechenden Eigenschaften steht entweder die Bezeichnung public oder private. Diese Vorgaben vergeben die Zugriffsrechte auf die entsprechenden Eigenschaften. Steht die Bezeichnung public vor der Bezeichnung, ist es möglich, von außerhalb der Klasse-also aus dem Hauptprogramm-auf die entsprechenden Werte zuzugreifen und sie zu verändern. Steht hingegen der Begriff private vor der Eigenschaft, ist das nicht möglich. Das schützt die Variable. In diesem Fall ist es lediglich möglich, aus der gleichen Klasse auf die Variablen zuzugreifen.

Es ist auch möglich, die Variablen bereits in der Klasse zu initialisieren.

```
1 <?php
2 class Auto
3 {
4     private $sitzeplaetze = 5;
5     public $geschwindigkeit;
6     public $kraftstoffverbrauch;
7 }
8 ?>
```

## 9.2 Mit einer Klasse ein Objekt erzeugen

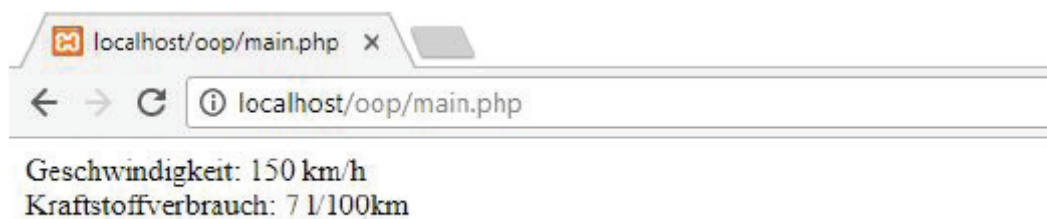
Nachdem die Klasse definiert wurde, ist es möglich, ein Objekt mit den entsprechenden Eigenschaften zu erzeugen. Da es üblich ist, die Beschreibung der Klasse in eine eigene Datei auszulagern, ist es hierfür zunächst notwendig, die entsprechende Datei einzubinden. Bei der Verwendung von Klassen ist die Vorgehensweise identisch. Sollte die Klasse in der gleichen Datei wie das Hauptprogramm definiert werden, ist dieser Schritt nicht notwendig. Wenn die entsprechenden Informationen bereitstehen, muss sich der Programmierer einen Namen für das entsprechende Objekt überlegen.

```
1 $meinAuto = new Auto();
```

Danach ist es möglich, auf die einzelnen Member des Objektes zuzugreifen.

Um auf die Inhalte zuzugreifen, ist es notwendig, den Namen des Objekts mit vorangestelltem Dollarzeichen einzufügen. Danach folgt das Symbol "->". Schließlich ist der Name des Members erforderlich - in diesem Fall jedoch ohne das Dollarzeichen.

```
1 <?php
2 include ("class_auto.php");
3 $meinAuto = new Auto();
4
5 $meinAuto->geschwindigkeit = 150;
6 $meinAuto->kraftstoffverbrauch = 7;
7
8 print "Geschwindigkeit: ".$meinAuto->geschwindigkeit." km/
9 h<br>\n";
10 print "Kraftstoffverbrauch: ".$meinAuto->kraftstoffverbrauch."
11 l/100km<br>\n";
12 ?>
```



Bei dem Befehl `$meinAuto->sitzplaetze = 2;` zum Einsatz. Dabei erscheint diese Fehlermeldung.



Der Grund, dafür besteht darin, dass dieser Member als privat deklariert wurde. Das bedeutet, dass ein Zugriff aus dem Hauptprogramm nicht möglich ist.

### 9.3 Methoden für die Arbeit mit Objekten

Um mit Objekten zu arbeiten, kommen häufig Methoden zum Einsatz. Diese sind sehr ähnlich aufgebaut wie Funktionen, die bereits im vorhergehenden Kapitel beschrieben wurden. Der Unterschied besteht darin, dass sich Methoden lediglich auf Objekte der entsprechenden Klasse anwenden lassen. Methoden erlauben es, auf private Eigenschaften des Objekts zuzugreifen.

Im vorherigen Abschnitt wurde gezeigt, dass der Zugriff auf die private Eigenschaft `sitzplaetze` nicht direkt aus dem Hauptprogramm möglich ist. Das folgende Beispiel soll zeigen, wie der Programmierer den Wert durch eine Methode anpassen kann.

Dies Methode steht direkt in der Definition der Klasse. Daher hat sie Zugriff auf alle Member - selbst auf die privaten. Eine Methode ist praktisch gleich aufgebaut, wie eine Funktion. Sie wird auch mit dem gleichen Schlüsselbegriff eingeführt. Der einzige Unterschied besteht darin, dass es auch hierbei notwendig ist, zu definieren, ob die Methode öffentlich oder privat ist.

```
1 public function setSitzplaetze ($anzahl)
2 {
3     $this->sitzplaetze = $anzahl;
4 }
```

Um innerhalb einer Klasse auf eines ihrer Member zuzugreifen, ist es notwendig das Schlüsselwort `$this` gefolgt vom Symbol `"->"` und dem Namen des Members zu verwenden. Die Funktion muss öffentlich sein, damit das Hauptprogramm darauf zugreifen kann. Beim Aufruf der Methode überprüft das Programm den Wert, der eingefügt werden soll. Dieser wird in der Variable `$anzahl` festgehalten. Wenn eine Methode ein Member verändert, ist es üblich, den Funktionsnamen aus dem Bestandteil `set` und dem Namen des Members zusammenzusetzen. Wenn sie hingegen einen Wert abfragt, ist es gebräuchlich, den Namen aus dem Begriff `get` und dem Namen der Eigenschaft zusammenzusetzen.

Wenn die entsprechende Methode in die Klasse integriert ist, lässt sie sich aus dem Hauptprogramm aufrufen. Dabei ist es notwendig, den Objektnamen und das Symbol `"->"` vor den Namen der Funktion zu setzen: `$meinAuto->setSitzplaetze (2);`

Die vollständige Klasse sieht demnach folgendermaßen aus:

```
1  <?php
2  class Auto
3  {
4      private $sitzplaetze = 5;
5      public $geschwindigkeit;
6      public $kraftstoffverbrauch;
7
8      public function setSitzplaetze ($anzahl)
9      {
10         $this->sitzplaetze = $anzahl;
11     }
12 }
13 ?>
```

Für das Hauptprogramm kommt folgender Code zum Einsatz:

```
1  <?php
2  include ("class_auto.php");
3
4  $meinAuto = new Auto();
5
6  $meinAuto->geschwindigkeit = 150;
7  $meinAuto->kraftstoffverbrauch = 7;
8
9  print "Geschwindigkeit: ".$meinAuto->geschwindigkeit." km/
10 h<br>\n";
```



```
11 print "Kraftstoffverbrauch: ".$meinAuto->kraftstoffverbrauch."  
12 1/100km<br>\n";  
13  
14 $meinAuto->setSitzplaetze (2);  
15 ?>
```

## 9.4 Übungen: Klassen, Objekte und Methoden anwenden

1. Erzeugen Sie eine Klasse für Produkte, die es ermöglicht, diese Eigenschaften festzulegen. Verwenden Sie für die Namen der einzelnen Member die Bezeichnungen aus dem assoziativen Array, der im weiteren Verlauf des entsprechenden Kapitels beschrieben wird. Verhindern Sie dabei einen Zugriff von außerhalb der Funktion.
2. Erstellen Sie Methoden, die es ermöglichen, Änderungen an den einzelnen Members vorzunehmen. Erstellen Sie ein Hauptprogramm, das das Objekt Bohrmaschine erzeugt und die Eigenschaften wie im Array vorgibt.
3. das letzte Beispiel aus diesem Kapitel ermöglicht es zwar, die Anzahl der Sitzplätze zu ändern. Allerdings lässt sich der Wert noch nicht anzeigen. Erstellen Sie hierfür die Methode `getSitzplaetze`. Wenden Sie diese Methode im Hauptprogramm an.

## 10 Dateien für die Speicherung von Daten

Wenn ein PHP-Programm ausgeführt wird, werden alle benötigten Daten in Variablen gespeichert. Das bringt jedoch das Problem mit sich, dass die Informationen verloren gehen, sobald das Programm beendet ist. In vielen Fällen ist es jedoch notwendig, die Dateien auch weiterhin zu verwenden. daher ist erforderlich, sie auf eine andere Weise abzuspeichern. Zu diesem Zweck bieten sich Dateien an. Diese können viele unterschiedliche Werte aufnehmen.

### 10.1 Daten aus einer Datei einlesen

Bevor es möglich ist, eine Datei einzulesen, ist es notwendig, diese zu erstellen. Dazu dient ebenfalls der Texteditor. Für dieses Bsp. soll eine Datei erstellt werden, die die Zahlen von 1 bis 10 enthält - jeweils in einer eigenen Zeile. Sie soll unter dem Namen `beispiel.txt` gespeichert werden.

Nun soll das eigentliche Programm erstellt werden. Dafür kommt der Befehl `fopen` zum Einsatz. Diesem folgt eine runde Klammer, in der zuerst der Dateiname in Kodierungszeichen steht. Nach einem Komma muss der Programmierer - ebenfalls in Kodierungszeichen - den Modus angeben. Hierbei ist es möglich, den Wert `r` zu verwenden. Dieser erlaubt es lediglich, die Datei zu lesen. Eine Alternative dazu stellt die Kombination `w+` dar. So erhält das Programm sowohl die Rechte zum Lesen als auch zum Schreiben. Die Funktion erzeugt als Rückgabewert ein sogenanntes "Handle ". Dieses bietet im Programm zugriff auf die Datei und muss in einer Variable gespeichert werden. Es ist üblich, hierfür den Variablennamen `$handle` oder `$fh` (für "file handle") zu verwenden.

```
1 $handle = fopen("beispiel.txt","r");
```

Da es manchmal beim Öffnen einer Datei zu Fehler kommt, ist es sinnvoll, zunächst eine `if`-Abfrage einzubauen. Diese sorgt dafür, dass das Programm nur ausgeführt wird, wenn der Vorgang erfolgreich war - also wenn die Variable `$handle` existiert. Ist dies nicht der Fall, entsteht eine Fehlermeldung:

```
1 if ($handle)
2 {
3     Hier stehen die Befehle für den Umgang mit der Datei
4 }
5 else
6 {
7     print "Die Datei konnte nicht geöffnet werden.<br>\n";
8 }
```

Nun ist es an der Zeit, die Datei einzulesen. Dazu dient der Befehl `fgets`. Aus der Funktionsweise des `fgets`-Befehls geht hervor, dass dieser jede Zeile einzeln einliest. Das bedeutet, dass hierfür eine `while`-Schleife notwendig ist, die diesen Vorgang so oft wiederholt, bis das Ende des Dokuments erreicht ist.

```
1  <?php
2  $handle = fopen("beispiel.txt","r");
3
4  $ergebnis = array();
5
6  if ($handle)
7  {
8      $i = 0;
9      while (!feof($handle))
10     {
11         $inhalt = fgets($handle);
12         $ergebnis[$i] = $inhalt;
13         $i++;
14         print $inhalt."<br>\n";
15     }
16
17     fclose($handle);
18 }
19
20 else
21 {
22     print "Die Datei konnte nicht geöffnet werden.<br>\n";
23 }
24 ?>
```

## 10.2 Daten in einer Datei speichern

Zu Beginn ist es notwendig, die Datei zu öffnen. Das geschieht genau wie beim Einlesen der Daten über den Befehl `fopen`. Hierbei ist es wichtig, auf den passenden Modus zu achten. Dafür bieten sich die Bezeichnungen `w` zum reinen Schreiben und `w+` sowohl zum Lesen als auch zum Schreiben an. Bei diesem Befehl wird eine eventuell bereits bestehende Datei ersetzt. Sollen die Daten hingegen an die bestehenden Inhalt angehängt werden, ist der Modus `a` (für "append") zu verwenden.

Um die Daten in die Datei zu schreiben, kommt der Befehl `fputs` zum Einsatz. In der Klammer hinter dem Befehl muss zunächst der Name des Handles stehen.

```
1 <?php
2 $handle = fopen("beispiel2.txt","w");
3
4 if ($handle)
5 {
6     for ($i=0;$i<10;$i++)
7     {
8         fputs($handle, ($i+1) . "\n");
9     }
10
11     fclose($handle);
12 }
13
14 else
15 {
16     print "Die Datei konnte nicht geöffnet werden.<br>\n";
17 }
18 ?>
```

Bei vielen Internetanwendungen ist die Interaktion mit den Nutzern sehr wichtig. Hierfür dienen HTML-Formulare. Das HTML-Formular soll folgendermaßen aufgebaut sein:

```
1 <form method="post" action="datenabfrage.php">
2     Name: <input type="text" name="name"><br>
3     E-Mail: <input type="text" name="e-mail"><br>
4     <input type="submit" value="Senden">
5 </form>
```

Hierbei ist wichtig, als Methode immer post anzugeben. Im action-Attribut steht der Dateiname, der für dieses Programm verwendet werden soll. das führt dazu, dass nach dem Absenden die gleiche Seite erneut erscheint.

Um auf die Werte, die der Nutzer eingegeben hat, zuzugreifen, ist der Ausdruck `$_REQUEST[name]` notwendig. In der eckigen Klammer steht dabei der Name, der den einzelnen Formularfeldern zugewiesen wurde.

```
1 <form method="post" action="datenabfrage.php">
2     Name: <input type="text" name="name"><br>
3     E-Mail: <input type="text" name="e-mail"><br>
4     <input type="submit" value="Senden">
5 </form>
6
7 <?php
8
9 $handle = fopen("beispiel3.txt","w");
```

```
10 if ($handle)
11 {
12     if (!empty($_REQUEST['name']) && !empty($_REQUEST['e-mail']))
13     {
14         if ($_REQUEST['name'] != "")
15         {
16             $name = $_REQUEST['name'];
17             fputs($handle, $name."\n");
18         }
19         if ($_REQUEST['e-mail'] != "")
20         {
21             $email = $_REQUEST['e-mail'];
22             fputs($handle, $email."\n");
23         }
24     }
25     fclose($handle);
26 }
27 else
28 {
29     print "Die Datei konnte nicht geöffnet werden.<br>\n";
30 }
31 }
32 ?>
```



Dieses Programm schreibt die Werte, die der Nutzer über die Eingabefelder eingibt, in eine neue Datei mit dem Namen beispiel3.txt. Diese entstehen im gleichen Ordner, in dem sich auch das Programm befindet. Erklärungsbedürftig ist noch die if-Abfrage mit der Bedingung (!empty(\$\_REQUEST['name'])) && !empty(\$\_REQUEST['e-mail'])). Dies sorgt dafür, dass der folgende Teil nur dann ausgeführt wird, wenn der Nutzer bereits einen Wert in das Formularfeld eingegeben hat. Wenn dies noch nicht geschehen ist, sind die Inhalte von \$\_REQUEST['name'] und \$\_REQUEST['e-mail'] noch nicht definiert. Das würde zur Fehlermeldung führen. Außerdem ist die Abfrage mit der Bedingung (\$\_REQUEST['name'] != "") vorhanden. Diese überprüft, ob ein Wert über das Formularfeld eingegeben wurde. Falls das nicht der Fall ist, schreibt das Programm keine Daten in die Datei.

### 10.3 Die Dateirechte beachten

Auf UNIX-artigen Systemen - also auch auf Linux-Webservern - gibt es Zugriffsrechte auf Dateien. Diese sollen verhindern, dass Unbefugte die Datei verwenden, löschen oder ändern. Die Rechte werden dabei in einer Zahl gespeichert: 0 bedeutet, dass überhaupt keine Rechte an der Datei bestehen. Der Wert 1 sagt aus, dass die Datei ausgeführt werden darf. 2 und 4 stehen für Lese- beziehungsweise Schreibrechte. Dabei ist es möglich, die verschiedenen Werte zu addieren. Die Zahl 6 bedeutet daher, dass Lese- und Schreibrechte vorhanden sind. Die Zahl 3 sagt aus, dass der Nutzer die Datei ausführen und lesen darf.

```
1 chmod ("beispiel.txt", 0666);
```

#### 10.4 Übung: Dateien für die Datenspeicherung verwenden

1. Schreiben Sie ein Programm, das ein Formularfeld im Browser anzeigt. Dieses soll den Nutzer dazu auffordern, eine beliebige Zahl einzugeben. Erstellen Sie ein Textdokument mit verschiedenen Zahlen (jeweils in einer eigenen Zeile) und lesen Sie dieses mit Ihrem Programm ein. Multiplizieren Sie daraufhin diese Zahl mit dem Wert, den der Nutzer über den Browser eingegeben hat. Erstellen Sie daraufhin ein neues Textdokument und speichern Sie die Ergebnisse darin ab.