

ipso Bildung AG

In Kooperation mit dem HERDT-Verlag stellen wir Ihnen eine PDF inkl. Zusatzmedien für Ihre persönliche Weiterbildung zur Verfügung. In Verbindung mit dem Programm HERDT-Campus ALL YOU CAN READ stehen diese PDFs für Forschung und Lehre nur Mitarbeiterinnen und Mitarbeitern sowie Studierenden der oben genannten Hochschule zur Verfügung. Eine Nutzung oder Weitergabe für andere Zwecke ist ausdrücklich verboten und unterliegt dem Urheberrecht. Jeglicher Verstoß kann zivil- und strafrechtliche Konsequenzen nach sich ziehen.

PHP 8.0

Stephan Heller

1. Ausgabe, Mai 2021

ISBN 978-3-86249-992-2

Dynamische Webseiten
erstellen

GPHP8



Bevor Sie beginnen ...	4	5.10 Den passenden Array-Typ verwenden	90
		5.11 Weitere Informationen zu Arrays in PHP	90
		5.12 Übungen	92
1. Einführung in PHP	5	6 Mit Formularen arbeiten	94
1.1 PHP-Code in Webseiten	5	6.1 Interaktion mit PHP	94
1.2 Informationen zu PHP	8	6.2 Formulare mit PHP auswerten	97
1.3 PHP-Version 8.0	11	6.3 Übungen	109
2 Grundlegende Sprachelemente	13	7 Funktionen	111
2.1 PHP in HTML einbinden	13	7.1 Funktionen erstellen und aufrufen	111
2.2 Codieren von PHP-Skripten	16	7.2 Mit Funktionen arbeiten	115
2.3 Daten im Browser ausgeben	18	7.3 Der Gültigkeitsbereich von Variablen	127
2.4 Grundlagen zur Fehlersuche in PHP-Skripten	25	7.4 PHP-Dateien einbinden mit <code>include()</code> und <code>require()</code>	130
2.5 Übung	29	7.5 Übungen	133
3 Variablen und Operatoren	30	8 Mit Daten aus externen Dateien arbeiten	135
3.1 Variablen	30	8.1 Externe Dateien nutzen	135
3.2 Variablen und Operatoren für Zahlen	32	8.2 Dateien öffnen, lesen und schließen	136
3.3 Variablen und Operatoren für Zeichenketten	35	8.3 Weitere Möglichkeiten zum Lesen von Dateien	141
3.4 Konstanten	41	8.4 In Dateien schreiben	144
3.5 Übungen	44	8.5 Weitere Datei-Funktionen	147
4 Kontrollstrukturen	46	8.6 Zugriffszähler für eine Webseite	148
4.1 Kontrollstrukturen einsetzen	46	8.7 Übung	150
4.2 Die einfache <code>if</code> -Anweisung	49	9 Zeichenketten-Funktionen	151
4.3 Die <code>if</code> -Anweisung mit <code>else</code> -Zweig	52	9.1 Zeichenketten ausgeben	151
4.4 Verschachtelte <code>if</code> -Anweisungen	55	9.2 Zahlen formatieren	155
4.5 Erweiterte <code>if</code> -Anweisung mit <code>elseif</code>	57	9.3 Nach Zeichenketten suchen	157
4.6 Fallauswahl mit der <code>switch</code> -Anweisung	58	9.4 Position und Teil einer Zeichenkette ermitteln	161
4.7 Fallzuweisung mit dem <code>match</code> -Ausdruck	62	9.5 Zählen innerhalb von Zeichenketten	163
4.8 Schleifen	64	9.6 Zeichenketten vergleichen	165
4.9 Mit der <code>while</code> -Schleife arbeiten	64	9.7 Zeichenketten modifizieren	166
4.10 Mit der <code>for</code> -Schleife arbeiten	66	9.8 Mit Arrays und Zeichenketten arbeiten	170
4.11 Schleifen abbrechen	68	9.9 Übungen	172
4.12 Übungen	70	10 Datum und Uhrzeit	174
5 Arrays	73	10.1 Datum und Zeit ermitteln	174
5.1 Grundlagen zu Arrays	73	10.2 Datum und Zeit formatieren	176
5.2 Indizierte eindimensionale Arrays erstellen	75	10.3 Datumsangabe an Sprache anpassen	179
5.3 Assoziative eindimensionale Arrays erstellen	76	10.4 Länder- und Spracheinstellungen ändern	180
5.4 Arrays mit der Kurzschreibweise erstellen	78	10.5 Zeitfunktionen	182
5.5 Mit eindimensionalen Arrays arbeiten	79	10.6 Datumsangaben überprüfen	186
5.6 Daten aus eindimensionalen Arrays extrahieren	81	10.7 Übungen	188
5.7 Mehrdimensionale indizierte Arrays erstellen	84		
5.8 Mit mehrdimensionalen assoziativen Arrays arbeiten	86		
5.9 Daten aus mehrdimensionalen Arrays extrahieren	87		

11 Sessions	190
11.1 Mit Sessions arbeiten	190
11.2 Session starten bzw. fortsetzen	192
11.3 Daten in einer Session speichern	194
11.4 Daten einer Session abrufen	197
11.5 Session-Daten und Session löschen	198
11.6 Fallbeispiel „Shop“	200
11.7 Übung	208
12 Grundlagen Datenbank MySQL	210
12.1 Die Datenbanken MySQL und MariaDB	210
12.2 MySQL-Datenbanken mit phpMyAdmin verwalten	210
12.3 MySQL-Datenbanken mit phpMyAdmin erstellen	213
12.4 Mit einer MySQL-Tabelle arbeiten	217
12.5 SQL-Dumps exportieren und importieren	218
12.6 PHP und MySQL	220
12.7 MySQL-Abfragen	227
12.8 Rückgabe aus MySQL-Abfrage auswerten	230
12.9 Formulardaten in einer MySQL-Datenbank speichern	232
12.10 Übung	234
A Installation und Konfiguration der Software	236
A.1 Installation und Konfiguration von XAMPP	236
A.2 Mit XAMPP arbeiten	239
A.3 Installation und Konfiguration von Notepad++	242
A.4 Mit den XAMPP-Konfigurationsdateien arbeiten	244
A.5 Mit MySQL und phpMyAdmin arbeiten	246
Stichwortverzeichnis	250

Bevor Sie beginnen ...

HERDT BuchPlus – unser Konzept:

Problemlos einsteigen – Effizient lernen – Zielgerichtet nachschlagen

Nutzen Sie dabei unsere maßgeschneiderten, im Internet frei verfügbaren Medien:



Wie Sie schnell auf diese BuchPlus-Medien zugreifen können, erfahren Sie unter www.herdt.com/BuchPlus.

Empfohlene Vorkenntnisse

- ✓ Umgang mit Betriebssystemen, Webbrowser
- ✓ HTML-Grundlagen (mindestens Zeichen- und Absatzformatierung, Tabellen, Formulare)
- ✓ Grundkenntnisse Datenbanken (MariaDB/MySQL)

Hinweise zu Soft- und Hardware

PHP ist eine Open-Source-Software. Sie benötigen zur Erstellung von Webseiten mit PHP-Anweisungen nicht mehr als einen Texteditor. Damit Sie Ihre PHP-Programme direkt an Ihrem Rechner testen können, empfehlen wir Ihnen folgende Software:

- ✓ die kostenfreie Entwicklungsumgebung XAMPP in der Version 8.0.1 vom 5. Januar 2021 (<https://www.apachefriends.org/de/>), unter anderem mit den Bestandteilen:
 - ✓ PHP, Version 8.0.1 ✓ Apache-Webserver, Version 2.4.46 ✓ MariaDB 10.4.17Die XAMPP-Software wurde in allen Beispielen mit den Standardeinstellungen verwendet. Installations- und Konfigurationshinweise zur Software finden Sie im Anhang.
- ✓ den Freeware-Texteditor Notepad++ in der Version 7.9.2 (<https://notepad-plus-plus.org/downloads/>);
- ✓ einen Webbrowser.

In diesem Buch sind alle Screenshots mit dem Mozilla Firefox-Browser auf Windows 10 erstellt. Grundsätzlich spielt der Browser für das Arbeiten mit PHP keine Rolle. Sie können ebenfalls Google Chrome, Microsoft Edge, Apple Safari oder auch Opera verwenden.

1

Einführung in PHP

 **Beispieldateien:** Dateien im Ordner *Kapitel_01*

1.1 PHP-Code in Webseiten

Was ist PHP?

PHP (Abkürzung für **P**H_P: **H**ypertext **P**reprocessor, früher auch **P**ersonal **H**ome **P**age **T**ools) ist eine **Open- Source-Skriptsprache**, die speziell für den Einsatz im **Internet** entwickelt wurde.

Die Stärken von PHP liegen in der recht leichten Erlernbarkeit und in der breiten Funktionspalette.

PHP setzt dort an, wo HTML seine Grenzen erreicht: HTML-Seiten sind starr. Mithilfe von PHP können auf einer Webseite

- ✓ Interaktionen eingebaut,
- ✓ Datenbanken gesteuert oder
- ✓ die Seite individuell an das Benutzerverhalten angepasst werden.

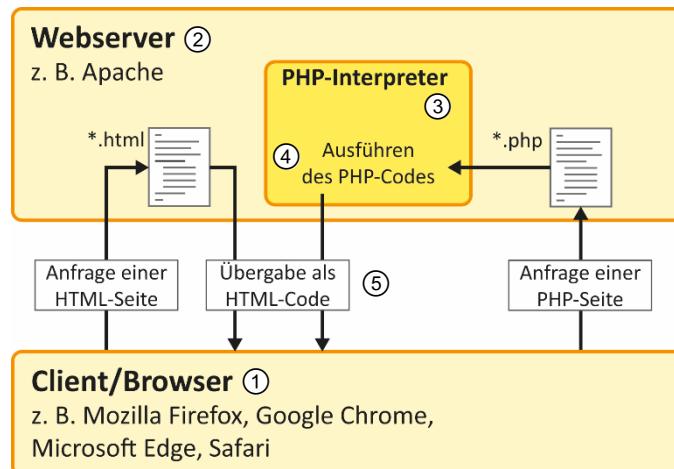
Webseiten mit PHP-Code verarbeiten

PHP-Code wird von einem Webserver, also serverseitig verarbeitet. Das Ergebnis wird danach vom Webserver als HTML-Code (Quellcode) an den Browser gesendet. Ob der HTML-Code aus einer statischen HTML-Datei stammt oder das Ergebnis eines ausgeführten PHP-Skripts ist, ist für den Browser nicht ersichtlich. Zur Darstellung erhält der Browser in beiden Fällen HTML-Code.

Öffnet der Betrachter eine PHP-Webseite im Browser ①,

- ✓ werden die Anweisungen von PHP auf dem Server ② interpretiert ③,
- ✓ ausgeführt ④ und
- ✓ das Ergebnis als HTML-Code an den Browser zurückgesendet ⑤.

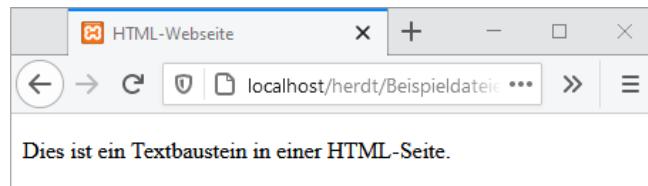
Der Nutzer sieht im Browser nur die Darstellung des zurückgelieferten HTML-Codes (Resultat nach den Vorgängen ③ und ④). Der PHP-Quellcode selbst ist dem Betrachter nicht zugänglich. Programmieralgorithmen, die Verarbeitung von Daten oder beispielsweise Zugriffe auf Datenbanken bleiben dem Nutzer verborgen.



Webseite ohne PHP-Code – Beispiel: *html-inhalt.html*

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>HTML-Webseite</title>
  </head>
  <body>
    <p>Dies ist ein Textbaustein in einer HTML-Seite.</p>
  </body>
</html>
```

HTML-Dateien können ohne weitere Software von jedem beliebigen Browser angezeigt werden. Es reicht ein Doppelklick auf den Dateinamen, um die Datei zu öffnen. In der Adresszeile des Browsers erscheint der Pfad des Dateisystems, in dem die HTML-Datei liegt.



Webseite um PHP-Code erweitern

- **Schritt 1:** Kopieren Sie die Datei *html-inhalt.html* und ändern Sie den Dateinamen in *html-inhalt-und-php.php*.

Achten Sie auf die Angabe der Dateinamenerweiterung *php*. Sobald die Dateinamenerweiterung von *html* in *php* geändert ist, haben Sie eine lauffähige PHP-Datei erzeugt.

PHP-Dateien benötigen im Gegensatz zu HTML-Dateien einen Webserver, der PHP verarbeitet (interpretiert). Webserver werden von Internet-Providern zur Verfügung gestellt. Mit dem XAMPP-Paket (vgl. Abschnitt A.1) können Sie einen lokalen Webserver für die Entwicklung von PHP auf Ihrem eigenen Rechner installieren.

Die Ausgabe der Datei *html-inhalt-und-php.php* ist identisch mit der Datei *html-inhalt.html*. Beide Dateien enthalten den gleichen HTML-Code und sonst keine weiteren Inhalte. HTML-Code wird vom PHP-Interpreter weder interpretiert noch ausgeführt, sondern direkt an den Browser gesendet und dort dargestellt.

Über PHP-Code soll zusätzlich der Text Ich freue mich auf PHP! ausgegeben werden:

- **Schritt 2:** Öffnen Sie die Datei *html-inhalt-und-php.php* im Editor (z. B. Notepad++) und fügen Sie PHP-Code in den HTML-Code ein:

Beispiel: *html-inhalt-und-php.php*

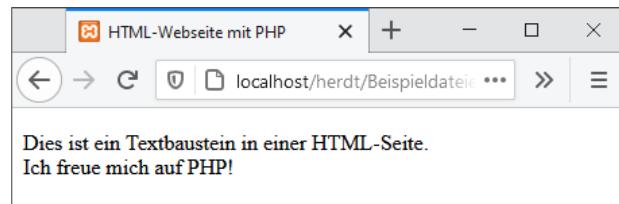
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>HTML-Webseite mit PHP</title>
  </head>
  <body>
    <p>Dies ist ein Textbaustein in einer HTML-Seite.<br>
      ①
      ② <?php
      ③   echo "Ich freue mich auf PHP!";
      ④     ?>
    </p>
  </body>
</html>
```

- ① Fügen Sie hinter dem vorhandenen Text einen **Zeilenumbruch** `
` ① ein, um die Ausgabe, die vom HTML- und PHP-Code erzeugt wird, in zwei Zeilen zu bewirken.
- ② Beginnen Sie einen PHP-Block im HTML-Code mit dem öffnenden PHP-Tag `<?php` ②.
- ③ Verwenden Sie zur Ausgabe des Textes den PHP-Befehl `echo`, gefolgt vom Text in Anführungszeichen. Befehle werden in PHP durch ein **Semikolon** abgeschlossen.
- ④ Beenden Sie den PHP-Block mit einem schließenden PHP-Tag `?>` ④.

PHP-Code kann an jeder **beliebigen** Stelle und **beliebig** oft im HTML eingefügt werden. Wichtig ist, dass jeder PHP-Block von dem öffnenden PHP-Tag `<?php` und dem schließendem PHP-Tag `?>` umschlossen ist.

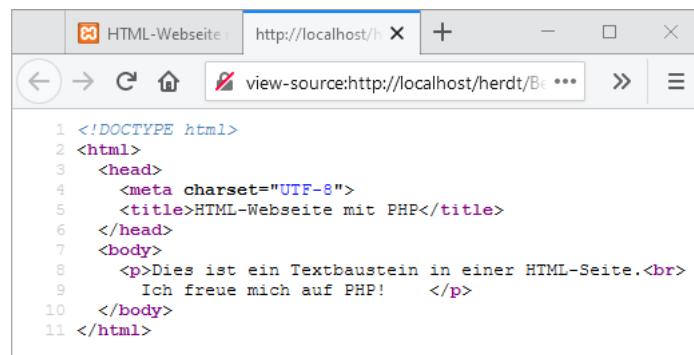
- **Schritt 3:** Starten Sie den XAMPP-Webserver über das Control Panel, falls dieser nicht als Dienst installiert ist (vgl. Installationshinweise Abschnitt A.1). Rufen Sie die PHP-Datei im Browser mit folgender URL auf: `http://localhost/[Pfad zur Datei]/html-inhalt-und-php.php` bzw. `http://localhost:8080/[Pfad zur Datei]/html-inhalt-und-php.php` auf einem Mac. Falls Sie die Datei *html-inhalt-und-php.php* direkt im Ordner `/htdocs` des Webservers gespeichert haben, lautet der Aufruf im Browser `http://localhost/html-inhalt-und-php.php`. Haben Sie die Datei in einem Unterordner abgelegt, müssen Sie `[Pfad zur Datei]` durch den Namen Ihrer Ordner ersetzen (vgl. Kapitel 2).

Die nebenstehende Abbildung zeigt die Webseite „html-inhalt-und-php.php“ im Browser.



Den vom Server zurückgelieferten HTML-Code sehen Sie in nebenstehender Abbildung.

Die PHP-Anweisungen wurden durch den auszugebenden Inhalt ersetzt. Der Betrachter sieht den ursprünglichen PHP-Code im HTML-Code nicht.



Merkmale von PHP

- ✓ PHP wurde speziell für die Programmierung dynamischer Webseiten entwickelt. Funktionen in PHP sind deshalb auf die Programmierung von Internetanwendungen abgestimmt.
- ✓ PHP zeichnet sich durch einen großen Funktionsumfang aus.
- ✓ PHP ist plattformunabhängig. Webserver mit PHP-Unterstützung stehen sowohl für Linux, macOS und Windows zur Verfügung.
- ✓ Die PHP-Anweisungen werden direkt auf dem Server ausgeführt und nicht vom Browser (Client).
- ✓ Der PHP-Quellcode ist für den Betrachter nicht sichtbar, sondern nur die auf dem Webserver generierte HTML-Ergebnisdatei.
- ✓ PHP ist fehlertoleranter als andere Programmiersprachen wie beispielsweise JAVA.
- ✓ PHP ist kostenlos erhältlich.

1.2 Informationen zu PHP

Entwicklung von PHP

1994 begann Rasmus Lerdorf mit der Entwicklung einer Skriptsprache für das Internet. Die Sprache sollte einfach zu erlernen sein. Die erste Version nannte er PHP, abgeleitet von „Personal Home Page Tools“, einer Sammlung von diversen Skripten zum Einbinden in Webseiten. Der Interpreter beherrschte wenige Befehle, sodass der Funktionsumfang klein war.

Ein Jahr später erweiterte Lerdorf den Funktionsumfang, indem er den Interpreter neu programmierte. Hinzu kamen die Funktionen des vorhandenen Formular Interpreters, kurz FI. Dieser Interpreter war in der Lage, Formulardaten ohne CGI (Common Gateway Interface) zu verarbeiten. Die Kombination der Home Page Tools und des Formular Interpreters (PHP2) nannte er PHP/FI. Das Neue an dieser Version war die Anbindung an die MySQL-Datenbank.

Das Privatprojekt Lerdorfs wurde 1997 von einem Team von Programmierern übernommen. Mit dabei Andi Gutmans und Zeev Suraski, die Gründer der Firma *Zend Technologies Ltd.* Die Funktionen der PHP/FI-Version wurden in das PHP3-Format portiert und neue Befehle hinzugefügt.

Mit der Version PHP3 und ihrer Anbindung an verschiedenste Datenbanken begann die Verbreitung der Skriptsprache zur dynamischen Seitengenerierung. Konkurrenzprodukte sind andere serverseitige Programmiersprachen wie ASP.NET, Java Server Pages (JSP) oder ColdFusion.

Seit PHP4 bildet ein von Zend entwickelter Compiler zur schnelle(re)n Ausführung des Codes, die sogenannte „Zend Engine“, das Rückgrat der PHP-Programmierung. PHP wurde durch Erweiterungen seines Funktionsumfangs, wie z. B. das Sessionmanagement, verbessert.

PHP-Versionen und -Verbreitung

PHP5 ist seit Sommer 2004 auf dem Markt. Eine stark verbesserte Zend Engine II ist ebenso wie zahlreiche Verbesserungen, z. B. in der objektorientierten Programmierung und in Fragen rund um die Sicherheit der Programmierung, dafür verantwortlich, dass die Zahl der Webseiten mit PHP-Unterstützung schnell wächst.

Bereits ein Jahr, nachdem PHP 5 veröffentlicht wurde, wurde 2005 mit der Entwicklung von PHP 6 begonnen. Unter anderem sollte eine native Unicode-Unterstützung implementiert werden. Die Entwicklung von PHP 6 wurde mittlerweile eingestellt.

2014 wurde mit der Entwicklung einer neuen Hauptversion von PHP begonnen. Da die Entwicklung von PHP 6 als gescheitert bezeichnet werden kann, haben sich die Entwickler entschieden, die 6er Version einfach zu überspringen und die neue PHP-Version PHP 7 zu nennen. Damit ist PHP 7 die direkte Nachfolgeversion von PHP 5.6.



PHP 7 ist aktuell (April 2021) bei den großen deutschen Providern die Standardversion. PHP 8 wurde im November 2020 veröffentlicht. Bevor Provider die nächste PHP-Version anbieten, warten diese eine Weile, bis die meisten Fehler in der neuen Version gefunden und behoben wurden. Erfahrungsgemäß (und unverbindlich) ist 2022 mit PHP 8 bei den meisten Providern zu rechnen. Bevor Sie also neue Features von PHP 8 verwenden wollen, prüfen Sie vorher, welche PHP-Version Ihr Provider anbietet.

PHP wird auf 79 % aller Webseiten eingesetzt (Stand: Ende 2019, Quelle: <https://de.wikipedia.org/wiki/PHP>) und ist damit die am häufigsten verwendete Programmiersprache zum Erstellen von Webseiten.

Funktionsumfang von PHP

PHP hat heute einen großen Funktionsumfang, mit dem weitestgehend alle gängigen Anforderungen im Webumfeld gelöst werden können. Eine Auswahl oft benötigter Funktionen finden Sie nachfolgend:

- ✓ HTML-Seiten generieren, in denen benutzerbezogene Daten verarbeitet sind
- ✓ Verarbeiten unterschiedlicher Datentypen, wie Integer, Fließkommazahlen, Zeichenketten oder Arrays

- ✓ Auslesen, Überprüfung und Verarbeiten von Formulardaten
- ✓ Daten aus Datenbanken auslesen, bearbeiten und wieder in Datenbanken speichern
- ✓ Zeichenketten auslesen, verändern, abschneiden, umwandeln, in bestimmten Formaten wieder ausgeben
- ✓ Datums- und Zeitangaben auslesen, erkennen, generieren, verändern
- ✓ E-Mails erstellen und versenden
- ✓ Grafiken öffnen oder generieren, verändern, speichern
- ✓ Dateien (z. B. Text- oder CSV-Dateien) öffnen, auslesen, verändern, speichern
- ✓ Über Schnittstellen Dienste anderer Webseiten aufrufen, einlesen, auswerten
- ✓ Cookies speichern und auslesen, Sessions verwalten

Über die Kernfunktionalitäten ist PHP zusätzlich um verschiedenste Bibliotheken erweiterbar, beispielsweise ImageMagick. Die Bibliothek ImageMagick bietet mehr Funktionalitäten als PHP-eigene Funktionen zur Bildverarbeitung.

Im Rahmen der Grundlagen von PHP 8.0 werden die Basisfunktionalitäten von PHP gezeigt, die ausreichend sind, um eine dynamische Webseite zu programmieren.

Die Kombination von PHP und der Datenbank MySQL bzw. MariaDB ist auf der Mehrzahl aller Webserver zu finden. PHP ist jedoch nicht auf MySQL beschränkt, sondern kann mit unterschiedlichen Datenbanken arbeiten – entweder direkt über eigene PHP-Funktionalitäten oder auch über ODBC-Schnittstellen. So kann PHP beispielsweise auch mit Oracle-Datenbanken arbeiten.

PHP unterstützt darüber hinaus die Kommunikation mit anderen Services, z. B. Protokollen wie LDAP, IMAP, SNMP, NNTP, POP3 oder HTTP.

Zudem ist PHP zur Kommandozeilenprogrammierung (z. B. für sogenannte „cronjob“-Skripte, die wiederkehrende Aufgaben automatisieren) oder für die Entwicklung von Desktop-Anwendungen einsetzbar.

Informationen zu PHP im Internet

Folgende ausgewählte Webseiten bieten ausführliche Informationen zu PHP:

https://www.php.net/	Die wichtigste Seite für PHP-Programmierer. Hier finden Sie die komplette Dokumentation von PHP, alle Funktionen einschließlich PHP-Code-Beispielen und PHP-Versionshinweise. Auch Probleme mit Funktionen werden dort besprochen.
https://www.php.net/manual/de/	Deutschsprachiges Onlinehandbuch der PHP-Dokumentationsgruppe
https://www.w3schools.com/php/ https://www.w3schools.com/html/ https://www.w3schools.com/css/	Die Webseite von W3Schools bietet umfangreiche Tutorials zu PHP und allen weiteren Webtechniken wie HTML und CSS. Die Seite ist gut strukturiert, die Erklärungen sind ergänzt durch viele hilfreiche Beispiele.



<https://www.php.net/> ist die Standardreferenz für PHP-Entwickler. Große Teile der Referenz stehen in Deutsch zur Verfügung. Gerade bei neueren Features lohnt sich aber der Wechsel in die englische Variante, da diese mitunter auf einem neueren Stand ist.

1.3 PHP-Version 8.0

Die herausragende Neuerung von PHP 8 ist wahrscheinlich der JIT-Compiler (just in time). PHP-Skripte werden beim Aufruf in eine Maschinensprache übersetzt und dann auf dem Rechner ausgeführt. Bislang wurde Code, der auch mehrfach durchlaufen wird, jedes Mal neu übersetzt. PHP 8 erkennt sich wiederholende Programmabschnitte, übersetzt diese nur 1-mal und speichert diese für die wiederholte Ausführung in einem Cache. Das zieht eine wesentlich schnellere Ausführung eines PHP-Skriptes nach sich. Das bedeutet damit auch, dass Anwendungen, aufgrund deren Performance bislang C oder C++ notwendig waren, mitunter auch durch PHP ersetzt werden können.

Die **Fehler-Level** wurden überarbeitet, sodass viele Programmierfehler, die bislang nur eine Warnung bewirkt haben, nun einen wirklichen Fehler erzeugen und – falls kein Fehler-Handling implementiert ist – Skripte zum Abbruch bringen können. Dies wird zwangsläufig zu Problemen führen, wenn Provider vollständig auf PHP 8 umstellen. Alte Skripte können eventuell nicht mehr laufen. Das gleiche bewirken einzelne Funktionen, die entweder entfernt oder in der Funktionalität verändert wurden. Damit ist PHP 8 nicht abwärtskompatibel. Die strengeren Fehler-Level bewirken aber gleichzeitig hochwertigere PHP-Skripte, da Entwickler zur sauberen Programmierung gezwungen sind.

Darüber hinaus umfasst PHP 8 viele Änderungen, die jedoch kaum Auswirkung auf das Lernen der Grundlagen haben. Relevant ist hier das neue Sprachkonstrukt `match()` (siehe Kapitel 4.7), welches eine zusätzliche Kontrollstruktur liefert sowie `str_contains()`, `str_starts_with()` und `str_ends_with()` (Kapitel 9.3), welche das Suchen in Zeichenketten intuitiver machen.

Die wichtigsten Änderungen und Neuerungen von PHP 8 sind nachfolgend aufgeführt:

Neuerungen und Änderungen von PHP 8

Neuerungen bzw. Änderungen	Erklärung	Kapitel
JIT-Compiler	Bessere Performance durch gecachten Maschinencode. Vermutlich die wichtigste Neuerung in PHP 8, was jedoch für die Syntax von PHP irrelevant ist	–
Überarbeitung der Error-Klassifikation	Verschiedene <i>notice</i> -Meldungen sind nun <i>warning</i> oder <i>error</i> .	2.4, 3.3
Concatenation Precedence	Rechenoperation hat Vorrang vor Konkatenation (vergleichbar mit Punkt-vor-Strich-Rechnung)	3.3
Constructor Promotion	Sichtbarkeitsmerkmale wie <code>public</code> , <code>private</code> oder <code>protected</code> können nun in der <code>_construct</code> -Methode verwendet werden und stellen so eine Kurzschreibweise dar.	–
<code>match()</code>	Neue Kontrollstruktur, ähnlich wie <code>switch()</code>	4.7
Trailing comma in parameter lists	Erlaubt ein Komma hinter der letzten Variablen in einem Funktionsaufruf. Bislang erzeugte dies in PHP einen Error.	7.1
Passing mandatory arguments after optional is deprecated	Die Reihenfolge von erforderlichen und optionalen Parametern in Funktionsköpfen ist nun beliebig (zuvor mussten optionale Parameter hinter Parametern ohne Vorbelegung stehen).	7.2
Union Types	Bei der Deklaration eines Parameters können nun mehrere optionale Variablentypen angegeben werden.	7.2
<code>mixed</code>	Neuer Wert zur Typ-Deklaration. Entspricht dem Union Type <code>array bool callable int float object resource string null</code>	–
Named Arguments	Funktionen können benannte Variablen übergeben werden. Die Benennung legt fest, welcher Parameter welcher Variablen zugewiesen wird.	7.2
<code>str_contains()</code>	Prüft, ob eine Zeichenkette in einer anderen Zeichenkette enthalten ist	9.3
<code>str_starts_with()</code> und <code>str_ends_with()</code>	Prüft, ob eine Zeichenkette mit einer bestimmten Zeichenkette anfängt bzw. endet	9.3
<code>fdiv(\$num1, \$num2)</code>	Dividiert den 1. Wert durch den 2. Wert. Das Teilen durch 0 führt nicht zum Fehler. Die Funktion gibt in dem Fall einen Wert <code>INF</code> oder <code>NAN</code> zurück.	–
@-Operator entfernt	Der @-Operator zum Unterdrücken von Fehlermeldungen ist nicht mehr verfügbar.	–

2

Grundlegende Sprachelemente

 **Beispieldateien:** Dateien im Ordner *Kapitel_02*

2.1 PHP in HTML einbinden

Dateiendung für PHP-Dateien

PHP kann direkt in den HTML-Code eingefügt werden. Damit der Webserver erkennt, dass es sich um eine Datei mit PHP-Anweisung(en) handelt, werden die Dokumente mit der Dateiendung (Dateinamenerweiterung) ***.php** versehen. Die Dateiendung ***.php** ist für den Webserver das Signal, den PHP-Interpreter aufzurufen, dieser führt dann die PHP-Datei aus und liefert das Ergebnis, also das generierte HTML an den Browser aus.

Die Dateinamenerweiterung ***.php** ist die gängige Schreibweise und entspricht der Standardkonfiguration. In der Vergangenheit wurde auch ***.php4**, ***.php5** oder ***.phtml** als Dateiendung für PHP-Dateien verwendet. Mit welchen Dateiendungen der Webserver Dateien als PHP-Dateien erkennt, kann in der Datei *httpd.conf* bzw. bei der XAMPP-Software in der Datei *httpd-xampp.conf* konfiguriert werden.

In normalen HTML-Dateien (Dateiendung ***.htm** bzw. ***.html**) werden PHP-Anweisungen nicht interpretiert.

PHP-Anweisungen einfügen

PHP-Code kann und darf an jeder beliebigen Stelle im HTML vorkommen. PHP-Anweisungen können Sie sowohl vor dem einleitenden HTML-Tag `<!DOCTYPE html>` einfügen, aber auch im `<head>` oder im `<body>` des HTML-Quelltextes, je nachdem, wo Sie den PHP-Code benötigen.

<code><?php PHP-Anweisung (en) ?></code>	XML-konforme Standardschreibweise Ein sogenannter PHP-Block wird durch das Tag <code><?php</code> geöffnet und nach den PHP-Anweisungen mit dem Tag <code>?></code> geschlossen. Diese XML-konforme Schreibweise ist sowohl die empfohlene als auch gängige Art, PHP-Blöcke auszuzeichnen.
--	--

PHP-Code wird als PHP-Block in das HTML eingefügt. Ein PHP-Block wird mit einem öffnenden PHP-Tag `<?php` eingeleitet und durch ein schließendes PHP-Tag `?>` beendet, die PHP-Befehle schreiben Sie innerhalb dieser PHP-Tags. Der PHP-Interpreter erkennt anhand dieser Tags die PHP-Blöcke und führt den PHP-Code darin aus.

In einer HTML-Datei können beliebig viele PHP-Blöcke vorkommen. Auch innerhalb einer Zeile im HTML können mehrere PHP-Blöcke eingefügt werden.

Alternative PHP-Tags

<code><? PHP-Anweisung(en) ?></code>	Kurzschreibweise Falls die Konfigurationsvariable <code>short_open_tag</code> in der Datei <code>php.ini</code> auf den Wert <code>On</code> gesetzt wurde, können Sie <code>php</code> im öffnenden PHP-Tag weglassen.
--	---

Alternativ zur Standardschreibweise können Sie auch PHP-Tags in der Kurzschreibweise `<? ... ?>` verwenden, in der das `php` im öffnenden PHP-Tag entfällt. Gerade bei vielen PHP-Blöcken spart das Tipparbeit.

Allerdings müssen Sie die Kurzschreibweise zuvor in der `php.ini` aktivieren, was auf dem eigenen Rechner für die Entwicklung möglich ist. Das Aktivieren an sich stellt das geringere Problem dar. Das eigentliche Problem liegt darin, dass Ihr PHP-Skript später auf einem Webserver eines professionellen Providers laufen soll. Dort haben Sie jedoch keinen Zugriff auf die Konfigurationsdatei `php.ini` und können die Kurzschreibweise nicht aktivieren. Auch wenn Ihr PHP-Code während der Entwicklung funktioniert hat, läuft er später auf dem gemieteten Webserver nicht.

Vor PHP 7.0 gab es zwei weitere Schreibweisen, um PHP-Blöcke im HTML einzufügen. Zum einen die ASP-Schreibweise `<% ... %>` zum anderen die Skript-Schreibweise `<script language="php"> ... </script>`. Auch in älteren PHP-Versionen waren beide Schreibweisen nicht gängig, funktionierten aber, wenn diese entsprechend in der `php.ini` konfiguriert waren. Mit PHP 7.0 ist diese Syntax weggefallen und würde zum Fehler führen, wenn Sie diese Art der PHP-Einbindung verwenden würden.

Verwenden Sie nach Möglichkeit die allgemeingültige **XML-Schreibweise** (`<?php ... ?>`) zum Einfügen von PHP-Anweisungen in Ihren Skripten. Dies ist die gängige Einstellung der Webserver von Internet-Providern. Damit stellen Sie sicher, dass Ihre Seite auf allen Servern läuft.

Informationsdatei zur PHP-Konfiguration anzeigen

Sie können sich mit einem Skript die PHP-Konfiguration des Servers anzeigen lassen, auf dem dieses Skript ausgeführt wird. Zu diesem Zweck stellt PHP die Funktion `phpinfo()` bereit. Damit erhalten Sie detaillierte Informationen zur installierten PHP-Version samt installierten Erweiterungen.

Wenn Sie mit dem empfohlenen XAMPP-Paket arbeiten, rufen Sie im Webbrowser Ihren lokalen Rechner als (PHP-)Webserver auf. Über die Adresse <http://localhost> – oder alternativ über die IP-Adresse <http://127.0.0.1> – wird automatisch das im Webserver definierte sogenannte *document root*-Verzeichnis geöffnet. Bei einer XAMPP-Standardinstallation ist das unter Windows der Ordner *C:\xampp\htdocs*. Unter macOS finden Sie das *root*-Verzeichnis unter *lampp/htdocs* (in der gemounteten Virtuellen Maschine, siehe Kapitel A am Buchende).

Beispiel: *phpinfo.php*

- ▶ Erstellen Sie folgende Datei und speichern Sie sie im Verzeichnis *C:\xampp\htdocs* (*lampp/htdocs* auf dem Mac) unter dem Dateinamen *phpinfo.php*.
- ▶ Um das Skript zu testen, rufen Sie im Browser die PHP-Datei mit folgender Adressangabe auf Ihrem lokalen Webserver auf:
<http://localhost/phpinfo.php> oder <http://127.0.0.1/phpinfo.php> bzw. auf dem Mac
<http://localhost:8080/phpinfo.php>

```
<?php
    phpinfo();
?>
```

Beispieldatei „*phpinfo.php*“

PHP Version 8.0.1 ← Installierte PHP-Version	
System	Windows NT DESKTOP-NKSJSL9 10.0 build 19041 (Windows 10) AMD64
Build Date	Jan 5 2021 23:36:34
Build System	Microsoft Windows Server 2016 Standard [10.0.14393]
Compiler	Visual C++ 2019
Architecture	x64
Configure Command	cscript /nologo /e:jscript configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-pdo-oci=c:\php-snap-build\dep-oci8\dep-oci8-12c-c:\php-snap-build\dep-oci8\instantclient_12_1\ sdk,shared" "--with-oci8-19-c=c:\php-snap-build\dep-oci8\instantclient_19_9\ sdk,shared" "--enable-object-out-dir=..\\obj" "--enable-com-dotnet=shared" "--without-analyzer" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	<i>no value</i>
Loaded Configuration File	<i>C:\xampp\php\php.ini</i>

Ausschnitt aus der Anzeige der PHP-Konfiguration

In der Kopfzeile der Seite wird Ihnen die installierte PHP-Version angezeigt. Auf der Seite selbst erscheinen dann diverse Informationen zur PHP-Installation, unter anderem Pfade zu Konfigurationsdateien, Umgebungsvariablen oder installierte Pakete.

! PHP-Dateien können nicht mit einem Doppelklick auf die Datei geöffnet werden. Sie können die Datei zwar mit der Maus in den Browser ziehen, dann wird diese jedoch lediglich als HTML-Datei dargestellt, PHP-Anweisungen werden dann nicht interpretiert. Damit der PHP-Code ausgewertet wird, muss der Aufruf der PHP-Datei über einen PHP-Interpreter erfolgen, in den Beispielen in diesem Buch immer über die Serveradresse <http://localhost> oder <http://127.0.0.1> und anschließend jeweils Pfad- und Dateiname.

2.2 Codieren von PHP-Skripten

PHP-Skripte können aus Hunderten von Codezeilen bestehen. Oft ist es sinnvoll, Codezeilen mehrfach zu verwenden. Achten Sie deshalb darauf, dass Ihr Code

- ✓ leicht zu lesen und zu verstehen ist,
- ✓ gut kommentiert bzw. dokumentiert ist,
- ✓ auch von anderen Programmierern leicht zu verstehen und damit einfach zu pflegen ist,
- ✓ selbsterklärende Variablen und Funktionsnamen berücksichtigt,
- ✓ keine kryptischen Bezeichner enthält.

Nachfolgend finden Sie einige Tipps, wie Sie Ihre PHP-Skripte verständlich und übersichtlich gestalten können.

PHP-Anweisungen zeilenweise schreiben

Jede PHP-Anweisung wird durch ein Semikolon beendet. Daran erkennt der PHP-Interpreter das Ende einer und den Anfang der nächsten Anweisung. Schreiben Sie zur besseren Übersicht jede PHP-Anweisung in eine eigene Zeile und verwenden Sie Einrückungen, um den Code übersichtlicher zu gestalten. (PHP selbst benötigt keine Zeilenumbrüche, ein PHP-Skript mit mehreren Anweisungen, die per Semikolon getrennt sind, könnte auch in einer einzelnen Zeile stehen.)

Beispiel

```
<?php  
    PHP-Anweisung;  
    PHP-Anweisung;  
?>
```

Skripte gut kommentieren

Um Quellcode zu erläutern, werden Kommentare verwendet:

- ✓ Kommentare erhöhen die Nachvollziehbarkeit der Programmierung.
- ✓ Kommentare enthalten z. B. eine Kurzbeschreibung der Verwendung einer Funktion.
- ✓ Kommentare beschreiben den Zweck von Variablen oder erklären den Algorithmus eines Programms.

Gute Kommentare beschreiben, was **nicht** im PHP-Code steht, sondern z. B., welcher Ansatz für eine Berechnung verwendet wurde oder den Link zu einer verwendeten Bibliothek. Beim Kommentieren sollten Sie immer die Frage „Was bräuchten die Kollegen, um diesen Code weiterentwickeln zu können“ im Kopf haben.

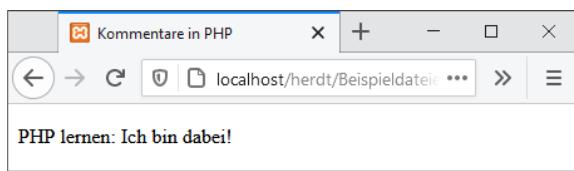
PHP-Kommentare sind für den Betrachter **nicht** zu sehen, auch nicht im HTML-Quelltext im Browser. Die Kommentare werden beim Auswerten des PHP-Codes auf dem Webserver ignoriert und nicht ausgegeben.

Syntax und Bedeutung der Kommentare

Beispiel: *kommentar.php*

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Kommentare in PHP</title>
  </head>
  <body>
    <?php
①      echo "<p>PHP lernen: " // einzeiliger Kommentar
②      // einzeiliger Kommentar (komplette Zeile)
③      # einzeiliger Kommentar per Raute
④      /*
        mehrzeilige Kommentare werden häufig verwendet,
        um den Quellcode ausführlicher zu beschreiben
      */
⑤      echo "Ich bin dabei!</p>";
      ?>
    </body>
  </html>
```

- ①+② Einzelige Kommentare beginnen mit den Zeichen `//` und benötigen kein abschließendes Zeichen, um dem PHP-Interpreter zu signalisieren, dass der Kommentar beendet ist. Der Zeilenumbruch im PHP-Skript beendet diesen Kommentar. Sie werden dazu verwendet, Erläuterungen ① direkt hinter einem Befehl zu notieren oder auch ② eine gesamte Zeile als Kommentar zu kennzeichnen.
- ③ Auch das Raute-Zeichen `#` kann für einzelne Kommentare verwendet werden. Zwischen `//` und `#` besteht für PHP kein Unterschied.
- ④+⑤ Mit den Zeichen `/*` werden mehrzeilige Kommentare eingeleitet ④ (sogenannte Kommentarblöcke). Sie ermöglichen eine ausführliche Erklärung des Quellcodes. Mit den Zeichen `*/` werden diese Kommentare beendet ⑤.



Kommentare im PHP-Code sind bei der Ausgabe im Browser herausgefiltert ...



... und auch im HTML-Quelltext sind keine PHP-Kommentare zu finden.

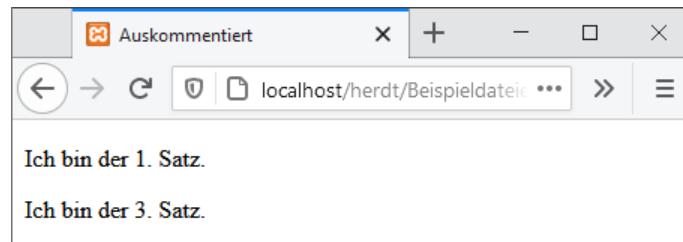
Kommentare können auch genutzt werden, um einen Abschnitt des PHP-Codes – für Testzwecke oder um Fehler einzuschränken – als Kommentar zu kennzeichnen und damit vor dem PHP-Interpreter zu verstecken (dies wird als „Auskommentieren“ bezeichnet), sodass dieser Abschnitt vorerst nicht ausgeführt wird.

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Auskommentiert</title>
  </head>
  <body>
    <?php
      echo "<p>Ich bin der 1. Satz.</p>";
      // echo "<p>Ich bin der 2. Satz.</p>";
      echo "<p>Ich bin der 3. Satz.</p>";
    ?>
  </body>
</html>
<?php

```

- ① Der 2. echo-Befehl wurde mit einem -Kommentarzeichen auskommentiert. Diese Zeile wird dann nicht von PHP ausgeführt. Im Browser sieht man das Ergebnis, wenn ein Befehl auskommentiert ist.



Ausgabe der Beispieldatei „auskommentiert.php“

2.3 Daten im Browser anzeigen

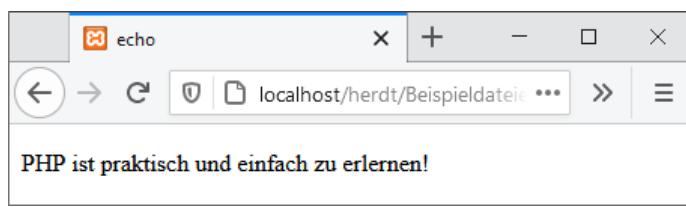
Der Befehl echo

PHP ist eine Programmiersprache, die auf dem Webserver ausgeführt wird. Das Ergebnis einer Berechnung kann z. B. in einer Variablen gespeichert werden. Die Ausführung von PHP-Befehlen führen nicht zwangsläufig zu einer Ausgabe im Browser oder zu generiertem HTML-Quelltext.

PHP kann jedoch Ausgaben erzeugen. Zeichen und mehrere Zeichen hintereinander (sogenannte Zeichenketten) werden über den Befehl echo in die HTML-Ergebnisdatei geschrieben und damit auf dem Bildschirm ausgegeben.

Syntax der echo-Anweisung

```
echo "<p>PHP ist praktisch und einfach zu erlernen!</p>";
```



Anzeige der Beispieldatei „echo.php“

Umgang mit HTML-Syntax bei der Ausgabe

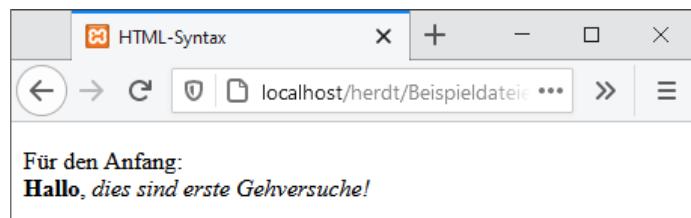
Die echo-Anweisung kann auch HTML-Tags enthalten.

- Schreiben Sie beliebige HTML-Tags einfach in die Anführungszeichen des Befehls echo.

Zeichenketten in Anführungszeichen – und dazu zählen auch HTML-Tags – werden an den Browser übergeben und vom Browser als normaler HTML-Code dargestellt.

Beispiel: *html1.php*

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>HTML-Syntax</title>
  </head>
  <body>
    <?php
      echo "<p>Für den Anfang: <br>";
      echo "<strong>Hallo</strong>, <em>dies sind erste
          Gehversuche!</em></p>";
    ?>
  </body>
</html>
```



Ausgabe der Beispieldatei „html1.php“

Beispiel: *html2.php*

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>HTML-Syntax</title>
  </head>
  <body>
    <?php
      echo '<p>Für den Anfang: <br>';
      echo '<strong>Hallo</strong>, <em>dies sind erste
          Gehversuche!</em></p>';
    ?>
  </body>
</html>
```

Der Quelltext *html2.php* ist fast der gleiche wie im Beispiel *html1.php*. Er unterscheidet sich dadurch, dass hier die Zeichenketten mit einfachen Anführungszeichen (Hochkommata) umschlossen sind. Beide Schreibweisen sind gültig, beide PHP-Skripte erzeugen die gleiche Ausgabe.

Der Unterschied von Zeichenketten in einfachen und doppelten Anführungszeichen ist relevant für den PHP-Interpreter und wie er mit diesen umgeht. Zeichenketten in doppelten Anführungszeichen werden geparsst (ausgewertet), sprich: der PHP-Interpreter „schaut“ in die Zeichenkette rein, findet er PHP-Variablen oder Steuerungszeichen, werden diese interpretiert.

Zeichenketten in einfachen Anführungszeichen werden hingegen nicht ausgewertet. Variablen und Steuerungszeichen werden nicht interpretiert und als solche ausgegeben (Im Browser erscheint z. B. `$zahl` statt dem Wert, welchen die Variable hat). Da das Parsen von Zeichenketten in einfachen Anführungszeichen wegfällt, wird diese Variante von PHP schneller verarbeitet.

Welche Variante Sie einsetzen, hängt davon ab, ob Sie Variablen innerhalb von Zeichenketten verwenden, die von PHP ausgewertet werden sollen. In dem Fall verwenden Sie doppelte Anführungszeichen. HTML-Code selbst gilt als normaler Text und wird so ausgegeben, wie er in der Zeichenkette steht, dieser muss nicht geparsst werden und kann von daher in einfachen Anführungszeichen stehen.

Fehler durch Anführungszeichen

Sind innerhalb einer Zeichenkette in doppelten Anführungszeichen weitere doppelte Anführungszeichen enthalten, beendet das erste doppelte Anführungszeichen innerhalb der Zeichenkette die Zeichenkette. Danach erwartet PHP weiteren PHP-Code, findet jedoch den Rest der Zeichenkette, was zu einem Fehler führt. Das gleiche gilt entsprechend für den Einsatz von einfachen Anführungszeichen.

Einen häufigen Fehler sehen Sie im folgenden – **falschen** – Beispiel:

```
Dieses Anführungszeichen innerhalb der Zeichenkette beendet die Zeichenkette.  
↓  
echo "Der "Eiffelturm" ist das Wahrzeichen von Paris."; // FALSCH!
```

Das Beispiel funktioniert nicht wie erwartet. Die Zeichenkette wird durch das Anführungszeichen vor dem Wort *Eiffelturm* beendet. Danach folgt bis zum Zeilenende normaler Text, den PHP nicht auswerten kann. Sie erhalten eine Fehlermeldung, die Sie auf einen Syntaxfehler aufmerksam macht.

Anführungszeichen ausgeben

Um dem Problem der Anführungszeichen zu begegnen, sieht PHP sogenannte **Escape-Sequenzen** vor. Das wichtigste Escape-Zeichen ist der Backslash \. Dieser **entwertet** Zeichen, die für die PHP-Syntax relevant sind:

Beispiel: sonderzeichen1.php

```
echo "<p>Der \"Eiffelturm\" ist das Wahrzeichen von Paris.</p>";  
echo "<p>Der 'Eiffelturm' ist das Wahrzeichen von Paris.</p>";  
echo '<p>Der \'Eiffelturm\' ist das Wahrzeichen von Paris.</p>';  
echo '<p>Der "Eiffelturm" ist das Wahrzeichen von Paris.</p>';
```

Escape-Sequenzen sind Zeichenkombinationen für Sonderzeichen, z. B. Anführungszeichen, Steuerzeichen oder Zeilenumbruch. Im obigen Beispiel verliert das Anführungszeichen durch den vorangestellten `\` seine Funktion als Begrenzer der Zeichenkette. Es wird nur das Anführungszeichen selbst übergeben und im Ergebnis dargestellt.

```
Der "Eiffelturm" ist das Wahrzeichen von Paris.  
Der 'Eiffelturm' ist das Wahrzeichen von Paris.  
Der 'Eiffelturm' ist das Wahrzeichen von Paris.  
Der "Eiffelturm" ist das Wahrzeichen von Paris.
```

Ausgabe der Beispieldatei „sonderzeichen1.php“

Beispiele für Escape-Sequenzen:

Escape-Sequenz	Ergebnis in der Zeichenkette	Hinweis
<code>\"</code>	"	Nur relevant, wenn die Zeichenkette von doppelten Anführungszeichen umschlossen ist
<code>\'</code>	'	Nur relevant, wenn die Zeichenkette von einfachen Anführungszeichen umschlossen ist
<code>\\"</code>	\	Nur relevant, wenn die Zeichenkette von doppelten Anführungszeichen umschlossen ist

Escape-Sequenz	Ergebnis in der Zeichenkette
<code>\n</code>	Zeilenumbruch, der nur im HTML-Quellcode der Ergebnisseite zu sehen ist. Dies dient der Übersichtlichkeit des Quellcodes, z. B. für eine Fehlersuche, da standardmäßig die gesamte Ausgabe eines PHP-Blocks nur eine Zeile im HTML-Code der Ergebnisseite einnimmt. Im Browser wird das <code>\n</code> als ein Leerzeichen angezeigt. Mehrere Leerzeichen und Zeilenumbrüche durch Steuerungszeichen im HTML-Code werden in der Browserdarstellung zu einem Leerzeichen zusammengefasst.

Sogenannte Steuerungszeichen wie `\n` für einen Zeilenumbruch im HTML-Quellcode werden nur innerhalb von doppelten Anführungszeichen vom PHP-Interpreter erkannt und ausgewertet. Steuerungszeichen in einfachen Anführungszeichen werden vom Browser als solche angezeigt und verlieren ihre beabsichtigte Funktion. Die Abbildung zeigt, dass Steuerungszeichen als einfache Anführungszeichen im Browser ausgegeben werden.

```
Der 'Eiffelturm' ist das Wahrzeichen von Paris.  
\n
```

Ausgabe der Beispieldatei „sonderzeichen2.php“

Beispiel: *sonderzeichen3.php*

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Sonderzeichen</title>
  </head>
  <body>
    <?php
①      echo "\n<h1>Personal Computer (\\"PC\\")</h1>\n";
②      echo "<p>'Windows' ist ein verbreitetes Betriebssystem.<br>\n";
③      echo "Installiert wird es meist im Verzeichnis
          C:\\Windows</p>\n";
    ?>
  </body>
</html>
```

- ① Wenn Sie den Ausgabebefehl echo mit doppelten Anführungszeichen verwenden, setzen Sie innerhalb der Zeichenkette einen Backslash \ vor die Anführungszeichen, damit das Zeichen für PHP **entwertet** und wie beabsichtigt angezeigt wird.



Ausgabe der Beispieldatei „sonderzeichen3.php“

- ② Alternativ können Sie auch einfache Anführungszeichen nutzen, um Text hervorzuheben. Wird die Zeichenkette durch doppelte Anführungszeichen begrenzt, beendet ein einfaches Anführungszeichen die Zeichenkette nicht. Dies gilt entsprechend umgekehrt, wenn die Zeichenkette mit einfachen Anführungszeichen umschlossen ist und Sie doppelte Anführungszeichen in der Zeichenkette verwenden.
- ③ Wenn Sie einen Backslash ausgeben möchten, setzen Sie einen zweiten Backslash davor.

Die nebenstehende Abbildung zeigt, wie übersichtlich der Quelltext der Datei ist. Dies kann Ihnen – vor allem bei längeren Skripten – bei der Fehlersuche helfen. Ohne Einsatz der im Beispiel verwendeten Escape-Sequenz \n würden Sie alle Ausgaben in einer Zeile finden.

```

1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="UTF-8">
5      <title>Sonderzeichen</title>
6    </head>
7    <body>
8
9   <h1>Personal Computer ("PC")</h1>
10  <p>'Windows' ist ein verbreitetes Betriebssystem.<br>
11  Installiert wird es meist im Verzeichnis C:\\Windows</p>
12
13 </body>
14 </html>
```

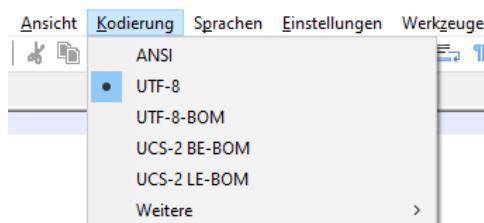
Quelltext der Beispieldatei „sonderzeichen3.php“

Deutsche Umlaute und ß – UTF-8-Zeichensatz

Seit PHP 5.6 ist der Standardzeichensatz für PHP UTF-8. Dieser Zeichensatz umfasst alle deutschen Umlaute. Das bedeutet, Sie können ohne weiteres ä, ö und ß einsetzen.

In älteren PHP-Versionen wird der Standardzeichensatz ISO 8859-1 verwendet. Je nach Angabe im charset-Metatag des HTML-Dokuments kann es zu Problemen bei der Darstellung von Umlauten kommen. Um eine fehlerhafte Darstellung zu vermeiden, können Sie statt der Umlaute und des ß-Zeichens die entsprechenden HTML-Entities (ä, ü usw.) verwenden.

Achten Sie darauf, dass die PHP-Dateien selbst mit dem UTF-8-Zeichensatz angelegt sind. Diese Einstellung kann im Editor Notepad++ über die obere Menüleiste vorgenommen werden. Verwenden Sie die UTF-8-Option (nicht UTF-8-BOM). Der BOM (Byte Order Mark) ist ein drei Bytes großes Zeichen am Anfang der Datei und kann in Browsern zu Darstellungsproblemen führen.



Einstellung des UTF-8-Zeichensatzes der PHP-Dateien über den Editor Notepad++

Ist die PHP-Datei selbst mit einem anderen Zeichensatz erstellt, kann es bei Umlauten zu Darstellungsproblemen kommen. Selbst wenn Sie den korrekten `<meta charset="UTF-8">` angegeben haben, verursachen Sonderzeichen mit einem anderen Zeichensatz häufig eine fehlerhafte Darstellung. Die folgenden Quelltextbeispiele zeigen einerseits die Problematik, aber auch Lösungen dafür. Generell gilt: Solange Sie die PHP-Dateien mit dem UTF-8-Zeichensatz angelegt haben und die Einstellungen von PHP in der Standardeinstellung belassen, ist beim Einsatz von Umlauten und dem ß keine Sonderbehandlung notwendig.

Beispiel: zeichensatz1.php (Datei mit falschem Zeichensatz, nicht UTF-8)

```
<!DOCTYPE html>
<html>
  <head>
    <title>falscher Zeichensatz 1</title>
  </head>
  <body>
    <?php
      ① echo "<p>Viel Spaß im Frühling!<br>\n";
      ② echo htmlentities("Viel Spaß im Frühling!", ENT_QUOTES,
                           "ISO-8859-1");
      echo "\n</p>";
    ?>
  </body>
</html>
```

- ① Die Datei `zeichensatz1.php` ist mit einem falschen Zeichensatz angelegt. Hier wird eine Zeile mit Sonderzeichen ausgegeben. Im folgenden Screenshot ist die Auswirkung in der ersten Zeile zu sehen, die Ausgabe ist fehlerhaft.

- ② Über die PHP-Funktion `htmlentities()` werden Umlaute und das ß in sogenannte HTML-Entities umgewandelt. Das sind Zeichen, die jeder Browser kennt und entsprechend richtig – und das unabhängig von einem Zeichensatz – anzeigt.

! Seit PHP 5.6 ist bei der Funktion `htmlentities` der Standardwert für den 3. Parameter, welcher für die Zeichenkodierung verantwortlich ist, an den `default_charset` angepasst worden. Der `default_charset` ist ebenfalls seit PHP 5.6 UTF-8, was bedeutet, dass bei der Umwandlung von Sonderzeichen in HTML-Entities eine UTF-8-Zeichenkette erwartet wird. Je nach Zeichensatz der PHP-Datei wandelt `htmlentities` Sonderzeichen gar nicht oder falsch um.

The screenshot shows a browser window titled "falscher Zeichensatz 1" displaying the text "Viel Spaß im Fröhling!". Below it is the browser's developer tools showing the source code:

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>falscher Zeichensatz 1</title>
5   </head>
6   <body>
7     <p>Viel Spaß im Fröhling!<br>
8   Viel Spässlig; im Früuml;hling!
9   </p> </body>
10 </html>

```

A callout box labeled "HTML-Entities" points to the character entities in the source code.

Fehlerhafte Anzeige von Sonderzeichen

Rechts im Quelltext sind in der Zeile 8 die Sonderzeichen als HTML-Entities zu sehen.

Beispiel: `zeichensatz2.php` (Datei mit falschem Zeichensatz, nicht UTF-8)

The screenshot shows a code editor with the following PHP code:

```

<!DOCTYPE html>
<html>
  <head>
    <title>falscher Zeichensatz 2</title>
  </head>
  <body>
    <?php
①      echo utf8_encode("<p>Viel Spaß im Frühling!</p>") ;
    ?>
  </body>
</html>

```

A circled number ① points to the line where `utf8_encode` is called.

Auch die PHP-Datei `zeichensatz2.php` ist mit dem falschen Zeichensatz angelegt.

- ① Über die Funktion `utf8_encode()` werden Nicht-UTF-8-Zeichen in UTF-8-Zeichen umgewandelt. Damit wird die korrekte Ausgabe des Satzes mit Umlauten und ß sicher gestellt. In der untenstehenden Abbildung links ist die fehlerfreie Darstellung zu sehen. In der Abbildung rechts können Sie sehen, dass die Umlaute und das ß auch im Quelltext richtig angezeigt werden.

The screenshot shows a browser window titled "falscher Zeichensatz 2" displaying the text "Viel Spaß im Frühling!". Below it is the browser's developer tools showing the source code:

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>falscher Zeichensatz 2</title>
5   </head>
6   <body>
7     <p>Viel Spaß im Frühling!</p> </body>
8 </html>

```

Die Zeichensatzproblematik tritt häufig dann auf, wenn Daten aus anderen Quellen (z. B. aus Datenbanken) stammen, auf deren Zeichensatz Sie keinen Einfluss haben. In dem Fall haben Sie mit diesen beiden Beispielen Lösungen für das Problem an der Hand. Röhrt das Problem aus einem falschen Zeichensatz einer Datei, ist die Umwandlung der Datei in eine UTF-8-Datei zu empfehlen. Dies erspart Ihnen umständliches Umwandeln von Zeichenketten.

Alle Beispieldateien, die im Buch verwendet werden und als Download bereitstehen (mit Ausnahme der beiden zuletzt vorgestellten) sind mit dem Zeichensatz UTF-8 erstellt. Das bedeutet, Sonderzeichen können als ä, ö, ü und ß im Quellcode verwendet werden, eine besondere Behandlung ist nicht notwendig.

2.4 Grundlagen zur Fehlersuche in PHP-Skripten

Fehlerarten in PHP

PHP kennt drei Kategorien von Fehlern:

Bezeichnung	Relevanz	Erläuterung
Fehler	Schwerwiegend – sofort beheben PHP-Skripte mit Fehlern brechen das Skript an der Stelle ab oder werden erst gar nicht ausgeführt, falls diese nicht explizit abgefangen werden (erst ab PHP 7.0 möglich, siehe weiter unten)	Am häufigsten treten schwerwiegende Fehler (<i>fatal error</i>) und Fehler bei der Analyse des PHP-Codes (<i>parse error</i>) auf. Eine Fehlermeldung wird ausgegeben. ✓ Bei schwerwiegenden Fehlern wird das Skript bis an diese Fehlerstelle ausgeführt. ✓ Bei einem <i>parse error</i> entdeckt der PHP-Interpreter den Fehler sofort und führt das Skript nicht aus.
Warnung	Wichtig – zu beachten, möglichst sofort beheben PHP-Skripte mit Warnungen werden trotzdem bis zu Ende ausgeführt.	Warnungen (<i>warning</i>) deuten meist auf schwere Fehler hin, die sofort behoben werden sollten. Es kann passieren, dass Ihr Skript nicht mehr wie gewünscht funktioniert. Eine Fehlermeldung wird ausgegeben, das Skript wird jedoch weiter ausgeführt.
Benachrichtigung	Gut zu wissen – Behebung empfohlen Benachrichtigungen haben keinen Einfluss auf die Ausführung von PHP-Skripten.	Benachrichtigungen (<i>notice</i>) stören selten den Ablauf des Skripts. Im Sinne einer „sauberen“ Programmierung sollten Sie allerdings die Ursache der <i>notice</i> beheben.

Seit PHP 5.3 werden bei der PHP-Standardinstallation Fehler, Warnungen und Benachrichtigungen im Browser angezeigt. Fehler lassen sich meist schnell finden, da neben der eigentlichen Meldung sowohl der Dateiname einschließlich Dateipfad sowie die Zeilennummer ausgegeben werden, in der der Fehler aufgetreten ist.

Treten Fehler auf, werden im Browser die englischen Bezeichnungen der Fehlerkategorien angezeigt:

- ✓ *error*
- ✓ *warning*
- ✓ *notice*

Wie können Fehler entstehen?

Fehler und Fehlerbehebung sind bei der PHP-Programmierung ein Teil des täglichen Entwicklungsprozesses. Häufig handelt es sich um Tipp- oder Flüchtigkeitsfehler. Beispiele hierfür sind:

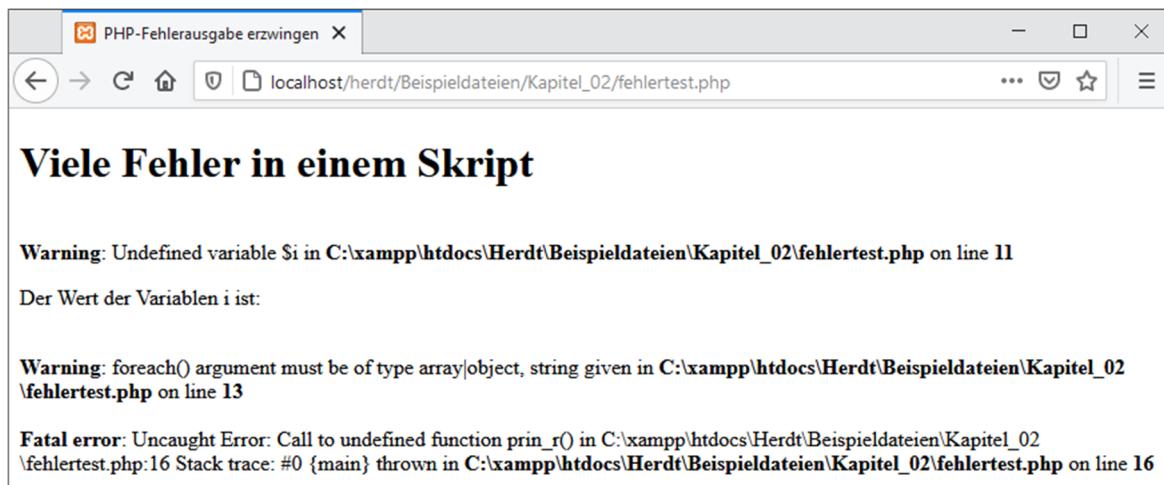
- ✓ fehlendes Semikolon zum Abschluss eines Befehls;
- ✓ fehlendes \$-Zeichen bei der Verwendung von Variablen, z. B. `$stadt` anstelle von `$$stadt`, oder ein falsches Zeichen, z. B. `§stadt` anstelle von `$stadt`;
- ✓ sonstige Syntaxfehler, z. B. Schreibfehler wie `eco` statt `echo`;
- ✓ fehlendes Komma oder fehlende Klammer.

Beispiel: `fehlertest.php`

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>PHP-Fehlerausgabe erzwingen</title>
    </head>
    <body>
        <h1>Viele Fehler in einem Skript</h1>
        <?php
①      error_reporting(E_ALL); // PHP-Standardeinstellung
②      echo "<p>Der Wert der Variablen i ist: " . $i . "</p>";
        // Variable nicht definiert: warning
③      $a = 'Katze';
        foreach ($a as $b => $c) { // falscher Variablentyp: warning
            echo "<p>$c</p>";
        }
④      prin_r($i); // Befehl falsch geschrieben: fatal error
        ?>
    </body>
</html>
```

- ① Über den Schalter `E_ALL` für die Funktion `error_reporting()` weisen Sie PHP an, Fehler aller Kategorien `error`, `warning` und auch `notices` anzuzeigen (`E_ALL` ist seit PHP 5.4 die Standardeinstellung für `error_reporting()`). Sprich: auch ohne die Zeile ① in dem PHP-Code werden `error`, `warning` und `notices` angezeigt.
- ② Durch die Verwendung der **nicht** definierten Variablen `$i` wurde vor PHP 8.0 eine Fehlermeldung der Kategorie `notice` angezeigt. Durch die Veränderung der Fehlerlevel in PHP 8.0 erzeugt dies nun eine Fehlermeldung der Kategorie `warning`. Durch die korrekte Deklaration (Bekanntgabe der Variablen) und Initialisierung (Wertzuweisung) z. B. `$i = 0` vor der Verwendung der Variablen wird der Fehler behoben und damit die Fehlermeldung beseitigt. Grundlagen zu Variablen in PHP finden Sie im folgenden Kapitel.
- ③ Die Variable `$a` wird mit einer Zeichenkette initialisiert, die Kontrollstruktur `foreach()` erwartet jedoch ein Array oder ein Objekt. Dadurch entsteht eine Fehlermeldung der Kategorie `warning`.

- ④ Eine falsche Schreibweise von `prin_r()` (`print_r` wäre richtig) erzeugt einen schwerwiegenden Fehler der Kategorie *fatal error*. Ein *fatal error* führt dazu, dass der PHP-Interpreter das Skript nicht weiter ausführt, er bricht die Verarbeitung des Skriptes an dieser Stelle sofort ab. Es sei denn, dieser wird über ein Error-Handling abgefangen, was mit PHP 7.0 eingeführt wurde.



Ausgabe Beispieldatei „fehlertest.php“ (Windows)

Hinweise zur Fehlersuche

Im Beispiel meldet PHP die jeweilige Zeile, in der ein Fehler aufgetreten ist. Zum Auffinden des Fehlers schauen Sie sich die Zeile an, die Ihnen in der Fehlermeldung mitgeteilt wird. Es kann vorkommen, dass die Fehlermeldung auf eine bestimmte Zeile hinweist, der Fehler jedoch in den Zeilen davor verursacht wurde, beispielsweise aus einer zuvor falsch erstellten Wertzuweisung. Finden Sie in der angegebenen Zeile keinen Fehler, suchen Sie in den vorhergehenden Zeilen (notfalls bis zum Dateianfang). In den meisten Fällen finden Sie jedoch den Fehler in der angegebenen Zeile.

Über `error_reporting(E_ERROR | E_WARNING | E_PARSE)` können Sie die Fehlermeldungen auf die wichtigsten Meldungen einschränken. Alternativ können Sie über `error_reporting(E_ALL ^ E_NOTICE)` alle Meldungen um die *notice*-Meldungen reduzieren. Das `^`-Zeichen vor `E_NOTICE` bedeutet "NICHT".

! Für Webseiten, die online sind, sollten Sie die Anweisung `error_reporting(0);` verwenden. Dieser PHP-Befehl mit dem Parameter 0 deaktiviert alle Fehlermeldungen. Fehlermeldungen irritieren einerseits die Webseitenbesucher, andererseits liefern sie potenziell wertvolle Informationen über Ihre Programmierung, was ein Sicherheitsrisiko darstellen kann. Auf dem Entwicklungssystem sollten hingegen möglichst alle Meldungen aktiviert sein, damit diese erkannt und behoben werden können.

Fehlerbehandlung ab PHP 7.0

Eine wesentliche Neuerung in PHP 7.0 war das **Throwable Interface**. Dies ermöglicht, auch *fatal error* abzufangen. Schwerwiegende Fehler führten in älteren PHP-Versionen **immer** zum Abbruch des PHP-Skriptes. Der Abbruch erfolgt ohne Fehlermeldung für den Endanwender, entweder mit weißem Browserfenster, wenn die Fehlerausgabe in dem Browser deaktiviert war, oder mit einem technischen Hinweis zum Fehler, welcher schlimmstenfalls z. B. Datenbank-Passwörter enthielt.

Über **Throwable** können auch schwerwiegende Fehler abgefangen werden. Sie können an Stellen, an denen Sie potenziell einen Fehler erwarten, einen sogenannten `try-catch`-Block einbinden, den Fehler abfangen, den weiteren Skriptverlauf entsprechend steuern und dem Nutzer eine neutrale Fehlermeldung anzeigen.

Die Fehlerbehandlung per Exception-Handler und das Throwable Interface sind kein Teil der PHP-Grundlagen.

2.5 Übung

Grundlegende Sprachelemente

Level		Zeit	ca. 10 min
Übungsinhalte	<ul style="list-style-type: none"> ✓ PHP in HTML einbinden ✓ Anführungszeichen ✓ Einsatz von Escape-Sequenzen (Steuerungszeichen) ✓ Ausgabe von Inhalten im Browser ✓ Kommentare 		
Übungsdatei	--		
Ergebnisdatei	<i>ausgabe.php</i>		

1. Schreiben Sie ein PHP-Skript, das eine Webseite erstellt. Die Webseite soll einige Informationen über Ihren Lieblingssportler bzw. -künstler enthalten. Verwenden Sie eine HTML-Überschrift wie z. B. *Gitarrenlegende Eric Clapton*.
 2. Wechseln Sie dann in einen PHP-Block: Mithilfe des Befehls echo soll die Seite in mehreren Zeilen Informationen über für Sie interessante Eckdaten der ausgewählten Person enthalten, z. B. beste CDs, Geburtstag etc.
- Bauen Sie in die Zeichenkette Sonderzeichen ein, wie z. B. Anführungszeichen. Verwenden Sie das p-Tag, welches HTML für Textblöcke (Paragraph) vorsieht. Verwenden Sie Steuerungszeichen, um den HTML-Code übersichtlicher darzustellen.
3. Kommentieren Sie Ihr PHP-Skript ausführlich.
 4. Formatieren Sie einen Satz innerhalb der echo-Befehle in einer anderen Farbe und in Fettdruck.
 5. Speichern Sie die Datei unter dem Namen *ausgabe.php* und rufen Sie die Seite in Ihrem Browser auf.
 6. Bauen Sie im Kapitel genannte mögliche Fehler ein und prüfen Sie die auftretenden Fehlermeldungen im Browser.



Beispiel-Ergebnisdatei „ausgabe.php“

3

Variablen und Operatoren



Beispieldateien: Dateien im Ordner *Kapitel_03*

3.1 Variablen

Mit Variablen arbeiten

Variablen dienen dazu, Informationen zu speichern, die für die weitere Ausführung des Programms notwendig sind. In PHP ergibt sich der Datentyp einer Variablen automatisch durch den Datentyp des Wertes, welcher der Variablen zugewiesen wird. Anders als in einigen anderen Programmiersprachen müssen Sie den Datentyp **nicht** definieren. Der Datentyp einer Variablen kann sich auch im laufenden Skript ändern, ohne dass es zu einem Fehler kommt.

Datentypen in PHP

PHP kennt folgende Datentypen:

Datentyp	Bezeichnung	Beispiel
Wahrheitswert	<i>bool</i> (auch <i>boolean</i>)	<code>true</code> (wahr) oder <code>false</code> (falsch)
Ganzzahl	<i>int</i> (auch <i>integer</i>)	42 oder -23
Fließkommazahl	<i>float</i> (auch <i>double</i>)	1.95883 oder -207.14
Zeichenkette	<i>String</i>	"HERDT-Verlag Bodenheim" oder 'Andreas'
ohne Wert	<i>NULL</i>	ohne Wert – einziger möglicher Wert: <code>NULL</code>
Array (ein- oder mehrdimensional)	<i>Array</i>	("Frankfurt", "Berlin", "Zürich") oder ("England" => "London", Frankreich" => "Paris")
Ressource	<i>Resource</i>	Referenz auf eine Ressource, wie z. B. eine Datenbankverbindung
Objekt	<i>Object</i>	wird in der objektorientierten Programmierung einer Klasseninstanz zugeordnet
Callbacks	<i>Callable</i>	Funktionsname als Referenz auf eine Funktion
Iterables	<i>Iterables</i>	Pseudotyp. Ein beliebiger Wert, der mit einer <code>foreach()</code> -Schleife durchlaufen werden kann

Namensgebung bei Variablen

Der Name einer Variablen muss in PHP immer mit dem Dollarzeichen `$` beginnen. Daran erkennt PHP, dass es sich um eine Variable handelt. Für die Benennung von Variablen gelten folgende Regeln:

Eine Variable

- ✓ darf nur aus Buchstaben, Ziffern und dem Unterstrich `_` bestehen. Andere Sonderzeichen sind nicht erlaubt.
- ✓ muss mit einem Buchstaben oder dem Unterstrich `_` beginnen (z. B. `$miete` oder `$_miete`). Danach kann eine beliebige Anzahl Buchstaben, Ziffern oder Unterstriche folgen.
- ✓ darf keine Leerzeichen enthalten.
- ✓ kann Groß- und Kleinbuchstaben enthalten, wobei zwischen Groß- und Kleinschreibung unterschieden wird, z. B. `$PrimZahl` ist nicht gleich `$primzahl`.
- ✓ sollte nach Möglichkeit keine Umlaute oder `ß` enthalten.
- ✓ Besteht ein Variablenname aus mehreren Begriffen, können die einzelnen Begriffe durch Unterstriche voneinander getrennt werden, z. B. `$post_versand_gebuehr`. Gebräuchlich ist auch die sogenannte CamelCase-Schreibweise, bei der jeder Begriff mit einem Großbuchstaben beginnt, z. B. `$PostVersandGebuehr`. Die Schreibweise sollte für alle Variablen einheitlich gewählt werden.
- ✓ Durch das Voranstellen eines einzelnen Buchstabens kann auf den Datentyp hingewiesen werden, wofür die Variable verwendet wird, z. B. `$iNummer` (`i` für `int`), `$fPreis` (`f` für `float`) oder `$sOrt` (`s` für `string`).
- ✓ sollte nicht identisch sein mit einem sogenannten **reservierten Wort**.

Reservierte Wörter (PHP-Keywords)

In PHP sind bestimmte Zeichenketten bzw. Wörter als Schlüsselwörter definiert. Diese können zwar als Variablenname verwendet werden, jedoch sollten Sie dies vermeiden, da Keywords als Variablenbezeichnung PHP-Code unnötig verkomplizieren. Als Konstantennamen (vgl. Abschnitt 3.4), Funktionsnamen (vgl. Kapitel 7) oder Klassennamen sind reservierte Wörter verboten.

Liste reservierter Schlüsselwörter				
<code>__halt_compiler()</code>	<code>abstract</code>	<code>and</code>	<code>array()</code>	<code>as</code>
<code>break</code>	<code>callable</code>	<code>case</code>	<code>catch</code>	<code>class</code>
<code>clone</code>	<code>const</code>	<code>continue</code>	<code>declare</code>	<code>default</code>
<code>die()</code>	<code>do</code>	<code>echo</code>	<code>else</code>	<code>elseif</code>
<code>empty()</code>	<code>enddeclare</code>	<code>endfor</code>	<code>endforeach</code>	<code>endif</code>
<code>endswitch</code>	<code>endwhile</code>	<code>eval()</code>	<code>exit()</code>	<code>extends</code>
<code>final</code>	<code>finally</code>	<code>fn</code>	<code>for</code>	<code>foreach</code>
<code>function</code>	<code>global</code>	<code>goto</code>	<code>if</code>	<code>implements</code>
<code>include</code>	<code>include_once</code>	<code>instanceof</code>	<code>insteadof</code>	<code>interface</code>

Liste reserverter Schlüsselwörter				
isset()	list()	match (neu in PHP 8.0)	namespace	new
or	print	private	protected	public
require	require_once	return	static	switch
throw	trait	try	unset()	use
var	while	xor	yield	yield from

Deklaration (Bekanntgabe) und Initialisierung (Wertzuweisung)

```
$testiable = 5;
```

Über diese Codezeile wird in PHP eine Variable deklariert und initialisiert. Die **Deklaration** ist die Namensvergabe, also die Bekanntmachung der Variable, die **Initialisierung** ist die Wertzuweisung. Diese Wertzuweisung wird über das Gleichzeichen (Zuweisungsoperator) `=` durchgeführt.

Anders als in anderen Programmiersprachen werden Variablen in PHP ohne Datentyp-Definitionen deklariert. Es wird vorher **nicht** explizit festgelegt, welcher Datentyp (z. B. `int`, `boolean`, `string`) in einer Variable vorkommen darf.

PHP „untersucht“ den Wert, welcher der Variablen zugewiesen wird und vergibt entsprechend automatisch den passenden Datentyp. Worte zum Beispiel erzeugen eine Variable vom Datentyp `string`, eine Ganzzahl vom Datentyp `int`. Weisen Sie innerhalb eines PHP-Skripts einer bereits definierten Variablen den Wert eines anderen Datentyps zu, passt PHP den Datentyp automatisch an.

Details zu diesem automatischen Verhalten finden Sie am Ende von Kapitel 3.3.

3.2 Variablen und Operatoren für Zahlen

Mit numerischen Datentypen arbeiten

Die numerischen Datentypen werden in Ganzzahl- und Fließkommazahl-Datentypen unterteilt. Sie werden für Berechnungen, Aufzählungen und Nummerierungen eingesetzt. Ganzzahlen, also Zahlen ohne Nachkommastellen, werden als Integer (Datentyp `int`) bezeichnet. Zahlen mit Nachkommastellen (Fließkommazahl, auch Gleitkommazahl) werden als `float` (auch `double`) bezeichnet.

In allen Beispielen ab diesem Kapitel werden die HTML-Tags `<!DOCTYPE html>`, `<html>`, `<head>`, `<meta>`, `<title>` und `<body>` nicht mit abgedruckt. Sie sind für das Verständnis der PHP-Skripte nicht relevant. Die Beispieldateien zum Buch enthalten diese Tags.

Beispiel: `preis.php`

Im folgenden Beispiel wird der Preis für den Einkauf von Äpfeln berechnet:

①	<pre><?php \$preis_apfel = 2.59;</pre>
---	---

```

② $menge = 4;
③ $gesamtpreis = $preis_apfel * $menge;
④ echo $gesamtpreis;
?>

```

- ① Es wird die Variable `$preis_apfel` angelegt und der Zahlenwert `2.59` zugewiesen.
`$preis_apfel` wird damit zu einer Variablen mit dem Datentyp *float* (Fließkommazahl).

! PHP verwendet den Punkt als Dezimaltrennzeichen (englische Notation), die deutsche Notation von Fließkommazahlen mit einem Komma ist in PHP nicht zulässig und führt zum Fehler.

- ② Die Variable `$menge` wird mit dem Zahlenwert `4` initialisiert. Die Variable `$menge` ist damit eine Variable mit dem Datentyp *int* (Ganzzahl).
③ Die Variable `$gesamtpreis` wird eingeführt. Ihr wird das Ergebnis aus der Multiplikation von `$preis_apfel` und `$menge` zugewiesen. Die Variable `$gesamtpreis` wird aufgrund des Rechenergebnisses ebenfalls zu einer Variablen vom Datentyp *float* (Fließkommazahl).
④ Über den Befehl `echo` wird der Wert der Variablen `$gesamtpreis` ausgegeben.

Arithmetische Operatoren

Mit arithmetischen Operatoren können mathematische Berechnungen durchgeführt werden. Sie erwarten entweder Ganzzahl- bzw. Fließkommazahl-Variablen oder feste Werte als Parameter und liefern ein numerisches Ergebnis zurück. Sie können folgende Operatoren verwenden:

Operator	Name	Bedeutung	Beispiel	Wert nach der Operation
<code>+</code>	Addition	<code>\$a + \$b</code> ergibt die Summe von <code>\$a</code> und <code>\$b</code> .	<code>\$a = 10;</code> <code>\$b = 2;</code> <code>\$c = \$a + \$b;</code>	<code>\$c = 12</code>
<code>-</code>	Subtraktion	<code>\$a - \$b</code> ergibt die Differenz von <code>\$a</code> und <code>\$b</code> .	<code>\$a = 10;</code> <code>\$b = 2;</code> <code>\$c = \$a - \$b;</code>	<code>\$c = 8</code>
<code>*</code>	Multiplikation	<code>\$a * \$b</code> ist das Produkt aus <code>\$a</code> und <code>\$b</code> .	<code>\$a = 10;</code> <code>\$b = 2;</code> <code>\$c = \$a * \$b;</code>	<code>\$c = 20</code>
<code>/</code>	Division	<code>\$a / \$b</code> ist der Quotient von <code>\$a</code> und <code>\$b</code> .	<code>\$a = 10;</code> <code>\$b = 2;</code> <code>\$c = \$a / \$b;</code>	<code>\$c = 5</code>
<code>**</code>	Potenz	<code>\$a ** \$b</code> ist die Potenz aus <code>\$a</code> (Basis) hoch <code>\$b</code> (Exponent).	<code>\$a = 10;</code> <code>\$b = 2;</code> <code>\$c = \$a ** \$b;</code>	<code>\$c = 100</code>
<code>%</code>	Modulo	<code>\$a % \$b</code> ist der Rest der ganzzahligen Division von <code>\$a</code> und <code>\$b</code> .	<code>\$a = 10;</code> <code>\$b = 3;</code> <code>\$c = \$a % \$b;</code>	<code>\$c = 1</code>
<code>++</code>	Präinkrement	<code>++\$a</code> erhöht die Variable <code>\$a</code> um 1 vor der weiteren Verwendung.	<code>\$a = 10;</code> <code>\$b = 2;</code> <code>\$c = ++\$a + \$b;</code>	<code>\$a = 11</code> <code>\$c = 13</code>

Operator	Name	Bedeutung	Beispiel	Wert nach der Operation
--	Prädekrement	--\$a verringert die Variable \$a um 1 vor der weiteren Verwendung.	\$a = 10; \$b = 2; \$c = --\$a + \$b;	\$a = 9 \$c = 11
++	Postinkrement	\$a++ erhöht die Variable \$a um 1 nach der Verwendung.	\$a = 10; \$b = 2; \$c = \$a++ + \$b;	\$a = 11 \$c = 12
--	Postdekrement	\$a-- verringert die Variable \$a um 1 nach der Verwendung.	\$a = 10; \$b = 2; \$c = \$a-- + \$b;	\$a = 9 \$c = 12
+=	Zuweisungsoperator	\$a += \$b weist der Variablen \$a den Wert \$a + \$b zu (Kurzschreibweise für \$a = \$a + \$b).	\$a = 10; \$a += 5;	\$a = 15
-=	Zuweisungsoperator	\$a -= \$b ist die Kurzschreibweise für \$a = \$a - \$b.	\$a = 10; \$a -= 5;	\$a = 5
*=	Zuweisungsoperator	\$a *= \$b ist die Kurzschreibweise für \$a = \$a * \$b.	\$a = 10; \$a *= 5;	\$a = 50
/=	Zuweisungsoperator	\$a /= \$b ist die Kurzschreibweise für \$a = \$a / \$b.	\$a = 10; \$a /= 5;	\$a = 2

Werden mehrere Berechnungen in einer Zuweisung durchgeführt (z. B. \$a = \$b - \$c * \$d), werden die üblichen mathematischen Rechenregeln angewandt:

- ✓ Bearbeitung der Berechnung von links nach rechts
- ✓ Punkt- vor Strichrechnung
- ✓ Geklammerte Ausdrücke werden zuerst ausgewertet.

Beispiel: berechnung.php

```

① <?php
    $preis_apfel = 2.59;
    $menge_jonagold = 4;
    $menge_idared = 10;
    $menge_elstar = 15;
    ② $gesamtmenge = $menge_jonagold + $menge_idared + $menge_elstar;
    ③ $gesamtpreis = $gesamtmenge * $preis_apfel;
    ④ echo $gesamtpreis;
?>

```

- ① Den Variablen `$preis_apfel`, `$menge_jonagold`, `$menge_idared` und `$menge_elstar` werden Werte zugewiesen. Verwenden Sie möglichst aussagekräftige Variablenbezeichnungen. Dies erhöht die Nachvollziehbarkeit Ihrer Skripte.
- ② Die Addition der Variablen `$menge_jonagold`, `$menge_idared` und `$menge_elstar` ergibt 29. Dieser Wert wird der neuen Variablen `$gesamtmenge` zugewiesen und somit zwischengespeichert.
- ③ Die Variable `$gesamtmenge` wird mit dem Preis `$preis_apfel` multipliziert (ergibt 75.11) und der Variablen zugewiesen. Mehrere Berechnungen, wie hier die Addition ② und die Multiplikation ③, können Sie in mehrere Schritte aufteilen. Das macht den PHP-Code verständlicher.
- ④ Abschließend wird der Wert der Variablen `$gesamtpreis` per `echo` im Browser ausgegeben.

3.3 Variablen und Operatoren für Zeichenketten

Mit Zeichen-Datentyp (*string*) arbeiten

Der Zeichen-Datentyp kann beliebige Zeichen des erweiterten Unicode-Zeichensatzes enthalten. Der in PHP verwendete Zeichen-Datentyp ist die Zeichenkette, auch *string* genannt. Zeichenketten werden bei der Wertzuweisung in Anführungszeichen bzw. Hochkommata eingeschlossen.

Zeichenkettenoperator – Konkatenation

Sie können mehrere Zeichenketten oder Zahlen (Ganz- oder Fließkommazahlen) und Zeichenketten über den Zeichenkettenoperator `.` miteinander verknüpfen. Der Fachbegriff für die Zeichenkettenverknüpfung ist **Konkatenation**. Das Ergebnis einer Konkatenation ist immer ein Wert vom Datentyp *string*.

Operator	Bedeutung	Beispiel
<code>.</code>	Verketten von Zeichenketten	<code>\$a = "Hamburg ist ";</code> <code>\$b = "eine schöne";</code> <code>\$c = \$a . \$b . " Stadt.";</code> <code>echo \$c;</code>
<code>.=</code>	Anhängen einer Zeichenkette an eine bereits vorhandene Variable	<code>\$a = "Hamburg ist";</code> <code>\$a .= " eine schöne";</code> <code>\$a .= " Stadt.";</code> <code>echo \$a;</code>

In beiden Beispielen erhalten Sie als Ausgabe *Hamburg ist eine schöne Stadt*. Sie sehen auch, dass Sie sich innerhalb der Zeichenketten selbst um die Leerzeichen am Übergang der einzelnen Zeichenketten kümmern müssen.

Geringere Priorität der Konkatenation seit PHP 8.0

Vor PHP 8.0 hatte die Konkatenation die gleiche Priorität wie eine Addition bzw. Subtraktion. Nun gilt: Strichrechnung vor Konkatenation. Das bedeutet, dass dieselbe Schreibweise in PHP 7 und davor etwas anderes ergibt als in PHP 8.

Beispiel: *konkatenation.php*

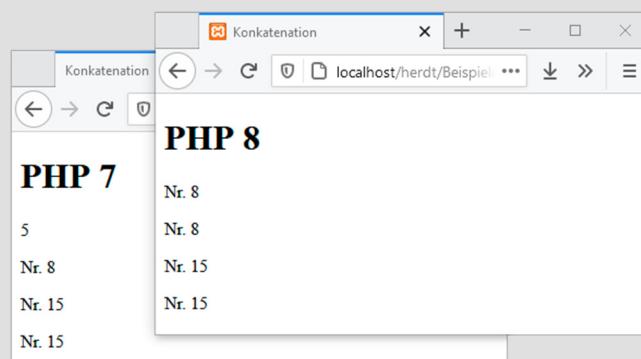
```

① $a = 3;
    $b = 5;
② echo '<p>Nr. ' . $a + $b . '</p>';
③ echo '<p>Nr. ' . ($a + $b) . '</p>';
④ echo '<p>Nr. ' . $a * $b . '</p>';
⑤ echo '<p>Nr. ' . ($a * $b) . '</p>';

```

- ① Die Variablen \$a und \$b werden deklariert und jeweils mit einer Zahl initialisiert.
- ② Hier werden die beiden Variablen addiert – im Browser wird *Nr. 8* ausgegeben.
- ③ In Zeile ③ werden durch die verwendeten Klammern zuerst die Variablen addiert und dann mit den Zeichenketten verbunden. Damit Ihre PHP-Skripte abwärtskompatibel sind, verwenden Sie diese Schreibweise, auch wenn die Klammern in PHP 8 nicht mehr notwendig sind.
- ④ Hier greift die Priorität Punktrechnung vor Konkatenation bzw. Strichrechnung. PHP multipliziert zuerst die beiden Variablen und führt dann die Konkatenation durch. Im Browser erscheint *Nr. 15*.
- ⑤ Die Zeile entspricht der Zeile ④. Hier können Sie getrost die Klammern weglassen, da die Punktrechnung immer vor der Konkatenation durchgeführt wird.

! Vor PHP 8 hatten Konkatenation und Addition bzw. Subtraktion die gleiche Priorität. Bis PHP 7 wird die Zeile ② wie folgt verarbeitet:



Dasselbe PHP-Skript liefert je nach PHP-Version unterschiedliche Ergebnisse.

Zuerst wird die Zeichenkette '*<p>Nr.* ' und die Variable \$a zu einer neuen Zeichenkette '*<p>Nr. 3*' verbunden. Danach führt PHP eine Addition mit der Zeichenkette '*<p>Nr. 3*' und der Variablen \$b durch. Bei einer Addition mit einer Zeichenkette ist der Wert dieser Zeichenkette 0 (vor PHP 8). Die Addition von 0 und 5 ergibt den *int* 5. Danach folgt die Konkatenation mit der Zeichenkette '*</p>*', im Browser wird 5 ausgegeben. Im Quelltext steht *5</p> - '<p>Nr. 3'* wurde durch die Addition als 0 zur 5 addiert.

Ausgabe von Variablen

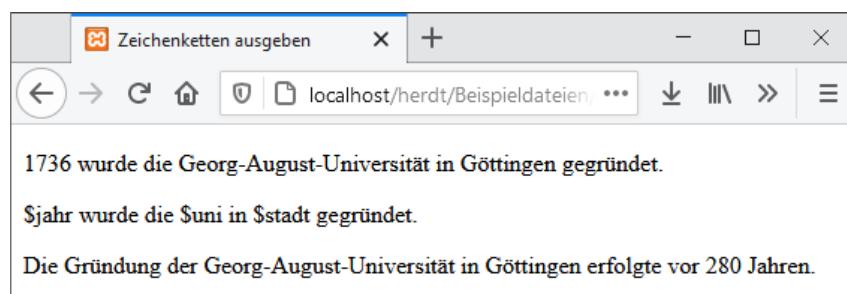
Bei der Ausgabe einer Variablen kann deren Wert oder deren Bezeichnung ausgegeben werden. Ob der Wert oder der Variablenamen angezeigt wird, hängt von der Einbettung der Variable im PHP-Code ab.

Beispiel: var_ausgabe.php

Hier werden die verschiedenen Ausgabemöglichkeiten aufgezeigt.

```
<?php
① $stadt = "Göttingen";
$uni = "Georg-August-Universität";
$jahr = 1736;
$heute = 2016;
// Wert der Variablen werden angezeigt
② echo "<p>$jahr wurde die $uni in $stadt gegründet.</p>";
// Name der Variablen werden angezeigt
③ echo '<p>$jahr wurde die $uni in $stadt gegründet.</p>';
// Ausgabe von Berechnungen mit Variablen sowie Zeichenketten
④ echo "<p>Die Gründung der $uni in $stadt erfolgte vor " . ($heute -
$jahr) . " Jahren.</p>";
?>
```

- ① Den Variablen \$stadt, \$uni, \$jahr und \$heute werden Zeichenketten- bzw. Ganzzahlenwerte zugewiesen.
- ② Zeichenketten in **doppelten Anführungszeichen** werden von PHP geparsst (ausgewertet). Die Variablen werden erkannt, ausgelesen und angezeigt.
- ③ Zeichenketten in **Hochkommata (einfache Anführungszeichen)** werden von PHP **nicht** geparsst. Hier werden die Werte der Variablen nicht ausgewertet, sondern die Variablennamen ausgegeben.
- ④ Wenn Sie eine Ausgabe mit dem Ergebnis einer Berechnung verknüpfen möchten, müssen Sie die Berechnung außerhalb der Zeichenketten vornehmen (und vor PHP 8 diese in Klammern setzen). Innerhalb der Anführungszeichen werden Operatoren nur als einfache Zeichen einer Zeichenkette verstanden.



Ausgabe der Beispieldatei „var_ausgabe.php“

Das \$-Zeichen, das als Erkennungszeichen der Variablen dient, wird von PHP innerhalb von Zeichenketten in **doppelten Anführungszeichen** erkannt und interpretiert. Soll das \$-Zeichen als Teil der Zeichenkette angezeigt werden, muss dem \$-Zeichen ein Backslash \ vorangestellt werden. Dadurch wird das \$-Zeichen vor PHP **versteckt**. PHP interpretiert dann das \$-Zeichen nicht, es wird deshalb als \$ im Browser angezeigt. Das Voranstellen eines Backslashes \ wird als „escapen“ bezeichnet.

Beispiel: zeichenkette.php

Nachfolgend werden Variablen auf verschiedene Arten mit Werten gefüllt und über den Befehl echo im Browser ausgegeben.

```
<?php
/*
Peter hat zu Hause noch amerikanische Dollar (USD) gefunden.
Welchen Wert (in Euro) hat sein Fund?
*/
① $dollar = 1240.45;
$kurs = 1.25; // Umrechnungskurs Dollar-Euro
$euro = $dollar / $kurs;
② $bezeichnung_dollar = "US Dollar (USD)";
$bezeichnung_euro = "EURO";
③ $ausgabe = "<p>Peter sagt: 'Meine " . $dollar . " " .
$bezeichnung_dollar;
④ $ausgabe .= " sind " . $euro . " " . $bezeichnung_euro . "
wert.'</p>";
⑤ echo $ausgabe;
⑥ echo "<p>Peter sagt: 'Meine $dollar $bezeichnung_dollar sind
$euro $bezeichnung_euro wert.'</p>";
⑦ echo '<p>Peter sagt: \'Meine $dollar $bezeichnung_dollar
sind $euro $bezeichnung_euro wert.\'</p>';
?>
```

- ① Den Variablen \$dollar und \$kurs werden Werte zugewiesen und der Wert der Variablen \$euro wird berechnet.
- ② Den Variablen \$bezeichnung_dollar und \$bezeichnung_euro werden Zeichenketten zugewiesen. Verwenden Sie sprechende Variablenbezeichnungen. Umso besser der Name der Variablen deren Aufgabe im PHP-Code beschreibt, umso einfacher ist ein PHP-Skript (für Sie und andere Entwickler) zu „lesen“. Siehe Zeilen ⑥ und ⑦.
- ③ Mithilfe des Operators `.=` wird der Variablen \$ausgabe eine Zeichenkette zugewiesen. Hierfür werden einzelne Zeichenketten, Variablen und HTML-Code miteinander verbunden. Bei der Verkettung der Zeichenketten müssen Sie selbst für die Leerzeichen zwischen den Wörtern sorgen.
- ④ Mithilfe des Operators `.=` wird die Zeichenkette \$ausgabe verlängert. Diese Vorgehensweise empfiehlt sich, um den Code übersichtlicher und gut lesbar zu gestalten.
- ⑤ Mit dem Befehl echo wird der Wert der Variablen \$ausgabe im Browser ausgegeben.
- ⑥ Sie können die gleiche Ausgabe auch direkt über eine Zeichenkette erreichen. Die Variablenwerte der angegebenen Variablen werden ausgegeben, wenn Sie die Zeichenkette durch doppelte Anführungszeichen begrenzen.
- ⑦ Steht hingegen eine Variable innerhalb einer durch Hochkommata begrenzten Zeichenkette, wird der Name der Variablen ausgegeben und nicht ihr Wert. Beachten Sie auch die Notwendigkeit, den Hochkommata innerhalb der Zeichenkette ein Backslash `\` voranzustellen. Fehlen diese Zeichen, erhalten Sie eine Fehlermeldung, da das zweite bzw. nachfolgende Hochkomma die Zeichenkette beenden würde.

Peter sagt: 'Meine 1240.45 US Dollar (USD) sind 992.36 EURO wert.'

Peter sagt: 'Meine 1240.45 US Dollar (USD) sind 992.36 EURO wert.'

Peter sagt: 'Meine \$dollar \$bezeichnung_dollar sind \$euro \$bezeichnung_euro wert.'

Ausgabe der Beispieldatei „zeichenkette.php“

Variablen in PHP

Variablen in PHP zu verwenden ist, gemessen an anderen Programmiersprachen wie z. B. JAVA, vergleichsweise einfach. Eine Variablen-deklaration zur Festlegung des Datentyps ist nicht notwendig. Es reicht, einer Variablen einen Wert zuzuweisen, PHP weist automatisch anhand des Wertes den richtigen Datentyp zu. Damit ist PHP wesentlich fehlertoleranter als andere Programmiersprachen.

Dieser einfache Umgang mit Variablen bringt jedoch einige Besonderheiten mit sich:

- ✓ Variablen können in Zeichenketten verwendet werden, z. B. zur Ausgabe durch den echo-Befehl. Ist die Zeichenkette durch doppelte Anführungszeichen begrenzt, werden die Werte der Variablen ausgelesen und zurückgeliefert. In anderen Programmiersprachen müssen häufig Variablen und Zeichenketten getrennt voneinander notiert und gegebenenfalls miteinander verkettet werden.
- ✓ Eine Variable in PHP kann während des Programmablaufs den Datentyp ändern. Dies kann entweder von Ihnen explizit so programmiert sein oder der Datentyp wird automatisch durch eine Berechnung geändert. Dieses Verhalten ist ausdrücklich erwünscht. Eine Fehlermeldung – wie sie viele Programmierer anderer Programmiersprachen erwarten – wird daher nicht ausgegeben.
- ✓ PHP erlaubt es mitunter, Rechenoperationen mit Zeichenketten durchzuführen. Das hat den Vorteil, dass ein Wert, der z. B. aus einer Datenbank als *string* "10" geliefert wird, also keine Variable vom Typ *int* oder *float* ist, trotzdem für eine Berechnung verwendet werden kann.
- ✓ PHP versucht, aus jeder Zeichenkette einen Wert zu ermitteln:
 - ✓ Beginnt eine Zeichenkette mit einem numerischen Wert, "10 Ziegen" oder "1.75 cm", wird der entsprechende *int* bzw. *float*-Wert ermittelt (also 10 bzw. 1.75), mit dem dann gerechnet werden kann.
 - ✓ Ist das erste Zeichen einer Zeichenkette ein Buchstabe oder ein Sonderzeichen, zeigt PHP seit der Version 8 einen *Fatal error* mit einer entsprechenden Fehlermeldung an. Falls kein Error-Management programmiert wurde, steigt PHP an der Stelle aus dem Skript aus. Vor PHP 8 wurde eine Zeichenkette, die nicht mit einer Nummer beginnt, bei einer Berechnung mit dem Wert 0 verwendet.



Mit PHP 8 wurde die Klassifizierung der Fehler-Level überarbeitet. Wenn Sie zuvor mit einer nicht deklarierten Variablen gerechnet haben, erzeugte PHP eine Meldung vom Typ *notice*. Bei den meisten Webservern werden *notice*-Meldungen nicht angezeigt. PHP 8 zeigt nun eine Meldung vom Typ *warning* an.

Vorteil: PHP „zwingt“ Entwickler nun, Variablen zu deklarieren, damit keine Fehlermeldungen angezeigt werden. Damit verbessert sich die Qualität von PHP-Skripten zwangsläufig.

Nachteil: Wenn Sie alte Skripte auf einem Server mit PHP 8 laufen lassen, werden Ihnen mit hoher Wahrscheinlichkeit extrem viele Fehler angezeigt werden. Wenn Webhoster von PHP 7 auf PHP 8 umgestellt werden, wird dies wahrscheinlich mit viel Nacharbeit verbunden sein.

Beispiel: var_verhalten.php

Im nachfolgenden Beispiel sehen Sie die beschriebenen Besonderheiten bei der Verwendung von Variablen in PHP. Dabei nimmt die Variable \$test im Verlauf des Skripts unterschiedliche Datentypen an:

```
<?php
① $test = "10";           // String
echo '$test: ' . $test . " (" . gettype($test) . ")<br>";

② $test = $test * 2;      // Integer (20)
echo '$test: ' . $test . " (" . gettype($test) . ")<br>";

③ $test = $test + 1.75;   // Fließkommazahl (21.75)
echo '$test: ' . $test . " (" . gettype($test) . ")<br>";

④ $test = 5 + "10 Tassen Tee";
// Integer (15) // Warning wird ausgegeben, da int und string
addiert werden soll.
echo '$test: ' . $test . " (" . gettype($test) . ")<br>";

⑤ $test = $test * "Kaffeetassen: 530";
// Fatal error - da "Kaffeetassen: 530" ein string ist,
// lässt PHP die Addition nicht zu.
⑥ echo '$test: ' . $test . " (" . gettype($test) . ")<br>";
?>
```

- ① Der Variablen \$test wird der Wert 10 zugewiesen, allerdings in Anführungszeichen. Damit handelt es sich um eine Zeichenkette, die Variable hat also den Datentyp *string*. Über die Funktion `gettype()` wird der Datentyp ausgegeben, um zu prüfen, welchen Datentyp die Variable hat.
- ② In diesem Beispiel wird die Variable \$test in eine Variable vom Datentyp *int* mit dem Wert 10 umgewandelt, welche dann für die weitere Berechnung verwendet wird.
- ③ Durch Addition einer Fließkommazahl ändert die Variable nochmals ihren Datentyp und wird zu einer Variablen des Datentyps *float* bzw. *double*.
- ④ Hier wird zu der Zahl 5 die Zeichenkette 10 Tassen Tee addiert. PHP wandelt die Zeichenkette 10 Tassen Tee automatisch in 10 um, um einen verwendbaren Wert für die Rechenoperation zu haben. Allerdings wird seit PHP 8 eine Fehlermeldung vom Typ *warning* angezeigt, wenn mit einer Zeichenkette gerechnet wird. Die Summe aus Zahl und Zeichenkette ergibt 15.

- ⑤ Die Variable \$test hat aus der vorherigen Rechenoperation den Wert 15. Durch die beabsichtigte Addition versucht PHP auch hier, den Zahlwert der Zeichenkette Kaffeetassen: 530 zu ermitteln. Da diese Zeichenkette nicht mit einem numerischen Wert beginnt, kann kein Zahlwert ermittelt werden, PHP reagiert mit einem Fatal Error. Bis zur Version PHP 7.x ermittelte PHP den Wert 0. Das Ergebnis der Addition war früher $15 + 0 = 15$, es wurde auch keine Fehlermeldung angezeigt.

The screenshot shows two side-by-side browser windows. The left window is titled 'PHP 7' and displays the output of the following code:

```
$test: 10 (string)
$test: 20 (integer)
$test: 21.75 (double)
$test: 15 (integer)
$test: 15 (integer)
```

The right window is titled 'PHP 8' and displays the output of the same code. It includes additional information about warnings and fatal errors:

```
$test: 10 (string)
$test: 20 (integer)
$test: 21.75 (double)

Warning: A non-numeric value encountered in C:\xampp\htdocs\Herd़t\Beispieldateien\Kapitel_04\var_verhalten.php on line 19
$test: 15 (integer)

Fatal error: Uncaught TypeError: Unsupported operand types: int + string in C:\xampp\htdocs\Herd़t\Beispieldateien\Kapitel_04\var_verhalten.php:22 Stack trace: #0 {main} thrown in C:\xampp\htdocs\Herd़t\Beispieldateien\Kapitel_04\var_verhalten.php on line 22
```

Im Gegensatz zu anderen Programmiersprachen ist PHP sehr fehlertolerant. Variablen können zu Laufzeit verschiedene Datentypen annehmen. Allerdings ist PHP 8 restriktiver als frühere PHP-Versionen, eine Rechenoperation mit einer Zeichenkette kann je nach Zeichenkette zum Fatal error führen.

Sie können jederzeit abfragen, welchen Datentyp eine bestimmte Variable aufweist und entsprechend darauf reagieren. Zur Abfrage des Datentyps können Sie die Funktion `gettype(<Variablename>)` verwenden, z. B.:

```
$testiable = 39;
echo gettype($testiable); // gibt in diesem Fall "integer" aus
```

3.4 Konstanten

Variablen können variable Werte haben. Über Operatoren können Variablen im laufenden PHP-Skript verändert werden und neue Werte annehmen. Es gibt allerdings auch Werte, deren Wertänderung durch das Skript nicht gewollt ist oder deren Änderung unsinnig wäre, wie z. B. die feststehende Länge einer Marathonstrecke oder der Kreiskonstanten Pi. Für diesen Zweck sieht PHP **Konstanten** vor.

Konstanten sind Variablen ähnlich. Der Unterschied besteht darin, dass Konstanten einmalig bei ihrer Definition ein Wert zugewiesen wird, der danach nicht mehr verändert werden kann.

Folgende Merkmale unterscheiden Konstanten von Variablen:

- ✓ Konstanten haben kein vorangestelltes \$-Zeichen im Bezeichner.
- ✓ Konstanten werden über die Funktion `define()` definiert, nicht durch eine einfache Zuweisung wie bei Variablen.

- ✓ Alternativ können Sie Konstanten über das Schlüsselwort `const` definieren. Dabei wird der Wert mit dem `=`-Operator wie bei normalen Variablen zugewiesen.
- ✓ Konstanten können skalare Datenwerte, Arrays und skalare Ausdrücke enthalten. Skalarwerte sind Ganzzahlen, Fließkommazahlen, Zeichenketten oder boolesche Werte.
- ✓ Seit PHP 5.5 können auch Arrays über das Schlüsselwort `const` als Konstante gespeichert werden.
- ✓ Seit PHP 5.6 sind konstante skalare Ausdrücke (Constant Scalar Expressions) möglich. Es können feststehende Ausdrücke (Rechenoperationen, die immer gleich bleiben) als Konstante definiert werden.
- ✓ Ab PHP 7.0 können Arrays auch über die Funktion `define()` deklariert und initialisiert werden. Dabei können indizierte, assoziative, ein- und mehrdimensionale Arrays (vgl. Kapitel 5) definiert werden.

Konstanten definieren

```
define ("NAME", Wert);
```

Zur Definition von Konstanten verwenden Sie die Funktion `define()`. Sie müssen zwei Argumente angeben: die Bezeichnung der Konstanten und den Wert, den Sie der Konstanten zuweisen wollen. Der Name der Konstanten muss dabei in **einfachen** oder **doppelten Anführungszeichen** stehen.

Definieren Sie Konstanten wie oben über `define("PARAM", 5)`, verhält sich diese Case-sensitiv (PHP unterscheidet zwischen Groß- und Kleinbuchstaben), sprich: `PARAM` ist eine andere Konstante als `param`. Falls Sie `define()` mit einem dritten Parameter `true` aufrufen, also `define("PARAM", 5, true)`, deaktivieren Sie die Case-Sensitivität, in dem Fall wäre `PARAM` und `param` dasselbe.

```
const NAME = Wert;
```

Alternativ können Sie Konstanten über das Schlüsselwort `const` definieren (seit PHP 5.3). Im Gegensatz zu `define()` muss beim `const` der Name der Konstanten ohne Anführungszeichen angegeben werden. Die Zuweisung der Werte geschieht über den Operator `=`. Ein weiterer Unterschied zu `define()` ist, dass über `const` definierte Konstanten im sogenannten *Top-Level-Scope* definiert werden müssen, sie können weder in Funktionen, Schleifen, `if`-Abfragen oder `try-catch`-Blöcken definiert werden.



Ist eine Konstante einmal definiert, kann sie zur Laufzeit des PHP-Skripts weder gelöscht noch neu definiert werden.

Konventionen für Konstanten-Bezeichnung

Schreiben Sie die Konstanten komplett in **Großbuchstaben**, dies entspricht gängigen Konventionen. Unabhängig davon, ob Sie für normale Variablen die Schreibweise mit Unterstrichen oder den CamelCase verwenden: Konstanten in Großbuchstaben sind im PHP-Code deutlich zu erkennen, Konstanten und Variablen sind so einfacher zu unterscheiden.

Beispiel: konstante.php

Im nachfolgenden Beispiel wird die Definition und Verwendung von Konstanten gezeigt.

```
<?php
① define("SEK_TAG", 86400); // Anzahl der Sekunden pro Tag
② define("MIN_TAG", 24 * 60); // Anzahl der Minuten pro Tag
③ define("GRUSS", "<hr><p>Ich wünsche Ihnen noch einen schönen
   Tag.<br>Herzliche Grüße...</p>");

④ const NETTO = 100;
const MWST = 0.19;
const BRUTTO = NETTO + NETTO * MWST;

⑤ define("WAEHRUNGEN", array('EURO', 'USD', 'GBP'));

⑥ echo "<p>Ein Tag besteht aus " . MIN_TAG . " Minuten oder " .
   SEK_TAG . " Sekunden.</p>";
⑦ echo "<p>Eine Woche besteht aus " . (7 * SEK_TAG) .
   " Sekunden</p>";
⑧ echo "<p>Eine Woche besteht aus 7 * SEK_TAG Sekunden</p>";
⑨ echo GRUSS;
echo "GRUSS";
⑩ echo "<p>Die Mehrwertsteuer beträgt " . MWST . "%.</p>";
echo "<p>Die Bruttopreis berechnet sich zu " . BRUTTO . "%
   aus dem Nettopreis.</p>";
⑪ echo "<hr><pre>";
print_r(WAEHRUNGEN);
echo "</pre>";
?>
```

- ① Einer Konstanten namens `SEK_TAG` wird der Zahlenwert 86400 zugewiesen.
- ② Der Konstanten `MIN_TAG` wird ebenfalls ein Zahlenwert zugewiesen, in diesem Fall das Ergebnis einer Rechenoperation.
- ③ Einer Konstanten namens `GRUSS` wird eine Zeichenkette zugewiesen. Hierbei kann es sich durchaus um komplexe HTML-Bausteine handeln.
- ④ Über `const` werden die Konstanten `NETTO`, `MWST` und `BRUTTO` definiert. Für die `BRUTTO`-Konstante wird ein konstant skalarer Ausdruck verwendet.
- ⑤ Über `define()` wird die Array-Konstante `WAEHRUNGEN` mit Währungsbezeichnungen definiert.

```
Konstanten definieren
localhost/herdt/Beispieldatei ... ▾

Ein Tag besteht aus 1440 Minuten oder 86400 Sekunden.
Eine Woche besteht aus 604800 Sekunden
Eine Woche besteht aus 7 * SEK_TAG Sekunden

Ich wünsche Ihnen noch einen schönen Tag.
Herzliche Grüße...

GRUSS

Die Mehrwertsteuer beträgt 0.19%.
Die Bruttopreis berechnet sich zu 119% aus dem Nettopreis.

Array
(
    [0] => EURO
    [1] => USD
    [2] => GBP
)
```

Ausgabe der Beispieldatei „konstante.php“

- ⑥ In dieser Zeile werden mehrere Zeichenketten und die definierten Konstanten MIN_TAG und SEK_TAG ausgegeben. Die Konstanten werden mit den Zeichenketten konkateniert.
- ⑦ Hier wird eine Rechenoperation mit der Konstanten SEK_TAG durchgeführt und das Ergebnis ausgegeben. Die Klammern um die Berechnung sind zwar nicht notwendig, unterstreichen jedoch, dass die Berechnung vor der Konkatenation durchgeführt wird.
- ⑧ Diese Zeile ist ähnlich der Zeile ⑦, nur dass hier die Konstante innerhalb der Zeichenkette (ohne Konkatenation) angegeben wurde. Im Resultat erhalten Sie nur die Ausgabe des Konstantennamens. Obwohl PHP Zeichenketten in doppelten Anführungszeichen parst, werden Konstanten in Zeichenketten im Gegensatz zu normalen Variablen nicht ausgewertet. PHP kann Zeichen und Konstanten nicht unterscheiden, zur Ausgabe einer Konstanten ist deswegen immer eine Konkatenation notwendig. Das gleiche gilt für Rechenoperationen, die in Zeichenketten notiert sind. Auch hier kann PHP nicht erkennen, ob es sich um das Zeichen `*` handelt oder den Operator `*`.
- ⑨ Es wird eine Konstante verwendet, die aus einem HTML-Block besteht. Es erfolgt die Ausgabe der definierten Zeichenkette. In der Folgezeile steht die Konstante wieder in einer Zeichenkette. Es erfolgt lediglich die Ausgabe des Konstantennamens, also die Zeichenfolge "GRUSS".
- ⑩ In dieser Zeile werden die Konstanten ausgegeben, die per `const` definiert wurden. Dort können Sie auch erkennen, dass die Konstante BRUTTO den berechneten Wert angenommen hat.
- ⑪ Die Array-Konstante WAEHRUNGEN aus Zeile ⑤ wird ausgegeben.

3.5 Übungen

Übung 1: Werte von Variablen erkennen

Level		Zeit	ca. 5 min
Übungsinhalte	<ul style="list-style-type: none"> ✓ Variablen ✓ Arithmetische Operatoren ✓ Zeichenkettenoperator 		
Übungsdatei	--		
Ergebnisdatei	antworten-3.php		

1. Welche Ausgabe erhalten Sie bei den nachfolgenden Codezeilen?

Die Variablen haben folgende Werte:

```
$a = 7;
$b = "30 Euro";
$c = "! ";
a) echo $a . $b . $c;
b) echo "Text";
c) echo "Text" . $a;
d) echo "Text" $a . $b;
e) echo $a + $b;
```

- f) echo \$a + \$b + \$c;
 g) echo \$a * \$b / \$c;
 h) echo ('\'Text\'' . \$a . " Text " . \$b);

Übung 2: Mit Variablen, Operatoren und Konstanten arbeiten

Level		Zeit	ca. 10 min
Übungsinhalte	<ul style="list-style-type: none"> ✓ Variablen ✓ Konstanten ✓ Arithmetische Operatoren ✓ Zeichenkettenoperator ✓ Bildschirmausgabe 		
Übungsdatei	--		
Ergebnisdatei	<i>büero.php</i>		

1. Erstellen Sie mit folgenden Angaben ein PHP-Skript, das Sie unter dem Namen *büero.php* speichern.

Variable	Bezeichnung	Variable	Preis (netto)
\$bezeichnung_tisch	Schreibtisch	\$preis_tisch	1999.00 €
\$bezeichnung_stuhl	Bürostuhl	\$preis_stuhl	589.00 €
\$bezeichnung_lampe	Lampe	\$preis_lampe	29.00 €
\$bezeichnung_pctisch	Computertisch	\$preis_pctisch	999.00 €

Berechnen Sie den Gesamtpreis (\$netto_gesamt) der eingekauften Artikel.

2. Berechnen Sie für den gerade berechneten Gesamtpreis den Bruttopreis (\$brutto_gesamt) mithilfe einer Konstanten namens MWST. Der Mehrwertsteuersatz, der zur Berechnung verwendet wird, beträgt 19 %. Die verwendete Zeichenkette für die Währung Euro stellen Sie bitte ebenfalls über eine Konstante (EURO) bereit.
3. Berechnen Sie zusätzlich die Bruttopreise aller Artikel.
4. Lassen Sie alle errechneten Werte in verständlicher Form mit Beschriftungen anzeigen.

Mit Variablen, Operatoren und Konstanten arbeiten

Netto-Gesamtpreis der eingekauften Artikel: 3616 Euro.

Brutto-Gesamtpreis der eingekauften Artikel: 4303.04 Euro.

Brutto-Preis Schreibtisch: 2378.81 Euro.

Brutto-Preis Bürostuhl: 700.91 Euro.

Brutto-Preis Schreibtischlampe: 34.51 Euro.

Brutto-Preis Computertisch: 1188.81 Euro.

Lösungsvorschlag „büero.php“

4

Kontrollstrukturen

 **Beispieldateien:** Dateien im Ordner *Kapitel_04*

4.1 Kontrollstrukturen einsetzen

Zeilenweise Ausführung

Die Anweisungen in einem PHP-Skript werden in der Reihenfolge ausgeführt, in der sie angegeben sind, falls keine Anweisung einen Befehl enthält, der diese Reihenfolge ändert. Ein PHP-Skript wird in diesem Fall sequenziell, also Zeile für Zeile, abgearbeitet.

Warum sind Kontrollstrukturen notwendig?

Oft ist es erforderlich, dass eine oder mehrere Anweisungen ...

- ✓ nur unter bestimmten Bedingungen durchgeführt werden sollen, z. B. ein Rabatt für einen Einkauf wird nur gewährt, falls eine Mindestbestellmenge vorliegt;
- ✓ wiederholt durchgeführt werden sollen, z. B. beim Lesen aller oder gefilterter Einträge aus einer Liste.

Um von der sequenziellen Abarbeitung der Befehle abzuweichen, setzen Sie Kontrollstrukturen ein. Sie können gezielt den Ablauf des Skripts durch Bedingungen oder Schleifen steuern.

Anweisungen über Bedingungen auswählen

Eine Bedingung ist eine Möglichkeit, den Ablauf eines Skripts durch Entscheidungen zu beeinflussen. In einer Bedingung werden Ausdrücke verglichen. Das Ergebnis dieses Vergleichs kann dabei entweder mit "Ja" (TRUE) oder mit "Nein" (FALSE) beantwortet werden. Hierbei können Sie mit folgenden Operatoren arbeiten:

Vergleichsoperatoren

Diese Operatoren, auch relationale Operatoren genannt, vergleichen Ausdrücke miteinander und liefern ein logisches Ergebnis, entweder TRUE (wahr) oder FALSE (falsch).

Operator	Name	Bedeutung	Beispiel	Ergebnis
<code>==</code>	Gleich	<code>\$a == \$b</code> ergibt TRUE, wenn \$a und \$b gleich sind.	<code>\$a = 4; \$b = 4.0; \$a == \$b;</code>	TRUE
<code>!= oder <></code>	Ungleich	<code>\$a != \$b</code> ergibt TRUE, wenn \$a und \$b ungleich sind.	<code>\$a = 4; \$b = 4; \$a != \$b;</code>	FALSE
<code>====</code>	Identisch	<code>\$a === \$b</code> ergibt TRUE, wenn \$a und \$b gleich und vom selben Datentyp sind.	<code>\$a = 4; \$b = 4.0; \$a === \$b;</code>	FALSE
<code>!==</code>	Nicht identisch	<code>\$a !== \$b</code> ergibt TRUE, wenn \$a und \$b ungleich oder nicht vom selben Datentyp sind.	<code>\$a = 4; \$b = 4.0; \$a !== \$b;</code>	TRUE
<code><</code>	Kleiner	<code>\$a < \$b</code> ergibt TRUE, wenn \$a kleiner \$b ist.	<code>\$a = 3; \$b = 4; \$a < \$b;</code>	TRUE
<code>></code>	Größer	<code>\$a > \$b</code> ergibt TRUE, wenn \$a größer \$b ist.	<code>\$a = 3; \$b = 4; \$a > \$b;</code>	FALSE
<code><=</code>	Kleiner gleich	<code>\$a <= \$b</code> ergibt TRUE, wenn \$a kleiner oder gleich \$b ist.	<code>\$a = 4; \$b = 4; \$a <= \$b;</code>	TRUE
<code>>=</code>	Größer gleich	<code>\$a >= \$b</code> ergibt TRUE, wenn \$a größer oder gleich \$b ist.	<code>\$a = 5; \$b = 4; \$a >= \$b;</code>	TRUE



Im Unterschied zum **Gleichheitsoperator** (zwei Gleichheitszeichen) wird beim **Identisch-Operator** (drei Gleichheitszeichen) zusätzlich der Datentyp verglichen. Wird beispielsweise die Zahl 1 mit der Zeichenkette "1" per `= =` verglichen, ist die Überprüfung wahr. Werden die beiden Werte hingegen per `= = =` verglichen, ist die Abfrage falsch, da die Datentypen nicht übereinstimmen.

Vor allem bei 0 und 1, sowohl als Zahl als auch als Zeichenkette, und bei true und false liefern Gleichheitsoperator und Identisch-Operator unterschiedliche Ergebnisse, wie die folgende Tabelle zeigt:

Vergleich	Ergebnis
<code>1 == "1"</code>	TRUE
<code>TRUE == 1</code>	TRUE
<code>FALSE == 0</code>	TRUE
<code>1 === "1"</code>	FALSE

Vergleich	Ergebnis
TRUE === 1	FALSE
FALSE === 0	FALSE
"1" === "1"	TRUE
1 === 1	TRUE
TRUE === TRUE	TRUE
FALSE === FALSE	TRUE

Spaceship- und Null coalescing-Operatoren

Der Spaceship-Operator und der Null coalescing-Operator sind neu seit PHP 7.0. Diese Operatoren sind eine Mischform aus einem Vergleichsoperator und einem Zuweisungsoperator, da sie beides tun.

Operator	Name	Bedeutung	Beispiel	Ergebnis
<=>	Spaceship	\$a <=> \$b gibt einen Integer zurück. Falls \$a < \$b ist, ist der Rückgabewert -1, falls \$a == \$b ist und 1, falls \$a > \$b ist.	\$a = 6; \$b = 9; \$a <=> \$b;	-1
??	Null coalescing	\$c = \$a ?? \$b weist \$c den Wert von \$a zu, falls die Variable \$a initialisiert ist, ansonsten den Wert von \$b .	\$b = 4; \$c = \$a ?? \$b	\$c = 4

Spaceship-Operator

```
$wert = $a <=> $b
```

- ✓ Vergleicht zwei Werte miteinander.
- ✓ Liefert einen Wert vom Datentyp *int* zurück.
- ✓ Ist der linke Wert kleiner als der rechte, ist der Rückgabewert -1.
- ✓ Sind beide Werte gleich, ist der Rückgabewert 0.
- ✓ Ist der linke Wert größer als der rechte, ist der Rückgabewert 1.
- ✓ Der Rückgabewert kann in einer Variablen gespeichert (hier *\$wert*) und weiterverarbeitet werden.
- ✓ Es wird keine Datentyp-Prüfung durchgeführt. Beispiel: 6 <=> "6" wird als gleich erkannt und liefert den Wert 0 zurück.
- ✓ Das Aussehen des Operators erinnert an ein Raumschiff und trägt deswegen auch den Namen „Spaceship“.

Null coalescing-Operator

Dieser Operator erinnert an einen ternären Operator (vgl. Abschnitt 4.3) und funktioniert auch ähnlich.

```
$wert = $a ?? $b;
```

- ✓ Es wird geprüft, ob die Variable \$a existiert.
- ✓ Falls ja, wird der Wert von \$a der Variablen \$wert zugewiesen.
- ✓ Falls nicht, wird Wert von \$b ohne weitere Prüfung der Variablen \$wert zugewiesen, auch wenn \$b selbst nicht definiert ist.
- ✓ Das Verhalten entspricht der PHP-Funktion `isset()`, welche ebenfalls prüft, ob eine Variable deklariert und initialisiert ist.
- ✓ Es wird nur auf NULL-Werte geprüft. Die Zahl `$a=0`; oder eine leere Zeichenkette `$a=" "`; werden als gültig erkannt und dann der Variablen \$wert zugewiesen.

Der Null coalescing-Operator kann nützlich sein, um zu prüfen, ob eine Variable vorhanden ist, und falls nicht, die im PHP-Skript verwendete Variable mit einem sinnvollen Standardwert zu versehen. Der im Beispiel verwendete Wert für \$b sollte von daher immer deklariert und initialisiert sein.

4.2 Die einfache `if`-Anweisung

In PHP erreichen Sie auf verschiedene Arten eine Verzweigung des Programmablaufs. Die einfachste und häufig eingesetzte Variante ist die Bedingungsprüfung mit der `if`-Anweisung. Wenn die Bedingung erfüllt ist, **dann** wird der in Klammern angegebene Anweisungsblock ausgeführt. Ist die Bedingung nicht erfüllt, wird der Anweisungsblock übersprungen.

Syntax und Bedeutung der `if`-Anweisung

- ✓ Die `if`-Anweisung beginnt mit dem reservierten Wort `if`.
- ✓ Die Bedingung ① steht in runden Klammern.
- ✓ Der Anweisungsblock ② steht in geschweiften Klammern.
- ✓ Die geschweiften Klammern sind optional. Falls diese fehlen, ist der Abweisungsblock der PHP-Code bis zum nächsten Semikolon.
- ✓ Als Bedingung ist ein logischer Ausdruck anzugeben, der einen der beiden Zustände `TRUE` (wahr) oder `FALSE` (falsch) zurückliefert.
- ✓ Liefert die Bedingung `TRUE` zurück, werden die Anweisungen ② ausgeführt. Ist die Bedingung `FALSE`, werden die Anweisungen ② ignoriert.
- ✓ Nach dem Ende der `if`-Anweisung werden alle weiteren Anweisungen unabhängig von der Bedingung abgearbeitet.

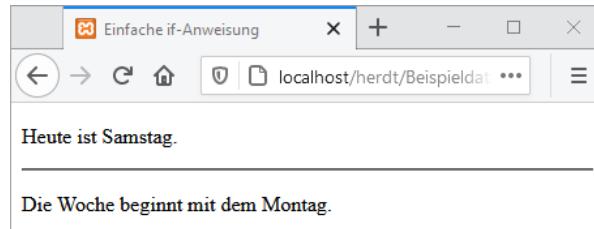
<code>if</code> (Bedingung) {	①
Anweisungsblock;	②
}	

Die auszuführenden Anweisungen ② werden auch Anweisungsblock genannt, auch wenn sie nur aus einer Anweisung bestehen. In einem Anweisungsblock, der von geschweiften Klammern ① ② steht, können mehrerer PHP-Anweisungen hinterlegt werden.

Beispiel: if-1.php

```
<?php
① $wochentag = "Samstag";
② if ($wochentag == "Samstag") {
③     echo "<p>Heute ist Samstag.</p>";
④ }
⑤ echo "<hr><p>Die Woche beginnt mit dem Montag.</p>";
?>
```

- ① Der Variablen \$wochentag wird der Wert Samstag zugewiesen.
- ② Es folgt die Prüfung der Bedingung, ob die Variable \$wochentag den Wert Samstag aufweist. Die Prüfung ergibt TRUE, sodass der folgende Anweisungsblock ausgeführt wird.
- ③ Der Anweisungsblock in geschweiften Klammern besteht aus einer Meldung, die auf dem Bildschirm ausgegeben wird.
- ④ Die geschweifte Klammer } beendet die if-Anweisung.
- ⑤ Die letzte Ausgabe wird unabhängig vom Ergebnis der Bedingungsprüfung auf dem Bildschirm angezeigt, da die Zeile hinter der schließenden geschweiften Klammer }, also hinter der if-Anweisung, liegt.



Anzeige der Beispieldatei „if-1.php“

Häufiger Fehler bei Bedingungsprüfungen

Ein häufiger Fehler ist, dass statt eines Vergleichsoperators mit zwei Gleichheitszeichen == nur ein einzelnes Gleichheitszeichen = geschrieben wird (oft als Flüchtigkeitsfehler). Ein einzelnes Gleichheitszeichen = ist jedoch ein Zuweisungsoperator und weist einer Variablen links vom = den Wert zu, der rechts neben dem = steht. Die if-Anweisung überprüft dann nicht den Vergleich, sondern den zugewiesenen Wert. Ist der zugewiesene Wert nicht NULL, FALSE, 0, '0' oder '' (leere Zeichenkette), ist die Überprüfung wahr, der if-Zweig wird dann ausgeführt.

Ein nützlicher Trick ist es, Variable und Prüfwert zu vertauschen:

"Samstag" == \$wochentag.

Für den Vergleich selbst spielt die Reihenfolge der einzelnen Argumente keine Rolle. Würde hier versehentlich ein Gleichheitszeichen vergessen, würde von PHP eine Fehlermeldung ausgegeben (*parse error*).

if-Anweisung ohne geschweifte Klammern

Beispiel: if-1a.php

Die if-Anweisung (wie auch andere Kontrollstrukturen) benötigt nicht zwingend geschweifte Klammern {}, die den Anweisungsblock umschließen. Werden diese weggelassen, erkennt PHP die Anweisung bis zum nächsten Semikolon als Anweisungsblock.

```
<?php
$wochentag = "Samstag";
if ($wochentag == "Samstag")
    echo "<p>Heute ist Samstag.</p>";
② echo "<hr><p>Die Woche beginnt mit dem Montag.</p>";
?>
```

Im Gegensatz zum Beispiel „if-1.php“ fehlen hier die geschweiften Klammern. Die Ausgabe ist jedoch identisch.

- ① Der Anweisungsblock der if-Anweisung hat hier keine geschweiften Klammern {}. Die vollständige Anweisung wird durch das Semikolon am Ende der Zeile ① beendet.
- ② Diese Zeile befindet sich nicht mehr in der if-Anweisung.

! Kontrollstrukturen ohne geschweifte Klammern {} sind schwieriger zu lesen, gerade wenn keine Einrückungen vorgenommen werden. Um Fehler zu vermeiden, sollten geschweifte Klammern {} verwendet werden.

Beispiel: if-2.php

Sie können auch mehrere if-Anweisungen nacheinander verwenden. So können Sie flexibel mehrere Werte oder Variablen prüfen und darauf individuell reagieren.

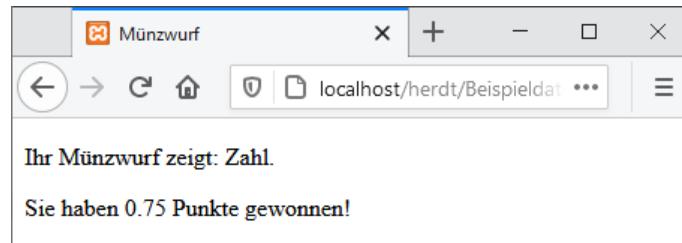
```
<?php
① $muenzwurf = "Zahl"; // alternativ: "Kopf" oder "Rand"
② if ($muenzwurf == "Kopf") {
    echo "<p>Ihr Münzwurf zeigt: Kopf.</p>";
    $gewinn = 0.5;
}
③ if ($muenzwurf == "Zahl") {
    echo "<p>Ihr Münzwurf zeigt: Zahl.</p>";
    $gewinn = 0.75;
}
④ if ($muenzwurf == "Rand") {
    echo "<p>Ihr Münzwurf zeigt: Rand.</p>";
    $gewinn = 2.5;
}
```

```

⑤ if ($gewinn > 2) {
    echo "<p><strong>Super! Das ist selten...</strong></p>";
}
⑥ echo "<p>Sie haben $gewinn Punkte gewonnen!</p>";
?>

```

- ① Der Variablen \$muenzwurf wird der Wert Zahl zugewiesen. Experimentieren Sie mit dem Skript: Tragen Sie als Wert zum Testen auch einmal Kopf bzw. Rand ein.
- ② Es wird geprüft, ob der Variablenwert gleich Kopf ist. Trifft die Bedingung zu, wird eine Ausgabe sowie eine Variablenzuweisung (\$gewinn) vorgenommen.
- ③ Nach einer weiteren Prüfung, ob der Variablenwert gleich Zahl ist, erfolgt – falls die Überprüfung erfolgreich ist – eine Ausgabe sowie eine weitere Variablenzuweisung (\$gewinn).
- ④ Es folgt eine dritte Prüfung, ob der Variablenwert gleich Rand ist. Bei erfolgreicher Prüfung wird der angegebene echo-Befehl ausgeführt. Es erfolgt zusätzlich eine weitere Variablenzuweisung (\$gewinn).
- ⑤ Abschließend wird geprüft, ob die Variable \$gewinn größer als 2 ist. In dem Fall erfolgt eine weitere Ausgabe.
- ⑥ Die letzte Ausgabe wird in jedem Fall ausgeführt. Sie gehört nicht mehr zu den vorherigen if-Anweisungen, da sie hinter der schließenden geschweiften Klammer } steht.



Ausgabe der Beispieldatei „if-2.php“

4.3 Die if-Anweisung mit else-Zweig

Soll nicht nur ein Anweisungsblock durchgeführt werden, falls die Bedingung erfüllt ist, sondern ein anderer, sofern die Bedingung nicht erfüllt ist, verwenden Sie die if-else-Anweisung.

Syntax und Bedeutung der if-else-Anweisung

- ✓ Die Alternative leiten Sie mit dem Schlüsselwort else ein.
- ✓ Danach folgen die alternativen Anweisungen.
- ✓ Falls die Bedingung erfüllt ist, wird Anweisungsblock 1 ausgeführt, sonst Anweisungsblock 2 (wenn-dann-sonst).
- ✓ Auch hier sind die geschweiften Klammern {} optional.

```

if (Bedingung) {
    Anweisungsblock 1;
} else {
    Anweisungsblock 2;
}

```

Beispiel: *ifelse.php*

Wenn die Variable \$menge größer als 5 ist, soll eine Meldung ausgegeben werden. Falls \$menge nicht größer 5, kleiner oder gleich 5 ist, soll eine alternative Meldung am Bildschirm angezeigt werden.

```
<?php
①   $menge = 4;
    echo "<p>Sie haben $menge Kilo bestellt.</p>";
②   if ($menge > 5) {
        echo "<p>Glückwunsch, der Versand ist ab 5 Kilo
            kostenfrei.</p>";
③   } else {
        echo "<p>Bei kleinen Mengen kostet der Versand leider drei
            Euro.</p>";
    }
?>
```

- ① Der Variablen \$menge wird der Wert 4 zugewiesen.
- ② Es erfolgt die Bedingungsprüfung, ob \$menge größer als 5 ist. Falls die Prüfung TRUE ergibt, wird der folgende Anweisungsblock ausgeführt.
- ③ Trifft die Bedingung nicht zu, wird der mit else eingeleitete alternative Anweisungsblock ausgeführt.

Kurzschrifweise: Ternärer Operator

```
wenn Bedingung ? dann TRUE : sonst FALSE
```

PHP kennt eine Kurzschrifweise für die if-else-Anweisung. Abfragen lassen sich damit kürzer gestalten. Die Nachvollziehbarkeit und Verständlichkeit der Programmierung kann allerdings durch Einsatz des ternären Operators leiden. Ternäre Operatoren sind schwieriger zu lesen als if-else-Anweisungen.

Beispiel: *ternaeroperator.php*

```
<?php
$schalter = 1; // mögliche Werte: 1 / 0
echo "<p>Das Licht ist " . ($schalter == 1 ? "AN" : "AUS") .
"?</p>";
?>
```

Die runden Klammern um den ternären Operator sind notwendig, damit der ternäre Operator ausgeführt und das entsprechende Ergebnis in den Satz eingefügt wird. Ohne die Klammern führt PHP den ternären Operator nicht korrekt aus, da zwischen der Zeichenkette und dem ternären Operator durch den . -Operator eine Konkatenation durchgeführt wird.

Verknüpfung von Bedingungen

Eine `if`-Anweisung ist nicht auf eine einzelne Bedingungsprüfung beschränkt, Sie können auch mehrere Anweisungen kombinieren und die Regeln definieren, ob nur eine oder alle Bedingung gültig sein müssen, damit die ganze `if`-Anweisung gültig ist. Über folgende logische Operatoren legen Sie diese Regeln fest:

Operator	Name	Bedeutung	Beispiel	Ergebnis
AND <code>&&</code>	UND	\$a and \$b ergibt TRUE, wenn sowohl \$a als auch \$b ungleich 0 sind, ansonsten wird FALSE zurückgegeben. Sobald eine der Bedingungsprüfungen FALSE ergibt, werden die weiteren Bedingungen nicht mehr geprüft.	<code>\$a = TRUE; \$b = FALSE; \$a AND \$b;</code>	FALSE
OR <code> </code>	ODER	\$a or \$b ergibt TRUE, wenn mindestens eine der beiden Variablen ungleich 0 ist. Sobald eine der Bedingungsprüfungen TRUE ergibt, werden die weiteren Bedingungen nicht mehr geprüft.	<code>\$a = TRUE; \$b = FALSE; \$a OR \$b;</code>	TRUE
XOR	ENTWEDER ODER	\$a xor \$b ergibt TRUE, wenn entweder \$a oder \$b ungleich 0 sind, aber nicht, wenn beide ungleich 0 sind.	<code>\$a = TRUE; \$b = FALSE; \$a XOR \$b;</code>	TRUE
!	NICHT	! \$a ergibt die Umkehrung des Wahrheitswertes.	<code>\$a = FALSE; ! \$a;</code>	TRUE

Beispiel: `bedingungen_verknuepfen.php`

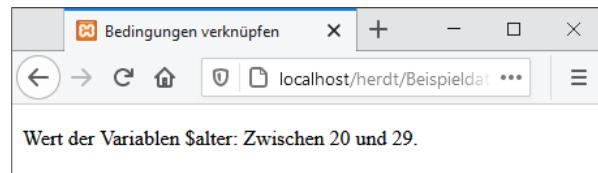
Durch Verknüpfung von Bedingungen können Sie z. B. prüfen, ob eine Altersangabe zwischen zwei Werten liegt und entsprechend darauf reagieren.

```
<?php
$alter = 26;
① if ($alter >= 20 && $alter < 30) {
    echo "<p>Wert der Variablen \$alter: Zwischen 20 und
29.</p>";
} else {
    echo "<p>Wert der Variablen \$alter: NICHT zwischen 20 und
29.</p>";
}
?>
```

- ① Der Wert der Variablen \$alter wird überprüft. Damit die Bedingung erfüllt ist und der Anweisungsblock im if-Zweig ausgeführt wird, müssen beide Bedingungen korrekt sein: \$alter >= 20 und \$alter < 30.

In diesem Beispiel wird das &&-Zeichen für die Und-Verknüpfung verwendet. Alternativ kann das Schlüsselwort AND verwendet werden: \$alter >= 20 AND \$alter < 30. Die Verknüpfung wäre dieselbe, es wäre lediglich eine andere Schreibweise derselben Verknüpfung.

- ② Wenn allerdings nur eine oder gar keine Bedingung zutrifft, wird der Anweisungsblock des else-Zweiges ausgeführt.



Ausgabe der Datei „bedingungen_verknuepfen.php“

4.4 Verschachtelte if-Anweisungen

Sie haben die Möglichkeit, Bedingungen und deren Anweisungsblöcke ineinander zu verschachteln. Ist die erste Bedingung wahr, wird kontrolliert, ob auch die nächste Bedingung zutrifft. Trifft eine Bedingung nicht zu, wird der alternative Zweig ausgeführt bzw. die verschachtelte Auswahl übersprungen.

```
if (Bedingung 1) {
    Anweisungsblock 1;
} else {
    if (Bedingung 2) {
        Anweisungsblock 2;
    } else {
        if (Bedingung 3) {
            Anweisungsblock 3;
        } else {
            Anweisungsblock 4;
        }
    }
}
```

Allgemeines Beispiel für eine verschachtelte if-Anweisung

- ✓ Für die einzelnen if-Anweisungen gelten die gleichen Regeln wie bei einer einfachen if-Anweisung.
- ✓ Bei einer verschachtelten if-Anweisung, beginnend mit einer öffnenden geschweiften Klammer {}, muss diese auch wieder durch eine schließende geschweifte Klammer {} abgeschlossen werden. Die Verschachtelung erkennen Sie in der Abbildung an den ein-gerückten Code-Zeilen.
- ✓ Der Anweisungsblock 1 wird abgearbeitet, wenn die Bedingung 1 zutrifft. Wenn sie nicht zutrifft, erfolgt die Prüfung der Bedingung 2. Trifft diese zu, erfolgt die Abarbeitung des Anweisungsblocks 2 usw. Trifft keine der Bedingungen zu, wird Anweisungsblock 4 abgearbeitet.

Rücken Sie verschachtelte if-Anweisungen sorgfältig ein, damit die Programmstruktur auch optisch erkennbar ist. Ohne entsprechende Einrückungen wird PHP-Code schnell unübersichtlich.

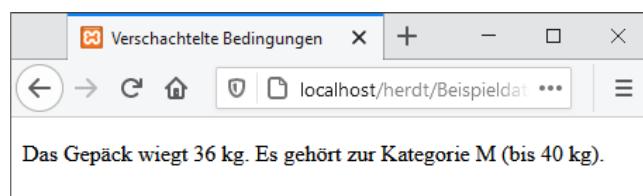
Beispiel: verschachtelung.php

Im folgenden Beispiel wird ein Gepäckstück anhand seines Gewichts in eine Gepäckkategorie einsortiert. Die Regeln lauten:

- ✓ Wieg das Gepäckstück bis einschließlich 20 kg, gehört es zur Kategorie S;
- ✓ wiegt das Gepäckstück mehr (bis einschließlich 40 kg), gehört es zu Kategorie M;
- ✓ wiegt es mehr als 40 und maximal 60 kg, ist es in Kategorie L einzurichten;
- ✓ ansonsten (über 60 kg) gehört es zur Kategorie XL.

```
<?php
① $gewicht = 36; // Beispiel-Gewicht eines Gepäckstücks in kg
echo "<p>Das Gepäck wiegt $gewicht kg. Es gehört zur
      Kategorie ";
② if ($gewicht <= 20) {
    ③ echo "S (bis 20 kg) ";
    ④ } else {
        ⑤ if ($gewicht <= 40) {
            ⑥ echo "M (bis 40 kg) ";
        ⑦ } else {
            ⑧ if ($gewicht <= 60) {
                ⑨ echo "L (bis 60 kg) ";
            ⑩ } else {
                ⑪ echo "XL (über 60 kg) ";
            ⑫ }
        ⑬ }
    ⑭ }
⑮ echo ".</p>";
?>
```

- ① Der Variablen \$gewicht wird der Wert 36 zugewiesen.
- ② Wenn \$gewicht kleiner oder gleich 20 ist, wird die nachfolgende echo-Anweisung ③ im if-Zweig ausgeführt. Danach ist die Bearbeitung der if-Anweisung beendet.
- ④ Ist \$gewicht größer als 20, wird im else-Zweig nach einer zutreffenden Bedingung gesucht. Da der else-Zweig aus weiteren Prüfungen besteht, finden Sie weitere Bedingungsprüfungen ⑤–⑧.



Anzeige der Beispieldatei „verschachtelung.php“
bei \$gewicht = 36

4.5 Erweiterte if-Anweisung mit elseif

Zusätzlich zu verschachtelten if-Anweisungen können Sie mit elseif weitere Bedingungen prüfen. Da Sie elseif beliebig oft einsetzen können, ist es möglich, eine beliebige Anzahl von zusätzlichen Bedingungen zu prüfen. Falls keine der vorherigen Bedingungen zutrifft, weder in der if-Anweisung noch in weiteren elseif-Anweisungen, wird der else-Zweig aufgerufen.

```
if (Bedingung 1) {
    Anweisungsblock 1;
} elseif (Bedingung 2) {
    Anweisungsblock 2;
} [elseif (Bedingung n) {
    Anweisungsblock n;
}] else {
    Anweisungsblock else;
}
```

Allgemeines Beispiel für eine if-Anweisung mit elseif

- ✓ Für die einzelnen if-Anweisungen und ebenso für die einzelnen elseif-Anweisungen gelten die gleichen Regeln wie bei einer einfachen if-Anweisung.
- ✓ Sie können in einer if-Anweisung beliebig viele elseif-Anweisungen verwenden.
- ✓ Der else-Zweig wird – wie bei der einfachen if-Anweisung – nur dann ausgeführt, wenn keine der vorherigen Bedingungen zutrifft.

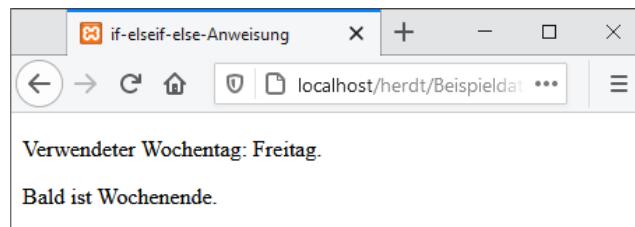
Beispiel: if-elseif.php

Im folgenden Beispiel wird ein Wochentag definiert. Durch eine if-Anweisung wird nach folgenden Regeln geprüft, ob es sich um einen Tag handelt, der dem Wochenende zuzurechnen ist oder nicht.

- ✓ Samstag oder Sonntag führen zur Ausgabe, dass Wochenende ist.
- ✓ Freitag führt zur Ausgabe, dass bald Wochenende ist.
- ✓ Alle anderen Werte erzwingen eine Ausgabe, dass kein Wochenende ist.

```
<?php
① $wochentag = "Freitag"; // testen mit "Mittwoch", "Freitag"
     und "Samstag"
    echo "<p>Verwendeter Wochentag: $wochentag.</p>";
② if ($wochentag == "Samstag" OR $wochentag == "Sonntag") {
    echo "<p>Es ist Wochenende.</p>";
③ } elseif ($wochentag == "Freitag") {
    echo "<p>Bald ist Wochenende.</p>";
④ } else {
    echo "<p>Es dauert noch bis zum Wochenende.</p>";
}
?>
```

- ① Der Variablen \$wochentag wird der Wert Freitag zugewiesen.
- ② Wenn \$wochentag den Wert Samstag oder Sonntag aufweist, erfolgt eine Ausgabe, dass Wochenende ist. Danach ist die Bearbeitung der kompletten if-elseif-else-Anweisung beendet.



Anzeige der Beispieldatei „if-elseif.php“ mit \$wochentag = Freitag

In der Bedingung wird das Schlüsselwort OR verwendet. Das doppelte Pipe-Zeichen $\sqcup \sqcup$ bewirkt dieselbe Verknüpfung (`$wochentag == "Samstag" || $wochentag == "Sonntag"`).

- ③ Mit der `elseif`-Anweisung wird ein weiterer möglicher Wert der Variablen `$wochentag` geprüft. Ist der Wert der Variablen Freitag, erfolgt die Ausgabe, dass bald Wochenende ist. Auch in diesem Fall ist die Bearbeitung der ganzen `if-elseif-else`-Anweisung beendet.
- ④ Der `else`-Zweig wird ausgeführt, wenn keine der vorhergehenden Bedingungen zutrifft. In diesem Fall wird ausgegeben, dass es noch dauert, bis Wochenende ist.

4.6 Fallauswahl mit der `switch`-Anweisung

Falls Sie per `if`-Anweisung viele Überprüfungen der gleichen Variablen durchführen, kann das je nach Anzahl der Überprüfungen schwer zu verstehen und übersichtlich sein. Als Alternative steht Ihnen in PHP die `switch`-Anweisung zur Verfügung.

Bei der `switch`-Anweisung bestimmt der Wert einer Variablen, welcher Anweisungsblock ausgeführt wird. Diese Vorgehensweise wird Fallauswahl oder Selektion genannt.

Syntax und Bedeutung der `switch`-Anweisung

```
switch ($variable) {
    case Wert 1:
        Anweisungsblock 1;
        break;
    case Wert 2:
        Anweisungsblock 2;
        break;
    default:
        Anweisungsblock 3;
}
```

- ✓ Die Fallauswahl wird mit dem reservierten Wort `switch` eingeleitet.
- ✓ Danach folgt die Variable, deren Wert in den folgenden Bedingungsfällen überprüft wird. Anschließend leitet die öffnende geschweifte Klammer `{` die Fallauswahl ein.
- ✓ Jede Auswahl wird durch das Schlüsselwort `case` (deutsch: *Fall*) eingeleitet, dem die Angabe eines Wertes oder einer Bedingung folgt. Zur Trennung vom folgenden Anweisungsblock wird am Ende der `case`-Angabe ein Doppelpunkt `:` gesetzt.
- ✓ Stimmt der Wert der Variablen mit einem der aufgeführten Auswahlwerte überein bzw. ergibt die Bedingungsprüfung `TRUE`, dann wird der Anweisungsblock danach bis zur Anweisung `break` ausgeführt. Die restlichen Auswahlblöcke werden nicht ausgeführt.
- ✓ Stimmt der Wert der Variablen mit keinem der angegebenen Werte überein, wird der Anweisungsblock nach der optionalen `default`-Anweisung durchgeführt. Wird der `default`-Auswahlblock weggelassen, führt das Programm in diesem Fall keine Anweisungen innerhalb der `switch`-Anweisung aus, sondern erst die nächste Anweisung nach dem Ende der Fallauswahl.
- ✓ Die Fallauswahl wird nach der schließenden geschweiften Klammer `}` beendet.



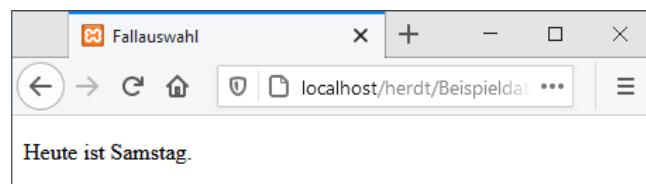
Die case-Zweige der switch-Anweisung werden der Reihe nach geprüft. Ergibt die Prüfung eine Übereinstimmung, arbeitet PHP den direkt folgenden Anweisungsblock dieses case-Zweigs bis zur Anweisung break ab. Ist kein break am Ende eines case-Zweiges angegeben, werden auch alle nachfolgenden case-Blöcke überprüft und wenn gültig, die folgenden Anweisungsblöcke ausgeführt, bis eine break-Anweisung erfolgt.

Beispiel: *switch-case-1.php*

Nachfolgend ein Beispiel, in dem der Wert einer Variablen geprüft wird:

```
<?php
①    $tag = "Samstag"; // Zum Testen Tag verändern
      // Test auf den Wochentag
②    switch ($tag) {
③        case "Samstag":
            echo "<p>Heute ist Samstag.</p>";
            break;
④        case "Sonntag":
            echo "<p>Heute ist Sonntag.</p>";
            break;
⑤        default:
            echo "<p>Schade, leider kein Wochenende.</p>";
    }
?>
```

- ① Der Variablen \$tag wird der Wert Samstag zugewiesen.
Zum Testen können Sie an dieser Stelle den Tag ändern.
- ② In der switch-Anweisung wird angegeben, dass die Variable \$tag geprüft werden soll.
- ③+④ Nacheinander werden die case-Zweige abgearbeitet. In diesem Beispiel findet die Prüfung auf Samstag und Sonntag statt. Die Prüfung entspricht einer if-Anweisung mit der Bedingung if (\$tag == "Samstag") usw. Die Prüfung, welche den Wert TRUE liefert, ist in Zeile ③ zu finden. Der folgende Anweisungsblock wird bis zum Befehl break ausgeführt. Die case-Überprüfung in Zeile ④ und der default-Block ⑤ wird nicht mehr durchlaufen, da das break die komplette switch-Anweisung beendet.
- ⑤ Falls keiner der case-Zweige zutrifft, wird der Anweisungsblock im default-Zweig ausgeführt.



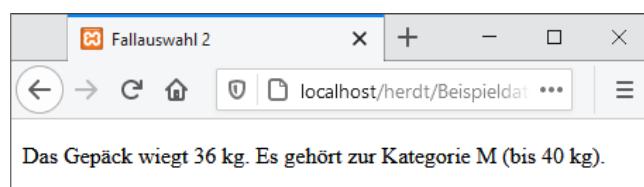
Ausgabe der Datei „switch-case-1.php“

Beispiel: switch-case-2.php

Bei einer switch-Anweisung können Sie nicht nur auf einen bestimmten Wert prüfen, sondern auch auf bestimmte Bedingungen. Statt dem Wert hinter dem case können Sie auch eine Bedingung angeben. Ist das Ergebnis der Prüfung TRUE, wird der entsprechende case-Zweig ausgeführt. In den case-Prüfungen wird nach einer Prüfung gesucht, die das Ergebnis TRUE liefert. Von daher leiten Sie die switch-Anweisung mit dem Wert TRUE ein. Es wäre auch denkbar, die Logik umzudrehen und das switch mit einem FALSE zu beginnen. Dann würde die switch-Anweisung die falschen Prüfungen finden.

```
<?php
    $gewicht = 36; // Beispiel-Gewicht eines Gepäckstücks in kg
    echo "<p>Das Gepäck wiegt $gewicht kg. Es gehört zur
          Kategorie ";
    switch (true) {
        ①    case ($gewicht <= 20):
            echo " S (bis 20 kg)";
            break;
        ②    case ($gewicht <= 40):
            echo " M (bis 40 kg)";
            break;
        default:
            echo " L (über 40 kg)";
    }
    echo ".</p>";
?>
```

- ①+② Anders als im ersten Beispiel wird nicht auf einen bestimmten Wert, sondern auf eine Bedingung geprüft.



- ② Die Prüfung von Gewicht 36 gibt im case `$gewicht <= 40` ein TRUE zurück. Der folgende Anweisungsblock wird ausgeführt und die switch-Anweisung durch das break beendet.

Bei einer switch-Anweisung können Sie alle denkbaren Fälle prüfen. Dies ist möglich, da in den case-Zweigen der zu testende Wert auf **Gleichheit** geprüft wird:

- ✓ numerische Werte, z. B. `case 5`: bei der Prüfung, ob eine Variable den Wert 5 hat
- ✓ Zeichenketten, z. B. `case "Spanischer Rotwein"`: bei der Prüfung, ob der Wert einer Variablen identisch mit der Zeichenkette "Spanischer Rotwein" ist
- ✓ einfache und verknüpfte Bedingungen, z. B. `case ($a > 10 && $a <= 20)`: bei der Prüfung, ob die Variable \$a einen Wert zwischen 10 und 20 aufweist.

Beispiel: switch-case-3.php

Darüber hinaus gibt es die Möglichkeit, mehrere Werte gleichzeitig zu prüfen. Geben Sie hierfür das Schlüsselwort `case` mehrfach mit den Werten an, welche Sie prüfen möchten.

```
<?php  
①    $note = 3;  
    switch ($note) {  
        ②    case 1: case 2: case 3: case 4:  
            echo "<p>Test bestanden.</p>";  
            break;  
        ③    case 5:  
        case 6:  
            echo "<p>Test leider nicht bestanden.</p>";  
            break;  
        ④    case "nicht bewertet":  
            echo "<p>Der Test wurde abgebrochen und daher nicht  
                bewertet.</p>";  
            break;  
        ⑤    default:  
            echo "<p>Ich kann keine ganze Note zwischen 1 und 6  
                erkennen.</p>";  
    }  
?>
```

- ① Der Variablen `$note` wird der Wert 3 zugewiesen. Zum Testen können Sie die Wertzuweisung ändern.
- ② In diesem `case`-Zweig der `switch`-Anweisung werden mehrere Fälle zusammengefasst. Hier sind sie einfach hintereinander notiert, jede Fallprüfung wird nach dem Prüfwert mit einem Doppelpunkt [:] vervollständigt.
- ③ Im diesem `case`-Zweig werden mehrere Fälle untereinander angegeben, jeweils komplett mit Prüfwert und dem folgenden Doppelpunkt [:]. Für PHP spielt es keine Rolle, ob die `case`-Fälle in einer oder in mehreren Zeilen stehen.
- ④ An dieser Stelle erfolgt die Prüfung, ob die Variable `$note` mit der angegebenen Zeichenkette "nicht bewertet" übereinstimmt.
- ⑤ Die `switch`-Anweisung bietet in diesem Beispiel einen optionalen `default`-Zweig, dessen dazugehöriger Anweisungsblock nur dann ausgeführt wird, wenn die Variable `$note` einen anderen Inhalt hat als 1, 2, 3, 4, 5, 6 oder nicht bewertet.

4.7 Fallzuweisung mit dem `match`-Ausdruck

Neu in PHP 8 ist der `match`-Ausdruck. Er erinnert zum einen an die `switch`-Anweisung, bei der ein Parameter mit verschiedenen Bedingungen geprüft wird, gibt aber einen Wert zurück, vergleichbar mit dem ternären Operator.

Syntax und Bedeutung des `match`-Ausdrucks

```
$ergebnis = match(Wert) {
    Bedinung1=> Rückgabewert1,
    Bedinung2, Bedinung3 => Rückgabewert2
};
```

- ✓ Das Schlüsselwort `match` leitet den `match`-Ausdruck ein. In runden Klammern wird ein Wert zur Überprüfung übergeben.
- ✓ Dem `match`-Ausdruck und dem Wert in runden Klammern folgen geschweifte Klammern `{ }` , welche die Bedingungen umfassen.
- ✓ Ein einzelner Eintrag in dem Ausdruck besteht aus der Bedingung, gefolgt von dem Zeichen `= >` und dem dahinter notierten Wert.
- ✓ In einem Eintrag können mehrere Bedingungen, getrennt mit Kommas `,`, angegeben werden.
- ✓ Es können beliebig viele Einträge angegeben werden, diese werden ebenfalls mit Kommas `,` voneinander abgegrenzt.
- ✓ Die Prüfung der Bedingung ist eine Prüfung auf Identität, neben dem Wert wird hier auch der Datentyp geprüft, entsprechend der Prüfung mit einem Identisch-Operator `= = =`.
- ✓ Gibt den ersten gültigen Treffer zurück, durchläuft keine weiteren Bedingungen wie bei einem `switch` ohne `break`
- ✓ Der `match`-Ausdruck liefert immer einen Wert zurück. Trifft keine gültige Bedingung zu, wirft PHP eine Fehlermeldung vom Typ `error` aus, das Skript bricht dann ab.
- ✓ Statt einem festen Rückgabewert kann auch eine Funktion (siehe Kapitel 7) aufgerufen werden.
- ✓ Der `match`-Ausdruck **muss** mit einem Semikolon `;` abgeschlossen werden. Fehlt dieses, meldet PHP einen Syntax-Fehler.

Beispiel: `match-1.php`

```
<?php
① $farbe = 'rot'; // Zum Testen verändern
② $ergebnis = match ($farbe) {
    "gruen", => "0 gewinnt.",
    "rot", => "Rote Zahlen gewinnen.",
    "schwarz", => "Schwarze Zahlen gewinnen.",
};
③ echo "<p>Der Farbe ist $farbe.</p>";
echo "<p><code>match()</code> gab zurück: $ergebnis</p>";
?>
```

- ① Die Variable \$farbe wird deklariert und initialisiert.
- ② Im match-Ausdruck wird die Farbe überprüft. match gibt den *string* Rote Zahlen gewinnen. zurück.
- ③ Zur Kontrolle werden hier der Wert der Variablen \$farbe und \$ergebnis ausgegeben.

Ausgabe der Beispieldatei match-1.php

Mehrfache Bedingungen und Identisch-Prüfung

Vergleichbar zum switch können Sie in einem Eintrag mehrere Bedingungen hinterlegen. Ebenfalls berücksichtigt match die Prüfung der Datentypen. Dies entspricht der Prüfung per Identisch-Operator `==` `!=` `===`.

Beispiel: match-2.php

```
<?php
①    $zahl = 3; // Zum Testen verändern
②    $ergebnis = match ($zahl) {
③        "1", "2", "3", "4", "5", => "Die \$zahl $zahl ist ein
<i>string</i>",
④        1, 2, 3, 4, 5 => "Die \$zahl $zahl ist ein
<i>integer</i>"
    };
⑤    echo "<p>$ergebnis</p>";
?>
```

- ① Es wird die Variable \$zahl deklariert und initialisiert.
- ② Diese wird dem match-Ausdruck übergeben.
- ③ In beiden Einträgen werden nun mehrere Bedingungen hinterlegt, jeweils mit Komma `,` voneinander getrennt. Da im ersten Eintrag die Zahlen in Anführungszeichen `" "` stehen, handelt es sich hier um Zeichenketten.
- ④ Im zweiten Eintrag sind die gleichen Zahlen hinterlegt, allerdings ohne Anführungszeichen, und werden so von PHP mit dem Datentyp *int* erkannt.
- ⑤ Das Ergebnis wird zur Überprüfung ausgegeben. Es wird deutlich, dass im match-Ausdruck die Bedingung des Integer zutraf und der entsprechende Wert zurückgegeben wurde.

Ausgabe der Beispieldatei match-2.php

4.8 Schleifen

Schleifen verwenden

Um einen bestimmten Teil des Programms mehrfach auszuführen bzw. zu wiederholen, benötigen Sie Schleifen. Durch die Verwendung von Schleifen sparen Sie Programmcode und können variabel festlegen, wie oft oder bis zum Eintreffen welcher Bedingung die Schleife durchlaufen werden soll.

Folgende Schleifen gibt es in PHP:

- ✓ while- bzw. do-while-Schleife:

In diesen Schleifen wird die Anzahl der Durchläufe von Bedingungen abhängig gemacht, die sich erst im Programmverlauf ergeben, z. B. wenn ein Wert über 100 ist. Wie oft diese Schleifen durchlaufen werden, ist also variabel.

- ✓ for-Schleife:

In for-Schleifen definieren Sie die Anzahl der Durchläufe selbst, wenn Sie z. B. die gleiche Anweisung genau 10-mal ausführen möchten. Die Schleife wird dann grundsätzlich so oft ausgeführt, wie Sie es definiert haben (solange Sie die Durchläufe nicht mit `break` oder `continue` beeinflussen).

- ✓ foreach -Schleife:

Bei einer `foreach`-Schleife hängt die Anzahl der Durchläufe von der Variablen ab, die im `foreach` verwendet wird. Da hierfür das Verständnis von Arrays notwendig ist, wird diese Schleife in Kapitel 5.6 vorgestellt.

4.9 Mit der `while`-Schleife arbeiten

In einer `while`-Schleife prüft PHP zu Beginn die vorgegebene Bedingung und führt den angegebenen Anweisungsblock so lange aus, bis die Bedingung nicht mehr erfüllt ist. Diese Prüfung wird automatisch **vor** jedem Durchlauf wiederholt. Ergibt der Test der Bedingung den Wert `TRUE`, wird die Schleife durchlaufen. Liefert die Bedingung den Wert `FALSE` zurück, werden die Anweisungen der Schleife übersprungen und die Abarbeitung des Programms wird nach der Schleife fortgesetzt.

Eine Prüfung **vor** der Ausführung des Anweisungsblocks wird als **kopfgesteuert** bezeichnet. Ergibt bereits die erste Bedingungsprüfung den Wert `FALSE`, wird der Anweisungsblock gar nicht ausgeführt, d. h., eine `while`-Schleife kann 0 Mal bis unendlich oft (sogenannte Endlosschleife) ausgeführt werden.

Syntax und Bedeutung der `while`-Schleife

- ✓ Das Schlüsselwort `while` leitet die `while`-Schleife ein. In runden Klammern wird die Bedingung angegeben, die vor der Abarbeitung überprüft wird.
- ✓ Ist der Rückgabewert der Bedingung `TRUE`, wird der Anweisungsblock im Schleifenkörper ausgeführt.

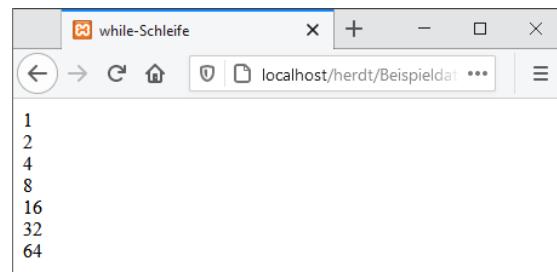
```
while(Bedingung) {  
    Anweisungsblock;  
}
```

- ✓ Nach der letzten Anweisung des Anweisungsblocks bzw. mit Erreichen der schließenden geschweiften Klammer springt PHP zum Anfang der Schleife zurück und überprüft erneut die Bedingung. Ist das Ergebnis der Bedingungsprüfung TRUE, wird der Anweisungsblock erneut ausgeführt.
- ✓ Diese Schritte werden so lange wiederholt, bis die Auswertung der Bedingungsprüfung am Beginn der Schleife den Wert FALSE liefert. Die Schleife wird dann verlassen und der PHP-Code nach der Schleife weiter abgearbeitet.

Beispiel: *while.php*

```
<?php
① $zahl = 1;
② while ($zahl <= 100) {
③   echo "$zahl<br>";
④   $zahl = $zahl + $zahl;
⑤ }
?>
```

- ① Der Startwert der Variablen \$zahl wird auf den Wert 1 festgelegt.
- ② Die Bedingung legt fest, dass die while-Schleife wiederholt werden soll, so lange der Wert der Variablen \$zahl kleiner oder gleich 100 ist. Bedenken Sie, dass die Schleife nicht ein einziges Mal ausgeführt wird, wenn der Wert der Variablen bereits beim ersten Schleifendurchlauf größer als 100 sein sollte.
- ③ In jedem Schleifendurchlauf erfolgt die Ausgabe der Variablen \$zahl.
- ④ Der Variablen \$zahl wird an dieser Stelle ein neuer Wert zugewiesen, im Beispiel wird sie verdoppelt.
- ⑤ Die while-Schleife wird durch die schließende Klammer ⑤ beendet. Nach jedem Schleifendurchlauf wird die Bedingung erneut geprüft ②. Der Anweisungsblock in der Schleife wird so lange ausgeführt, bis die Bedingungsprüfung FALSE ergibt. Damit ist dann die Schleife beendet und die Abarbeitung des Skripts wird nach der Schleife fortgesetzt.



Anzeige der Beispieldatei „while.php“

Häufige Fehler beim Einsatz von Schleifen

Die in diesem Abschnitt beschriebenen Fehler bewirken keine Fehlermeldungen, die von PHP gemeldet werden, sondern einzig ein unerwartetes Verhalten des Programms. Es handelt sich um **logische Fehler**:

- ✓ Es kann passieren, dass der Variablen, die geprüft werden soll, ein Wert zugewiesen wurde, bei dem die Bedingungsprüfung am Anfang der Schleife gleich beim ersten Schleifendurchlauf den Wert FALSE ergibt. Das heißt, die Schleife wird nicht ein einziges Mal ausgeführt, sondern übersprungen.

Endlosschleifen vermeiden

Schleifen bergen grundsätzlich die Gefahr, dass sie durch eine falsche Logik in der Programmierung in eine Endlosschleife laufen. Dies wird Ihnen nicht als PHP-Fehler gemeldet. Der Aufruf der Datei ist möglich, das Skript bricht ab, wenn die vom Server erlaubte Laufzeit erreicht ist (diese wird in der *php.ini* über den Parameter `max_execution_time` bestimmt und ist seit PHP 8 standardmäßig auf 120 Sekunden eingestellt (früher 30 Sekunden)). Aber auch das „Volllaufen“, also das Überschreiten des erlaubten Speicherlimits (*Allowed memory size*, wird über den Parameter `memory_limit` in der *php.ini* konfiguriert) kann Ihr Skript zum „Absturz“ bringen, wenn z. B. durch zu viele Schleifendurchläufe große Datenmengen verarbeitet werden sollen. Dies kann zum Absturz Ihres Browsers oder sogar des Webservers führen.

Endlosschleifen können Sie nur durch eine sorgfältige Programmierung vermeiden. Eine Möglichkeit ist z. B., in einer Hilfsvariablen die Anzahl der Durchläufe mitzuzählen und ab einer bestimmten Anzahl von Durchläufen die Schleife abzubrechen (vgl. Abschnitt 4.11).

Mit der `do-while`-Schleife arbeiten

Im Unterschied zur `while`-Schleife, die die Bedingung für die Wiederholung am Anfang hat, wird in der `do-while`-Schleife die Bedingung erst **nach** der letzten Anweisung in der Schleife überprüft. Man spricht deshalb von einer **fußgesteuerten** Schleife. Daraus folgt auch, dass der Anweisungsblock immer mindestens einmal ausgeführt wird. Auch hier gilt wie bei der `while`-Schleife: Liefert die Bedingung einen Wert `TRUE` zurück, wird die Schleife erneut durchlaufen, ansonsten wird sie verlassen.

Syntax und Bedeutung der `do-while`-Schleife

- ✓ Das Schlüsselwort `do` leitet die Schleife ein.
- ✓ Die Anweisungen im Schleifenkörper werden auf jeden Fall mindestens einmal ausgeführt.
Am Ende der Schleife wird die Bedingung der Schleife über das Schlüsselwort `while` ausgewertet. Die Bedingung wird in runden Klammern angegeben.
- ✓ Trifft die Bedingung zu, wird die Schleife erneut ausgeführt.

```
do {
    Anweisungsblock;
}
while (Bedingung);
```

4.10 Mit der `for`-Schleife arbeiten

Im Unterschied zur `while`-Schleife, die so oft durchlaufen wird, bis der Test der Bedingung `FALSE` ergibt, wird in der `for`-Schleife genau angegeben, wie oft die Schleife durchlaufen werden soll. Bei der `for`-Schleife kann berechnet werden:

- ✓ die Anzahl der Wiederholungen oder
- ✓ Beginn und Ende der Wiederholung.

Übergeben werden der Start- und Endwert sowie die Bedingung, wie der Endwert erreicht werden soll.

Syntax und Bedeutung der **for**-Schleife

```
for ( Initialisierung; Bedingung; Reinitialisierung ) {
    Anweisungsblock;
}
```

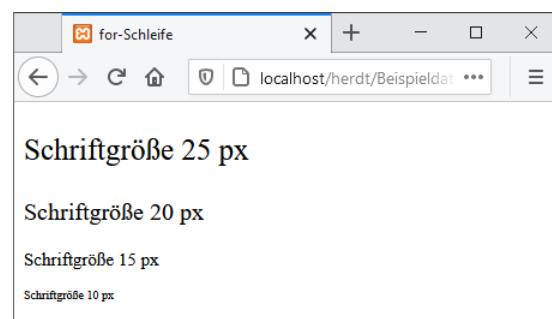
- ✓ Die **for**-Schleife beginnt mit dem reservierten Schlüsselwort **for**.
- ✓ Mit der Anweisung **Initialisierung** wird der Anfangswert für die Schleife festgelegt.
- ✓ Die Anweisung **Bedingung** legt fest, unter welchen Konditionen die Schleife durchlaufen wird. Diese Bedingung, die vor jedem Durchlauf abgefragt wird, liefert einen Wert **TRUE** oder **FALSE** zurück. Ist der Wert **TRUE**, wird die Schleife durchlaufen. Andernfalls wird das Programm mit den Anweisungen nach der **for**-Schleife fortgesetzt. Die Bedingung überprüft den Initialwert noch vor dem ersten Durchlauf. Ist das Ergebnis dieser ersten Überprüfung bereits mit den Initialwerten **FALSE**, wird die Schleife **nicht** durchlaufen.
- ✓ Die Anweisung **Reinitialisierung** legt fest, wie die Variable verändert werden soll, die in der **Bedingung** geprüft wird. Diese Veränderung geschieht am Ende der Schleife, das bedeutet, erst beim zweiten Durchlauf der Schleife ist der Wert, der in der **Initialisierung** gesetzt wurde, verändert.
- ✓ Die drei Anweisungen werden jeweils durch ein Semikolon voneinander getrennt.

Beispiel: *for.php*

Die Schrift eines angezeigten Beispieltextes wird mithilfe einer Schleife schrittweise verkleinert.

```
<?php
①  for ($groesse = 25; $groesse >= 10; $groesse -= 5) {
②      echo "<p style='font-size:" . $groesse . "px'>Schriftgröße
           $groesse px</p>";
③  }
?>
```

- ① Der Startwert der **for**-Schleife wird über die Zuweisung des Wertes 25 an die Variable **\$groesse** festgelegt. Die Bedingung besagt, dass die Schleife durchlaufen werden soll, solange der Wert der Variablen **\$groesse** größer oder gleich 10 ist. Nach jedem Durchlauf wird der Wert um 5 verringert.
- ② Die Variable **\$groesse** steuert über ihren Wert die Schriftgröße der betreffenden Zeile und wird per **echo** ausgegeben.
- ③ Das Ende der **for**-Schleife ist erreicht. Durch die Reinitialisierung wird der Wert der Variablen **\$groesse** um 5 subtrahiert und springt wieder zurück zur Zeile ①. Dort wird erneut geprüft, ob die Schleife nochmals durchlaufen werden muss.



Anzeige der Beispieldatei „for.php“

Beispiele für Operatoren in **for**-Schleifen

Bei der Arbeit mit einer **for**-Schleife können Sie z. B. mit folgenden Operatoren arbeiten:

Bedingung in der for -Schleife	Werte von \$i
for (\$i = 1; \$i <= 5; \$i++)	1, 2, 3, 4, 5
for (\$i = 1; \$i < 5; \$i++)	1, 2, 3, 4
for (\$i = 15; \$i >= 10; \$i--)	15, 14, 13, 12, 11, 10
for (\$i = 15; \$i > 10; \$i--)	15, 14, 13, 12, 11
for (\$i = 0; \$i < 100; \$i = \$i + 10)	0, 10, 20, 30, 40, 50, 60, 70, 80, 90
for (\$i = 1; \$i <= 10; \$i = \$i + 1.2)	1, 2.2, 3.4, 4.6, 5.8, 7, 8.2, 9.4

4.11 Schleifen abbrechen

Schleifenabbruch mit **break**

Nicht immer ist es notwendig, eine **for**-, **while**- oder **do-while**-Schleife so lange zu durchlaufen, bis die definierte Bedingung ein **FALSE** ergibt und damit die Schleife beendet wird. Auch innerhalb des Schleifendurchlaufs können andere Kriterien geprüft werden, die das weitere Durchlaufen der Schleife überflüssig machen.

Beispiel: *break.php*

```
<?php
    $budget = 50;
    $einzelpreis = 9;
    $menge = 1;
    while ($menge <= 15) { // Idealfall: 15 Stück kaufen
        $gesamtpreis = $einzelpreis * $menge;
        if ($gesamtpreis > $budget) { // Budget erschöpft? Wenn
            ja: Abbruch
            echo "<p><strong>Ihr Budget ist leider
            erschöpft.</strong></p>";
            break;
        }
        echo "<p>$menge Stück: $gesamtpreis Euro.</p>";
        $menge++;
    }
?>
```

- ① Der Startwert der while-Schleife wird über die Zuweisung des Wertes 1 an die Variable \$menge festgelegt.
- ② Die Bedingung besagt, dass die Schleife durchlaufen werden soll, solange der Wert der Variablen \$menge kleiner oder gleich 15 ist.
- ③ Die Schleife soll allerdings nur ausgeführt werden, bis der Gesamtpreis das zur Verfügung stehende Budget übersteigt. Wenn das passiert, wird die Schleife durch den Befehl break ④ im Anschluss an eine Meldung abgebrochen.
- ⑤ Die Variable \$menge wird bei jedem Schleifendurchlauf um eins erhöht (\$menge++).

```
<?php
$menge = 1;
$preis = 9;
$budget = 50;

while($menge <= 15) {
    echo $menge . " Stück: " . $preis . " Euro." . "\n";
    $menge++;
    $preis += 9;
}

echo "Ihr Budget ist leider erschöpft.";
```

Anzeige der Beispieldatei „break.php“

Vorzeitiger Sprung zum nächsten Schleifendurchlauf mit continue

In Schleifen wird der PHP-Code vom Schleifenkopf bis zum Schleifenende vollständig durchlaufen. Es ist jedoch denkbar, dass bei einer bestimmten Bedingung das Ausführen des Codes bis zum Schleifenende überflüssig ist, allerdings sollen weitere Schleifendurchläufe ausgeführt werden.

Für diesen Fall steht das Schlüsselwort `continue` zur Verfügung. Damit beenden Sie einen einzelnen Schleifendurchlauf, die Schleife wird mit dem nächsten Schleifendurchlauf fortgesetzt. Bei `for`-Schleifen wird hier auch die Anweisung der Reinitialisierung ausgeführt, auch wenn das Ende der Schleife noch nicht erreicht wurde.

Beispiel: `continue.php`

```
<?php
$zaehler = 5;
for ($nenner = -2; $nenner <= 2; $nenner++) {
    if ($nenner == 0) {
        echo "<p>Division durch 0 ist verboten.</p>";
        continue;
    }
    $ergebnis = $zaehler / $nenner;
    echo "<p>$zaehler / $nenner = " . $ergebnis . "</p>";
}
?>
```

- ① In einer `for`-Schleife sollen die Ergebnisse einer Division von `$zaehler` und `$nenner` berechnet und ausgegeben werden. `$zaehler` besitzt mit 5 einen festen Wert. Der Wert der Variablen `$nenner` wird durch die `for`-Schleife verändert (von -2 bis 2).
- ② Um keinen schwerwiegenden Fehler (Division durch 0) auszulösen, soll die Division nicht durchgeführt werden, wenn die Variable `$nenner` den Wert 0 aufweist. In dem Fall wird der **aktuelle Schleifendurchlauf** nach einer `echo`-Anweisung per `continue` ③ beendet.

Anzeige der Beispieldatei „continue.php“

- Beachten Sie, dass die Reinitialisierung `$nenner++` auch ausgeführt wird, wenn das Schleifenende noch nicht erreicht ist. Die Variable `$nenner` wird um 1 hochgezählt und die Schleife mit diesem Wert fortgesetzt.
- ④ Sofern der aktuelle Schleifendurchlauf nicht abgebrochen wurde, erfolgt an dieser Stelle die Berechnung und anschließend die Ausgabe des Ergebnisses.

Plus **Wissenstest:** Variablen bis Kontrollstrukturen

4.12 Übungen

Übung 1: Kontrollstrukturen in PHP: `switch`

Level		Zeit	ca. 5 min
Übungsinhalte	<ul style="list-style-type: none"> ✓ <code>switch</code>-Anweisung ✓ <code>case</code> und <code>break</code> 		
Übungsdatei	--		
Ergebnisdatei	<code>bewertung_switch.php</code>		

1. Erstellen Sie ein PHP-Skript (`bewertung_switch.php`), das die Bewertung eines Tests in Textform ausgibt. Realisieren Sie diese Aufgabe mithilfe einer `switch`-Auswahlschleife. Legen Sie die erreichte Punktzahl innerhalb des Skripts fest. Die Bedeutung der Punkte lautet:

10 Punkte:	Sehr gut
9 Punkte:	Gut
8 Punkte:	Befriedigend
7 Punkte:	Ausreichend
weniger als 7:	Leider zu wenige Punkte erreicht

Übung 2: Kontrollstrukturen in PHP: `for`

Level		Zeit	ca. 10 min
Übungsinhalte	<ul style="list-style-type: none"> ✓ switch-Anweisung ✓ case und break ✓ for-Schleife 		
Übungsdatei	--		
Ergebnisdatei	<code>bewertung_switch-2.php</code>		

2. Erweitern Sie das Programm `bewertung_switch.php`, indem Sie eine Schleife programmieren, die automatisch die Bewertung des Tests mithilfe einer `for`-Schleife für erreichte Punktzahlen von 10 bis 0 Punkte ausgibt. Speichern Sie die Datei unter dem Namen `bewertung_switch-2.php`.

```

bewertung_switch-2.php

10 Punkte ergeben folgende Bewertung: Sehr gut
9 Punkte ergeben folgende Bewertung: Gut
8 Punkte ergeben folgende Bewertung: Befriedigend
7 Punkte ergeben folgende Bewertung: Ausreichend
6 Punkte ergeben folgende Bewertung: Leider zu wenige Punkte erreicht.
5 Punkte ergeben folgende Bewertung: Leider zu wenige Punkte erreicht.
4 Punkte ergeben folgende Bewertung: Leider zu wenige Punkte erreicht.
3 Punkte ergeben folgende Bewertung: Leider zu wenige Punkte erreicht.
2 Punkte ergeben folgende Bewertung: Leider zu wenige Punkte erreicht.
1 Punkte ergeben folgende Bewertung: Leider zu wenige Punkte erreicht.
0 Punkte ergeben folgende Bewertung: Leider zu wenige Punkte erreicht.

```

Anzeige der Ergebnisdatei „bewertung_switch-2.php“

Übung 3: Kontrollstrukturen in PHP: **while** und **do-while**

Level		Zeit	ca. 15 min
Übungsinhalte	<ul style="list-style-type: none"> ✓ while-Schleife ✓ do-while-Schleife 		
Übungsdatei	--		
Ergebnisdatei	<i>while_do-while.php</i>		

1. Entwerfen Sie ein Programm (*while_do-while.php*), das untereinander die Zahlen 1–5 ausgibt. Weisen Sie einer Variablen `$zahl` den Startwert 1 zu und verwenden Sie eine `while`-Schleife. Realisieren Sie die Aufgabe anschließend zusätzlich in derselben Datei mit einer `do-while`-Schleife.
2. Wenn Sie die Ausgaben von 1–5 realisiert haben, weisen Sie der Variablen `$zahl` den Startwert 20 zu. Vergleichen Sie das Verhalten der `while`-Schleife mit dem der `do-while`-Schleife.

The figure consists of two side-by-side screenshots of a web browser window. Both windows have the title "Übung (Kapitel 4)". The left window displays the output of a PHP script with a start value of 1. It shows two sections: "while-Schleife (Startwert: 1)" containing the numbers 1, 2, 3, 4, 5, and "do-while-Schleife (Startwert: 1)" also containing the numbers 1, 2, 3, 4, 5. The right window displays the output of the same script with a start value of 20. It shows one section: "while-Schleife (Startwert: 20)" containing the number 20.

Anzeige der Ergebnisdatei „*while_do-while.php*“ mit Startwert 1 (links) und Startwert 20 (rechts)
für die Variable `$zahl`

5

Arrays



Beispieldateien: Dateien im Ordner *Kapitel_05*

5.1 Grundlagen zu Arrays

Was sind Arrays?

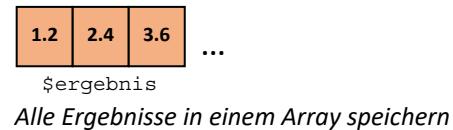
Bislang wurden in den Beispielen Variablen verwendet, die jeweils nur einen einzelnen Wert speichern. In diesem Kapitel lernen Sie Arrays kennen. Arrays sind spezielle Variablen, die nicht nur einen, sondern beliebig viele Werte (auch unterschiedlichen Datentyps) speichern können. Im Sprachgebrauch von Entwicklern ist die englische Bezeichnung „Arrays“ üblich. Mitunter werden Arrays auch Feldvariablen oder kurz **Felder** genannt.

Arrays sind hilfreich für die reihenweise Verarbeitung von Daten. Stellen Sie sich vor, eine Berechnung liefert Ihnen 20 Ergebnisse. Diese möchten Sie für die spätere Weiterverarbeitung zwischenspeichern. Mit den bislang bekannten Datentypen *boolean*, *int*, *float* oder *string* können Sie pro Variable nur einen Wert speichern. Um 20 Ergebnisse zu speichern, bräuchten Sie nun 20 einzelne Variablen (*\$ergebnis1*, *\$ergebnis2*, *\$ergebnis3*, *\$ergebnis4* ...). Dies ist umständlich und arbeitsaufwändig. In Situationen, in denen Sie die Anzahl der einzelnen Ergebnisse nicht kennen, wäre das Zwischenspeichern mit Einzelvariablen ebenfalls sehr umständlich zu lösen. Arrays bieten hier die Lösung: Alle Ergebnisse können in **einer** Array-Variablen gespeichert werden.

Nicht nur für das Speichern von zusammengehörigen Werten, sondern auch für die weitere Verarbeitung bieten sich Arrays an. Arrays können einfach mit Schleifen durchlaufen werden. Das ermöglicht eine schnelle Verarbeitung vieler Werte, ohne dass Sie für jeden einzelnen Wert eine einzelne Variable definieren müssen.

Bildlich können Sie sich Arrays wie eine Kommode mit vielen Schubladen vorstellen. Die Kommode wäre das Array, die einzelnen Schubladen die einzelnen Array-Einträge – wobei in jeder Schublade auch wieder eine kleine Kommode mit weiteren Schubladen sein kann (mehrdimensionales Array, siehe nächste Seite).

Der Zugriff auf die einzelnen Ergebnisse erfolgt jeweils über ihre Position innerhalb des Arrays.



Arrays in PHP

Jedes Element eines Arrays besteht aus einem Index bzw. Schlüssel und einem Datenwert. Dabei gibt es zwei Sorten von Arrays, die sich im Aufbau unterscheiden, aber auch dadurch, wie Sie auf die einzelnen Einträge in einem Array zugreifen können:

- ✓ Im **numerisch indizierten** Array werden die einzelnen Werte (*value*) innerhalb des Arrays über einen Index, eine fortlaufende Nummer, angesprochen.
- ✓ Im **assoziativen** Array werden die einzelnen Werte (*value*) innerhalb des Arrays über einen eindeutigen Schlüssel, *key* genannt, angesprochen.

Aufbau von Arrays

Arrays können ein- oder mehrdimensional sein:

- ✓ In einem **eindimensionalen** Array legen Sie eine Liste von Daten ab. Sie können beispielsweise eine Liste von Städten darstellen.
- ✓ Wenn Sie zusammengehörige verschachtelte Daten wie z. B. eine Liste von Ländern und dazugehörigen Werten wie Hauptstädten, Landessprache etc. abbilden möchten, können Sie dies mit einem **mehrdimensionalen** Array tun. Dabei ist es unerheblich, ob es sich um zwei oder mehrere Dimensionen handelt. Mehrdimensionale Arrays sind Array-Variablen, in denen einzelne Einträge selbst Array-Variablen sind.

Eigenschaften von Arrays

- ✓ In PHP können die Werte von Arrays von beliebigen Datentypen sein.
- ✓ Falls Sie keinen Index vergeben, erzeugen Sie automatisch ein **numerisch indiziertes** Array, die Array-Indizes beginnen standardmäßig mit 0. Das erste Element hat dementsprechend den Index 0, das zweite Element hat den Index 1, das dritte Element hat den Index 2 und so weiter. Der Index eines Arrays mit *n* Elementen reicht somit von 0 bis *n* - 1. Der Index eines Arrays mit vier Elementen reicht somit von 0 - 3.
- ✓ Arrays müssen in PHP nicht explizit definiert werden. Sie erzeugen ein Array, in dem Sie einer Variablen per `=` und dem Schlüsselwort `array()` die Array-Werte zuweisen – alternativ über die Kurzschreibweise `=` und `[]` (doppelte eckige Klammern). PHP weist dann automatisch den Datentyp *array* zu.
- ✓ Die Anzahl der Array-Einträge muss vorher nicht angegeben werden. Die Größe eines Arrays hängt von der Anzahl der Einträge ab, mit der Sie das Array erzeugen. Außerdem können Sie nach der Erstellung dem Array zusätzliche Elemente hinzufügen und einzelne Elemente löschen.

5.2 Indizierte eindimensionale Arrays erstellen

Ein indiziertes Array anlegen

Ein Array erzeugen Sie in PHP über das Schlüsselwort `array()`. In einem Schritt können Sie dem Array einfach und schnell die Werte zuweisen.

Syntax und Bedeutung der `array()`-Funktion

```
$feld = array(Wert1, Wert2, Wert3, ...);
```

- ✓ Um ein Array zu erzeugen, verwenden Sie das Schlüsselwort `array()`.
- ✓ Über den Zuweisungsoperator `=` weisen Sie das Array der Variablen `$feld` zu.
- ✓ Die einzelnen Array-Einträge notieren Sie innerhalb der runden Klammern.
- ✓ Die einzelnen Werte werden durch Kommata voneinander getrennt.
- ✓ `int`, `double` und `boolean`-Werte werden ohne, `string` (Zeichenketten) mit Anführungszeichen angegeben.
- ✓ Die Indizierung der einzelnen Elemente erfolgt automatisch in der Reihenfolge der Angabe.

Beispiel

```
$staedte = array("Frankfurt", "Berlin", "Bern");
```

Es befinden sich drei Werte in dem Array `$staedte`: Frankfurt, Berlin und Bern. Die Reihenfolge, in der die Werte zugewiesen worden sind, bestimmt die automatische Indizierung der Werte. Im Array ist dann jeder Wert mit einem Index fest verknüpft, also Index 0 mit Frankfurt usw.

Index	0	1	2
Wert	Frankfurt	Berlin	Bern

Auf indizierte Arrays zugreifen

Um auf ein bestimmtes Element in einem Array zuzugreifen, wird der Index des einzelnen Wertes verwendet. Bei numerisch indizierten Arrays ist dies der automatisch erzeugte fortlaufende Index. Der Index wird dabei in eckigen Klammern direkt hinter dem Variablennamen angegeben:

```
$staedte [Index];
```

Beispiel

```
echo $staedte [2];
```

Hier wird der Eintrag der Array-Variablen `$staedte` mit dem Indexwert 2 ausgegeben.

5.3 Assoziative eindimensionale Arrays erstellen

Ein assoziatives Array anlegen

Eine andere Art von Arrays sind **assoziativen Arrays**. Bei diesen werden für den Zugriff keine fortlaufenden Indizes benutzt, sondern Schlüssel, über deren Wert auf die einzelnen Werte zugegriffen werden kann. Der Vorteil assoziativer Arrays besteht darin, dass Sie zwei Informationen (ein Informationspaar) ablegen können. Die Schlüssel sind frei wählbar und tragen häufig eine Bedeutung – im Gegensatz zu einem numerischen und meist nicht bedeutungstragenden Index.

Syntax der `array()`-Anweisung bei assoziativen Arrays

Auch die assoziativen Arrays lassen sich über die `array()`-Anweisung füllen:

```
$feld = array(Schlüssel1 => Wert1, Schlüssel2 => Wert2, ...);
```

Beispiel:

```
$hauptstaedte = array("Schweiz" => "Bern",
                      "Frankreich" => "Paris");
```

- ✓ Die `array()`-Anweisung beginnt mit dem reservierten Wort `array`.
- ✓ Der Variablen werden die einzelnen Wertpaare, bestehend aus Schlüssel (`key`) und Wert (`value`), übergeben. Die einzelnen Wertpaare werden durch Kommata voneinander getrennt angegeben.
- ✓ Die Zuweisung von Schlüssel und Wert erfolgt über die Zeichenfolge `=>`. Dabei steht der Schlüssel links, der Wert rechts vom `=>`. Diese Zuweisung wird auch als Indizierung bezeichnet.
- ✓ Mögliche Datentypen für den Schlüssel sind `int` und `string`. Ein `double`-Wert als Schlüssel wird zu einem `int`-Wert umgewandelt, `null` zu einer leeren Zeichenkette. Arrays und Objekte sind als Schlüssel nicht erlaubt.
- ✓ Zeichenketten (`string`) als Schlüssel werden in Anführungszeichen angegeben.
- ✓ Sonderzeichen und Leerzeichen im Schlüssel funktionieren zwar, beim Einsatz von Sonderzeichen bei älteren PHP-Versionen oder abweichender Zeichensatzkonfiguration in der `php.ini` sind Probleme jedoch nicht auszuschließen. Leer- und Sonderzeichen sollten Sie von daher nicht als Array-Schlüssel verwenden.
- ✓ Der Schlüssel eines Arrays ist case-sensitiv (Groß- und Kleinschreibung wird berücksichtigt). Das heißt, "Schweiz" ist ein anderer Schlüssel als "schweiz". Beide Schlüssel können gleichzeitig in einem Array vorkommen.

Auf assoziative Arrays zugreifen

Beim Zugriff auf einen Wert des Arrays wird der Schlüssel direkt angegeben. Zum Ansprechen des Arrays kann auch eine Variable eingesetzt werden, die den Wert des Schlüssels gespeichert hat:

	Ergebnis
<code>\$auswahl = \$hauptstaedte ["Schweiz"] ;</code>	Die Variable \$auswahl hat den Wert "Bern".

oder

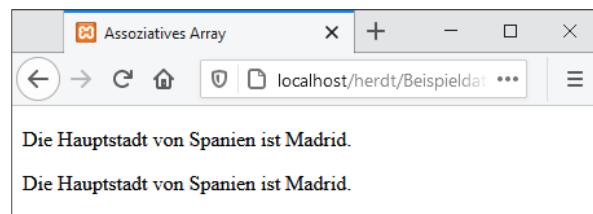
	Ergebnis
<code>\$k = "Schweiz"; \$auswahl = \$hauptstaedte [\$k] ;</code>	Die Variable \$auswahl hat den Wert "Bern".

Beispiel: array_asoz.php

Hier wird die Zuweisung der Städtenamen mit der `array()`-Anweisung realisiert.

<code><?php</code>	
①	<code>\$hauptstaedte = array("Schweiz" => "Bern", "Frankreich" => "Paris", "Deutschland" => "Berlin", "Spanien" => "Madrid");</code>
②	<code>\$k = "Spanien";</code>
③	<code>echo "<p>Die Hauptstadt von \$k ist " . \$hauptstaedte["Spanien"] . "</p>";</code>
④	<code>echo "<p>Die Hauptstadt von \$k ist " . \$hauptstaedte[\$k] . "</p>";</code>
	<code>?></code>

- ① Das assoziative Array wird über das Schlüsselwort `array()` und die Zuweisung der Schlüssel und Werte angelegt.
Die Einrückungen und Zeilenumbrüche dienen der Übersichtlichkeit des PHP-Skripts, sie sind jedoch für PHP nicht notwendig. Das ganze Array könnte auch in einer Zeile definiert sein.
- ② Der Variablen `$k` wird die Zeichenkette `Spanien` zugewiesen.
- ③ Das Array wird mit dem Schlüssel `Spanien` angesprochen und liefert den dazugehörigen Wert `Madrid`. Die Bezeichnung des Schlüssels wird als Zeichenkette und in Anführungszeichen in den eckigen Klammern angegeben.
- ④ Alternativ können Sie mit der Variablen `$k`, welche den Wert `Spanien` hat, das entsprechende Element aus dem Array auslesen und anzeigen. In diesem Fall verwenden Sie keine Anführungszeichen.



! Werden Arrays mit eigenen Schlüsseln aufgebaut, sind in dem Array keine numerisch indizierten Indizes vorhanden. `$hauptstaedte [3]` würde hier ein leeres Ergebnis liefern, da der Schlüssel 3 nicht definiert ist.

5.4 Arrays mit der Kurzschreibweise erstellen

Array-Kurzschreibweise

Alternativ zu der Syntax mit dem Schlüsselwort `array()` können Sie die Array-Kurzschreibweise einsetzen, um Arrays zu erzeugen. Über die Kurzschreibweise können Sie sowohl indizierte als auch assoziative Arrays erstellen. Die Kurzschreibweise ist schwerer zu lesen als die Array-Deklarationen mit `array()`, bietet jedoch die Möglichkeit, PHP-Code kompakter zu schreiben.

Syntax der Kurzschreibweise

```
$feld = [Wert1, Wert2, Wert3, ...];
```

- ✓ Das Schlüsselwort `array()` entfällt in der Kurzschreibweise.
- ✓ Statt der runden Klammern `()` werden die Werte in eckigen Klammern `[]` angegeben.
- ✓ Ansonsten gelten die gleichen Regeln wie bei indizierten und assoziativen Arrays.

Beispiel

```
$staedte = ["Frankfurt", "Berlin", "Bern"];
```

Beispiel: `array_kurzschreibweise.php`

Entsprechend dem Beispiel `array_assoz.php` wird in diesem Beispiel genau die gleiche Array-Variable definiert:

```
<?php
① $hauptstaedte = ["Schweiz" => "Bern",
                    "Frankreich" => "Paris",
                    "Deutschland" => "Berlin",
                    "Spanien" => "Madrid"];
② echo "<p>Die Hauptstadt von Spanien ist " .
                    $hauptstaedte["Spanien"] . "</p>";
?>
```

- ① Das assoziative Array wird über die eckigen Klammern `[]` und die Zuweisung der Schlüssel sowie der Array-Inhalte angelegt. Auch hier dienen Zeilenumbrüche und Einrückungen lediglich der Übersicht des PHP-Codes.
- ② Hier wird das Element durch die direkte Angabe des Schlüssels angesprochen. Die Bezeichnung des Schlüssels wird als Zeichenkette in den eckigen Klammern und in Anführungszeichen hinter dem Array-Variablenname angegeben.

5.5 Mit eindimensionalen Arrays arbeiten

Arrays ändern

Um einen Wert innerhalb einer Array-Variablen zu ändern, geben Sie bei der Wertzuweisung bei der Array-Variablen den entsprechenden Index bzw. Schlüssel des zu ändernden Wertes an und weisen Sie den neuen Wert zu.

```
Indiziert: $staedte[1] = "Paris";
Assoziativ: $staedte["Frankreich"] = "Paris";
```

Sofern bereits ein Element mit dem angegebenen Index bzw. Schlüssel vorhanden ist, wird der alte Wert durch den neuen ersetzt. Gibt es mit dem Schlüssel noch keinen Eintrag in dem Array, fügen Sie mit diesen Anweisungen neue Einträge im Array hinzu.

Array-Werte mit `unset()` zurücksetzen

Mit der Funktion `unset()` können Sie einen Eintrag aus einem Array löschen. In die runden Klammern geben Sie die genaue Bezeichnung des Index der Array-Variablen ein:

```
Indiziert: unset($staedte[1]);
Assoziativ: unset($staedte["Frankreich"]);
```

Es wäre auch denkbar, die Schreibweise `$staedte[1] = false` oder `$staedte[1] = ""` einzusetzen. Allerdings nehmen Sie mit dieser Syntax lediglich eine neue Wertzuweisung vor, der Wert von `$staedte[1]` wäre dann `false` bzw. leer, der Schlüssel `1` bleibt jedoch im Array erhalten. `unset()` hingegen löscht nicht nur den Wert, sondern die vollständige Schlüssel-Wert-Kombination.

Arrays erweitern

Arrays können einfach erweitert werden, indem Sie einen neuen Wert zuweisen – entweder mit oder ohne Schlüssel. In beiden Fällen wird ein neues Array-Element an das Ende des Arrays angefügt.

Syntax und Bedeutung

```
Indiziert: $feld[] = Wert;
Assoziativ: $feld["key"] = Wert;
```

- ✓ Um einem Array einen weiteren Wert (oder mehrere Werte) hinzuzufügen, brauchen Sie bei der Wertzuweisung in einem indizierten Array keinen Index anzugeben. Das Array wird am Ende um den Wert (bzw. die Werte) ergänzt und der Index automatisch erhöht.
- ✓ In einem assoziativen Array müssen Sie einen Schlüssel angeben, ansonsten verwendet PHP bei diesem Element des Arrays automatisch einen numerischen Indexwert.
- ✓ Sollte die Array-Variable noch nicht bestehen, erstellt PHP sie automatisch, sobald Sie der Variablen einen Wert zuweisen.

- ✓ Sollte der Schlüssel bereits vergeben sein, fügen Sie kein Element hinzu, sondern überschreiben den bestehenden Eintrag.
- ✓ Numerisch indizierte und assoziative Arrays schließen sich gegenseitig nicht aus. Fügen Sie einem assoziativen Array einen Wert ohne konkreten Schlüssel hinzu, fügt PHP automatisch einen numerischen Index hinzu. Ebenso können Sie einem numerisch indizierten Array ein Schlüssel-Werte-Paar hinzufügen. In beiden Fällen entsteht eine Mischform aus beiden Array-Typen, die sowohl numerische Indexwerte als auch selbst definierte Schlüssel hat. Der Einsatz von Arrays mit indizierten und assoziativen Schlüsseln gehört allerdings nicht zu einem guten Programmierstil.

Beispiel: *array_indiz.php* (Auszug)

```
$staedte = array("Frankfurt", "Berlin", "Bern");
$staedte[] = "Graz";
$staedte[] = "Rom";
```

ergibt:

Index	0	1	2	3	4
Wert	Frankfurt	Berlin	Bern	Graz	Rom

Indexwerte manuell vergeben

Bei assoziativen Arrays werden Schlüsselwerte stets vom Benutzer eingegeben und mit dem Wert durch die Zeichenfolge **=** zugewiesen. Bei indizierten Arrays können Sie ebenfalls Index-einträge selbst vergeben. Sie sind nicht an die automatisch vergebenen Indexwerte gebunden.

Beispiel: *array_indiz_manuell.php*

Hier wurde die Zuweisung der Städtenamen mit der `array()`-Anweisung realisiert.

```
<?php
① $staedte = array(1 => "Frankfurt", "Berlin", "Bern");
② $staedte[34] = "Graz";
③ $staedte[] = "Rom";
④ $staedte[5] = "Hamburg";
⑤ $staedte[] = "Köln";
⑥ echo "<pre>";
    print_r($staedte);
    echo "</pre>";
?>
```

- ① Das Array `$staedte` wird über das Schlüsselwort `array()` angelegt und mit Werten gefüllt. Sie weisen einen Indexwert zu, indem Sie den Indexwert zusammen mit den Zeichen `=>` vor den gewünschten Eintrag schreiben. In diesem Fall wird dem Eintrag `Frankfurt` der Index `1` zugewiesen. Da es sich um den ersten Eintrag im Array handelt, wird der Indexwert `0` nicht vergeben. Bei einer automatischen Indizierung wird die Nummerierung einfach nach dem höchsten vorhandenen Wert fortgesetzt, also keine eventuell vorhandenen Lücken aufgefüllt. `Berlin` erhält automatisch den Index `2`, `Bern` den Index `3`.
- ② Der Eintrag `Graz` erhält manuell den Indexwert `34`.
- ③ Der Indexwert für den Eintrag `Rom` wird automatisch vergeben. PHP sucht den größten Indexwert im Array (`34`) und setzt die Index-Nummerierung mit dem nächsthöheren Wert `35` fort.
- ④ Hier wird ein Index vergeben, der kleiner ist als der höchste Index im Array. Dieser Index wird für den Eintrag im Array verwendet. In der Ausgabe ⑥ können Sie allerdings beobachten, dass der Eintrag **hinter den existierenden Elementen in das Array eingefügt**, und **nicht** mit dem Index in das Array einsortiert wird. Für die Sortierung der Array-Elemente stehen unterschiedliche Array-Funktionen zur Verfügung (weiter unten).
- ⑤ Ein weiterer Eintrag ohne Index wird dem Array hinzugefügt. In der Ausgabe sehen Sie, dass der höchste Index `35` als Basis zur weiteren Nummerierung verwendet wird, und nicht der Index `5` des davor hinzugefügten Elements.
- ⑥ Die Funktion `print_r()` gibt den Inhalt der Array-Variablen `$staedte` mit allen Einträgen und dazugehörigen Indexwerten aus. Die Array-Variable wird in runden Klammern in der Funktion `print_r($staedte)` geschrieben. So können Sie bei Arrays überprüfen, welche Indexwerte den Einträgen zugeordnet sind. Das HTML-Tag `<pre>` sorgt dafür, dass das Array zeilenweise ausgegeben wird.

```
Array
(
    [1] => Frankfurt
    [2] => Berlin
    [3] => Bern
    [34] => Graz
    [35] => Rom
    [5] => Hamburg
    [36] => Köln
)
```

*Ausgabe des Array-Inhaltes
(Beispieldatei „array_indiz_manuell.php“)*

5.6 Daten aus eindimensionalen Arrays extrahieren

Arrays mit der `foreach()`-Schleife durchlaufen

Mit einer `foreach()`-Schleife sind Sie in der Lage, die einzelnen Werte der Array-Elemente auszulesen. Dabei wird jedes Element in einer neuen Variablen zwischengespeichert. Ein Vorteil der `foreach()`-Schleife ist, dass Sie weder die Schlüssel des Arrays noch die Anzahl der Array-Einträge kennen müssen. Die Schleife durchläuft das Array bis zum letzten Eintrag.

```

① foreach($feld as $wert) {
    Anweisungsblock;
}

② foreach($feld as $index => $wert) {
    Anweisungsblock;
}

```

- ✓ `foreach()` erwartet die Angabe des assoziativen oder indizierten Arrays, dessen Elemente durchlaufen werden sollen.
- ✓ In der ersten Variante ① werden mithilfe von `foreach` bei jedem Schleifendurchlauf der Variablen `$wert` nacheinander die Werte der Array-Elemente zugewiesen. Diese Variante wird häufig für indizierte Arrays eingesetzt, da hier die automatisch vergebenen Indexwerte oft ohne Bedeutung sind und nicht weiterverarbeitet oder ausgegeben werden sollen.
- ✓ Mithilfe der zweiten Variante ② können Sie zusätzlich den Index jedes Array-Elements auslesen. Bei jedem Durchlauf wird der Schlüssel des Array-Eintrags der Variablen `$index` zugewiesen, der dazugehörige Array-Wert der Variablen `$wert`. Diese Variante wird häufig bei assoziativen Arrays verwendet, wenn der Schlüssel für die weitere Verarbeitung benötigt oder ausgewertet wird. Aber auch bei indizierten Arrays kann diese Variante eingesetzt werden.

Beispiel: *foreach.php*

Verschiedene Länder der Welt und deren Hauptstädte werden in einem assoziativen Array gespeichert. Zur Ausgabe sollen das Land (= *Schlüssel*) und die dazugehörige Hauptstadt (= *Wert*) in einer Tabelle angezeigt werden:

```

<?php

① $hauptstaedte = array("Schweiz" => "Bern",
                           "Frankreich" => "Paris",
                           "Deutschland" => "Berlin");

② $hauptstaedte["Polen"] = "Warschau";
② $hauptstaedte["Italien"] = "Rom";
② $hauptstaedte["Spanien"] = "Madrid";
③ echo "<table border='1'>";
③ echo "<tr><th>Land</th><th>Hauptstadt</th></tr>";

④ foreach ($hauptstaedte as $land => $stadt) {
⑤     echo "<tr><td>$land</td><td>$stadt</td></tr>";
⑥ }

⑥ echo "</table>";
?>

```

Beispieldatei „foreach.php“

- ① Das assoziative Array `$hauptstaedte` wird über das Schlüsselwort `array()` erstellt und mit Werten gefüllt.
- ② Dem assoziativen Array `$hauptstaedte` werden weitere Elemente hinzugefügt.
- ③ Der HTML-Tabellenkopf und die Kopfzeile der Tabelle werden **vor** der `foreach`-Schleife erstellt, der Tabellenfuß **nach** der Schleife ⑥.
- ④ Mithilfe einer `foreach`-Schleife können Sie auf jedes Element des Arrays zugreifen. Die Schleife wird so lange ausgeführt, bis alle Elemente durchlaufen wurden. In jedem Schleifendurchlauf wird der Variablen `$land` der jeweilige Schlüssel und der Variablen `$stadt` der Wert des Arrays an der durch den Schlüssel festgelegten Position zugeordnet.
- ⑤ Die Werte werden in die Tabelle eingetragen und über den Ausgabebefehl `echo` am Bildschirm ausgegeben.

The screenshot shows a browser window titled "foreach" with the URL "localhost/herdt/Beispieldat...". The table has two columns: "Land" and "Hauptstadt". The data rows are:

Land	Hauptstadt
Schweiz	Bern
Frankreich	Paris
Deutschland	Berlin
Polen	Warschau
Italien	Rom
Spanien	Madrid

Ausgabe von Werten eines assoziativen Arrays
(Beispieldatei „foreach.php“)

Fehler beim Einsatz von `foreach()`-Schleifen vorbeugen

`foreach()`-Schleifen benötigen immer ein Array oder ein Objekt, damit diese Variablen durchlaufen werden können. Ist die Variable nicht gesetzt oder kein Array bzw. Objekt, meldet PHP eine Fehlermeldung vom Typ *warning*.

Im Programmieralltag arbeiten Sie häufig mit Arrays, die Sie nicht selbst definiert haben, sondern die z. B. als Ergebnis einer Datenbankabfrage bereitstehen. Hier können Sie nicht zwangsläufig davon ausgehen, dass eine Variable immer ein Array ist oder dass eine Variable überhaupt deklariert wurde. Um Fehlermeldungen zu vermeiden, egal ob als Ausgabe auf dem Bildschirm oder als Eintrag im Logfile, sichern Sie den Einsatz einer `foreach()`-Schleife ab.

Variablen mit `isset()` und `is_iterable()` prüfen

```
<?php
① // $werte = ['Köln', 'München', 'Hamburg'];
    // $werte = 'Köln';
② if (isset($werte) && is_iterable($werte)) {
    echo "<table>";
    foreach ($werte as $key => $value) {
        echo "<tr><td>$key</td><td>$value</td></tr>\n";
    }
    echo "</table>";
} else {
    echo "<p>Keine Werte ermittelt.</p>";
}
?>
```

Beispieldatei „is_iterable.php“

- ① Hier sind zwar Variablen angegeben, durch die Kommentarzeichen davor ist die Variable `$wert` jedoch weder deklariert noch initialisiert.

- ② Bevor Sie in die `foreach()`-Schleife gehen, wird die Variable geprüft: Mit `isset()` prüfen Sie, ob die Variable überhaupt gesetzt ist. Im Fehlerfall springt die `if`-Abfrage direkt in den `else`-Zweig. Mit `is_iterable()` prüfen Sie, ob die Variable `iterable`, also „durchlaufbar“ ist. Falls die Variable z. B. den Datentyp `string` oder `int` hat, ist die Prüfung ohne Erfolg, die `if`-Abfrage springt auch dann in den `else`-Zweig.
- ③ Falls überhaupt keine oder keine `iterable`-Variable gefunden wurde, wird eine entsprechende Meldung angezeigt. Nehmen Sie in der Zeile ① die Kommentarzeichen weg und schauen Sie, wie sich das PHP-Skript verhält.

Ausgabe der Beispieldatei „is_iterable.php“

5.7 Mehrdimensionale indizierte Arrays erstellen

Grundlagen zu mehrdimensionalen Arrays

Im folgenden Beispiel sehen Sie mehrere Angaben (*Vorname*, *Nationalität* und *Alter*), die jeweils einen Eintrag in einer Array-Variablen bilden. Damit ist dieses Array ein mehrdimensionales Array. Bei jeder der Angaben sind Mehrfachwerte möglich. Mehrdimensionale Arrays sind verschachtelte Arrays, man spricht hier von äußeren und inneren Arrays.

Mehrdimensionale Arrays können sowohl numerisch indiziert als auch assoziativ sein. Eine Mischform aus numerisch indizierten und assoziativen Arrays ist ebenfalls möglich.

Index 1	Index 2	Wert
0	0	Oliver
	1	spanisch
	2	37 Jahre
1	0	Maria
	1	deutsch
	2	23 Jahre
2	0	Oliver
	1	englisch
	2	46 Jahre

In dieser Tabelle wird bereits die Arbeitsweise mehrdimensionaler Arrays ersichtlich. Um beispielsweise an die Informationen von *Maria* zu gelangen, suchen Sie die Zeile mit der Angabe *Maria* und lesen die einzelnen Werte dieser Zeile aus: *Maria*, *deutsch*, *23 Jahre*.

Mit mehrdimensionalen indizierten Arrays arbeiten

Ein mehrdimensionales indiziertes Array hat statt **eines** Indexes – in Abhängigkeit von der Anzahl der verschachtelungen – **mehrere** Indizes.

Auf eine bestimmte Angabe in einem mehrdimensionalen indizierten Array zugreifen

```
$person = array(
    array("Oliver",
          "Spanisch",
          "37 Jahre"
    ),
    array("Maria",
          "Deutsch",
          "23 Jahre"
    ),
    array("Oliver",
          "Englisch",
          "46 Jahre"
    )
);

echo $person[1][0] . " ist " . $person[1][2] . " alt und spricht " .
    $person[1][1] . "<hr>";
```

Ausschnitt aus der Beispieldatei „mehrdimensional.php“

Um auf einzelne Elemente in einem mehrdimensionalen Array zuzugreifen, verwenden Sie die eckigen Klammern **[]**. Im ersten Klammerpaar verwenden Sie den Index des äußeren Arrays, im zweiten Klammerpaar den Index des inneren Arrays.



Ausgabe des oben gezeigten Ausschnitts aus der Beispieldatei „mehrdimensional.php“

Ein mehrdimensionales indiziertes Array erweitern

```
$person[3][0] = "Johanna";
$person[3][1] = "schwedisch";
$person[3][2] = "19 Jahre";
```

oder alternativ:

```
$person[] = array("Johanna", "schwedisch", "19 Jahre");
```

Beides Ausschnitte aus der Beispieldatei „mehrdimensional.php“

5.8 Mit mehrdimensionalen assoziativen Arrays arbeiten

Syntax eines mehrdimensionalen assoziativen Arrays

Um ein mehrdimensionales Array anzulegen, wird jedem Schlüssel ein weiteres Array mit Schlüssel-Wert-Paaren übergeben. Dabei wird jedem Schlüssel auf erster Ebene ein weiteres Array zugewiesen. In diesen verschachtelten Arrays werden dann jeweils mehrere Schlüssel mit Werten angegeben.

Beispiel: *mehrdimensional.php*

```
$land = array(
    "Spanien" => array("Hauptstadt" => "Madrid",
                         "Sprache" => "Spanisch",
                         "Waehrung" => "Euro",
                         "Flaeche" => "504645 qkm"
    ),
    "England" => array("Hauptstadt" => "London",
                         "Sprache" => "Englisch",
                         "Waehrung" => "Pfund Sterling",
                         "Flaeche" => "130395 qkm"
    ),
    "Portugal" => array("Hauptstadt" => "Lissabon",
                         "Sprache" => "Portugiesisch",
                         "Waehrung" => "Euro",
                         "Flaeche" => "92345 qkm"
    )
);
```

In diesem Array ist jeder Eintrag mit der entsprechenden Länderinformation über die aussagekräftigen Schlüssel Spanien, England oder Portugal direkt ansprechbar.

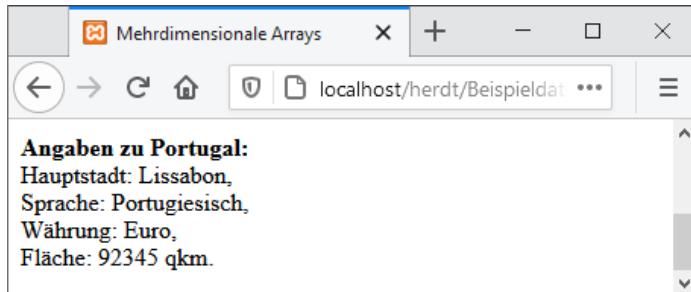
Auf ein bestimmtes Element in einem mehrdimensionalen assoziativen Array zugreifen

Bei numerisch indizierten Arrays greifen Sie mit dem numerischen Index auf die einzelnen Array-Einträge zu. Wenn Sie ein bestimmtes Array-Element in einem assoziativen Array direkt ansprechen wollen, müssen Sie die Elemente über den Schlüssel ansprechen. Mit dem ersten Schlüssel (hier Portugal) erhalten Sie das verschachtelte Array mit dem entsprechenden Index, mit dem zweiten Schlüssel (hier Hauptstadt usw.) ermitteln Sie dann den zugehörigen Wert.

Die Schreibweise sieht folgendermaßen aus: Sie geben den Variablenamen des Arrays an, dahinter zwei doppelte Pärchen von eckigen Klammern **[]**. Das erste Pärchen nimmt den ersten Schlüssel auf, das zweite Pärchen dann den Schlüssel des inneren Arrays, also `$wert ["key-1"] ["key-2"]`. Auch hier verwenden Sie Anführungszeichen für Zeichenketten-Schlüssel.

```
// Ausgabe der Angaben von Portugal
echo "<p><strong>Angaben zu Portugal:</strong><br>";
echo "Hauptstadt: " . $land["Portugal"]["Hauptstadt"] . ",<br>";
echo "Sprache: " . $land["Portugal"]["Sprache"] . ",<br>";
echo "Währung: " . $land["Portugal"]["Waehrung"] . ",<br>";
echo "Fläche: " . $land["Portugal"]["Flaeche"] . ".</p>";
```

Ausschnitt aus Beispieldatei „mehrdimensional.php“



Ausgabe des oben gezeigten Ausschnitts aus der Beispieldatei „mehrdimensional.php“

Ein mehrdimensionales assoziatives Array erweitern

```
$land["Ungarn"]["Hauptstadt"] = "Budapest";
$land["Ungarn"]["Sprache"] = "Ungarisch";
$land["Ungarn"]["Waehrung"] = "Forint";
$land["Ungarn"]["Flaeche"] = "93036 qkm";
```

oder alternativ:

```
$land["Ungarn"] = array("Hauptstadt" => "Budapest",
                        "Sprache" => "Ungarisch",
                        "Waehrung" => "Forint",
                        "Flaeche" => "93036 qkm");
```

Ausschnitt aus Beispieldatei „mehrdimensional.php“

5.9 Daten aus mehrdimensionalen Arrays extrahieren

Bei der Arbeit mit mehrdimensionalen Arrays kommt es häufig vor, dass Sie die kompletten Daten eines inneren Arrays benötigen. Mit einer `foreach`-Schleife können Sie alle Werte einer Array-Variablen auslesen. Bei der Verwendung einer `foreach`-Schleife müssen Sie keinen Index oder Schlüssel kennen. Bei jedem Schleifendurchlauf wird das aktuelle Element in einer von Ihnen angegebenen Variablen zwischengespeichert. Hierbei handelt es sich auch um eine Array-Variable mit dem aktuellen „Datensatz“.

Syntax

```
foreach($feld as $wert);
```

Bei jedem einzelnen Schleifendurchlauf wird das verschachtelte Array in der Variablen `$wert` gespeichert. Um nun die einzelnen Werte dieses Arrays auszulesen, wird eine noch nicht vorgestellte PHP-Funktion verwendet: die `list()`-Funktion.

Syntax

```
list($variable1, $variable2, ...) = $wert;
```

- ✓ `list()` dient dazu, die einzelnen Werte des inneren Arrays aufzunehmen und sie den Variablen zuzuweisen, die als Parameter in `list()` angegeben sind.
- ✓ In den runden Klammern der Funktion `list()` werden die Variablennamen angegeben, in denen die einzelnen Array-Werte gespeichert werden sollen. Diese können frei gewählt werden.
- ✓ Über dem Zuweisungsoperator `=` wird der Funktion `list()` die Array-Variable zugewiesen, im Beispiel `$wert`. Dies wiederholt sich beim `foreach()` bei jedem einzelnen Schleifendurchlauf.
- ✓ Nun werden die Werte der in `list()` angegebenen Variablen mit den einzelnen Einträgen des Arrays versehen. Die erste Variable erhält den Wert des ersten Eintrags aus dem Array `$wert`, die zweite Variable den des zweiten Eintrags aus dem Array usw.
- ✓ `list()` benötigt numerisch indizierte Arrays, die mit dem Schlüssel 0 beginnen und die nicht mit abweichenden Schlüsseln versehen worden sind.

! Die Funktionalität von `list()` hat sich mit der PHP-Version 7.0 verändert. In Kombination mit der Array-Kurzschreibweise `[]` werden Werte nicht mehr umgekehrt, sondern in angegebener Reihenfolge zugewiesen. Die Parameter innerhalb der `list()`-Funktion dürfen nicht mehr leer gelassen werden und `list()` kann nicht mehr auf Zeichenketten angewendet werden. Ältere PHP-Skripte können, falls Sie auf einem Webserver mit PHP 7.0 ausgeführt werden, zu Fehlern führen oder falsche Resultate liefern.

Beispiel: *array_mehrdimensional_list.php*

Verschiedene Angaben zu Ländern werden in einem mehrdimensionalen Array gespeichert. Zur Ausgabe sollen der Name des Landes, die Hauptstadt, die Sprache und die Landeswährung in separaten Variablen abgelegt werden.

```


| Land | Hauptstadt | Sprache | Währung |
|------|------------|---------|---------|
|------|------------|---------|---------|


```

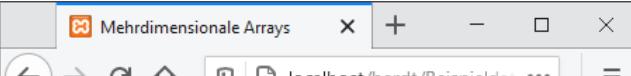
```

②  foreach ($staedte as $key => $ausgabe) {
③      list($hauptstadt, $sprache, $waehrung) = $ausgabe;
④      echo "<tr>";
④      echo "<td>" . $key . "</td>";
④      echo "<td>" . $hauptstadt . "</td>";
④      echo "<td>" . $sprache . "</td>";
④      echo "<td>" . $waehrung . "</td>";
④      echo "</tr>";
}
?>
</table>

```

- ① Das assoziative Array `$staedte` wird neu definiert und mit Werten gefüllt. Die Daten enthalten Informationen zur Hauptstadt eines Landes, zur Sprache und zur im Land verwendeten Währung. Der Name des Landes selbst wird als eindeutiger Schlüssel verwendet. Die verschachtelten Arrays selbst sind numerisch indizierte Arrays. Nur so kann die Funktion `list()` einwandfrei arbeiten.
- ② Mithilfe einer `foreach`-Schleife durchlaufen Sie das Array, bis das letzte Array-Element erreicht ist. Die `foreach()`-Schleife speichert dabei in jedem Durchlauf das verschachtelte Array in der Variablen `$ausgabe`. Im ersten Schleifendurchlauf enthält die Variable `$ausgabe` nur das Werte-Array des zuerst definierten Landes Japan (Tokio, Japanisch, Yen), im zweiten Schleifendurchlauf die Werte der Niederlande etc.
- ③ Über den Befehl `list()` werden die Daten zum aktuell abgerufenen Land (Hauptstadt, Sprache und Währung) in die entsprechenden Variablen `$hauptstadt`, `$sprache` und `$waehrung` überführt.
- ④ Über den Ausgabebefehl `echo` werden die Werte der Variablen am Bildschirm ausgegeben. Bei der Variablen `$key` handelt es sich um den Schlüssel (Landesnamen), der in ② durch die `foreach()`-Schleife zugewiesen wurde.

Hinweis: Die HTML-Attribute `border="1"` und `width="125"` in diesem Beispiel lassen die Testausgabe etwas geordneter erscheinen. Im modernen Webdesign würde man das Layout auf jeden Fall über CSS steuern.



The screenshot shows a web browser window with the title "Mehrdimensionale Arrays". The address bar displays "localhost/herdt/Beispieldat...". The main content area shows a table with four columns: "Land", "Hauptstadt", "Sprache", and "Währung". The table contains 10 rows of data, each representing a country with its capital, language, and currency.

Land	Hauptstadt	Sprache	Währung
Japan	Tokio	Japanisch	Yen
Niederlande	Amsterdam	Niederländisch	Euro
Polen	Warschau	Polnisch	Złoty
Indien	Neu Delhi	Indisch	Rupie
Island	Reykjavik	Isländisch	Krone
Italien	Rom	Italienisch	Euro
Frankreich	Paris	Französisch	Euro
Spanien	Madrid	Spanisch	Euro
England	London	Englisch	Pfund Sterling

Ausgabe von Werten eines mehrdimensionalen assoziativen Arrays
(Beispieldatei „array_mehrdimensional_list“)

5.10 Den passenden Array-Typ verwenden

Der passende Array-Typ hängt von den Daten ab, die Sie in der Array-Variablen speichern wollen. Die folgende Tabelle soll Ihnen bei der Auswahl des Array-Variablentyps helfen:

Daten	Anzahl an Informationen pro Eintrag	Bevorzugter Array-Typ
Einfache Liste (z. B. Namen, Länder, Automarken etc.)	1	✓ eindimensional/indiziert
Wertepaare (z. B. Länder – Hauptstädte, Artikel – Preis etc.)	2	✓ eindimensional/assoziativ, sofern ein Wert eindeutig ist (= Schlüssel), sonst ✓ mehrdimensional/indiziert oder assoziativ
Mehrere evtl. mehrfach verschachtelte Werte (z. B. Mitarbeiter und ihre Stammdaten, Artikelkategorien – Artikel – Artikeldetails etc.)	> 2	✓ mehrdimensional/indiziert oder assoziativ

5.11 Weitere Informationen zu Arrays in PHP

Arrays begegnen Ihnen in PHP nicht nur, wenn Sie selbst Array-Variablen definieren. PHP arbeitet intern ebenfalls mit Arrays, und zwar bei folgenden Aktionen, die in den nächsten Kapiteln des Buches besprochen werden:

- ✓ Die in einem HTML-Formular übermittelten Daten speichert PHP in einer Array-Variablen.
- ✓ Wenn Sie mit Sessions arbeiten, werden die zur Session gehörigen Daten automatisch in einem Array abgelegt.
- ✓ Bei Abfragen aus Datenbanken befinden sich die Ergebnisse ebenfalls in einer Array-Variablen.

Weitere Array-Funktionen

PHP bietet über 80 Funktionen für den Umgang mit Arrays. Für nahezu jede Fragestellung liefert PHP eine passende Funktion. Folgende Aufgabengebiete werden abgedeckt:

- ✓ Auslesen
- ✓ Auswerten (u. a. Dopplungen finden)
- ✓ Sortieren (vorwärts, rückwärts, vorwärts nach Schlüssel, rückwärts nach Schlüssel)
- ✓ Suchen
- ✓ Teilen und Zusammensetzen

- ✓ Verändern (u. a. Füllen, Hinzufügen, Löschen)
- ✓ mehrere Arrays zusammenführen
- ✓ neues Array aus den Array-Schlüsseln generieren

Eine Auswahl wichtiger Array-Funktionen wird in der nachfolgenden Tabelle vorgestellt:

Array-Funktion	Beschreibung
<code>array_flip(\$feld)</code>	Im Array werden Indizes mit Werten vertauscht.
<code>array_key_exists(wert, \$feld)</code>	Prüfung, ob ein Schlüssel in einem Array vorhanden ist
<code>array_keys(\$feld)</code>	Liefert die Indizes des angegebenen Arrays zurück
<code>array_merge(\$feld1, \$feld2...)</code>	Fügt die Elemente mehrerer Arrays zu einem Array zusammen
<code>array_push(\$feld, werte)</code>	Das Array wird um den oder die angegebenen Werte am Ende des Arrays erweitert.
<code>array_search(wert, \$feld)</code>	Das Array wird nach dem angegebenen Wert durchsucht.
<code>array_sum(\$feld)</code>	Addiert die Werte des Arrays und liefert das Ergebnis zurück
<code>array_unique(\$feld)</code>	Es wird ein neues Array erstellt, aus dem doppelte Werte des angegebenen Arrays gelöscht wurden.
<code>array_values(\$feld)</code>	Alle Werte des Arrays werden zurückgeliefert.
<code>count(\$feld)</code>	Gibt die Anzahl der Elemente des Arrays zurück
<code>sort(\$feld)</code> <code>rsort(\$feld)</code>	Sortiert die Werte des angegebenen Arrays aufsteigend bzw. absteigend
<code>ksort(\$feld)</code> <code>krsort(\$feld)</code>	Sortiert das angegebene Array nach Schlüssel aufsteigend bzw. absteigend

Beispiele zu Array-Funktionen finden Sie im Abschnitt 9.8.

Einen Überblick über alle Array-Funktionen finden Sie auf der Webseite von php.net:

<https://www.php.net/manual/de/ref.array.php>

5.12 Übungen

Übung 1: Mit eindimensionalen Arrays arbeiten

Level		Zeit	ca. 10 min
Übungsinhalte	<ul style="list-style-type: none"> ✓ Arrays ✓ Array-Elemente hinzufügen ✓ Array-Elemente löschen ✓ Nutzen der <code>foreach()</code>-Schleife ✓ HTML-Ausgabe 		
Übungsdatei	--		
Ergebnisdatei	<i>kennzeichen.php</i>		

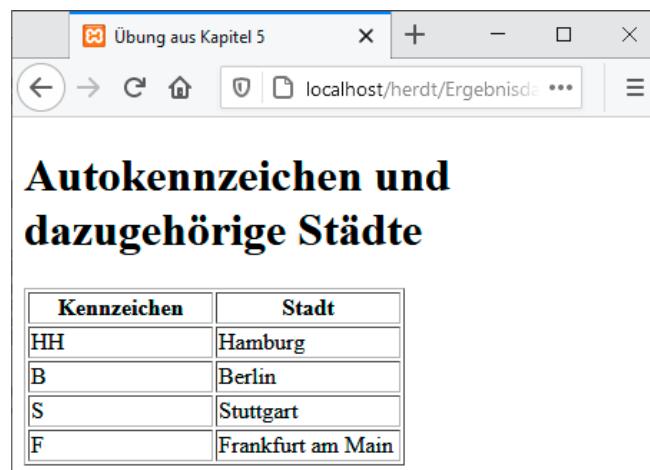
1. Erstellen Sie eine neue PHP-Datei unter dem Namen *kennzeichen.php*. Definieren Sie eine assoziative Array-Variable, die Sie mit der folgenden Liste von Autokennzeichen und den dazugehörigen Städten füllen:

HH	Hamburg
B	Berlin
S	Stuttgart

2. Ergänzen Sie die Liste nach der Erstellung mit den Elementen:

F	Frankfurt
HB	Bremen

3. Löschen Sie mithilfe des Befehls `unset()` das Element *Bremen* aus dem Array und definieren Sie den Eintrag *Frankfurt* neu – er soll nun *Frankfurt am Main* heißen.
4. Listen Sie mithilfe von `foreach` alle Elemente der Tabelle auf und vergeben Sie zusätzlich Überschriften für die Tabelle.



Kennzeichen	Stadt
HH	Hamburg
B	Berlin
S	Stuttgart
F	Frankfurt am Main

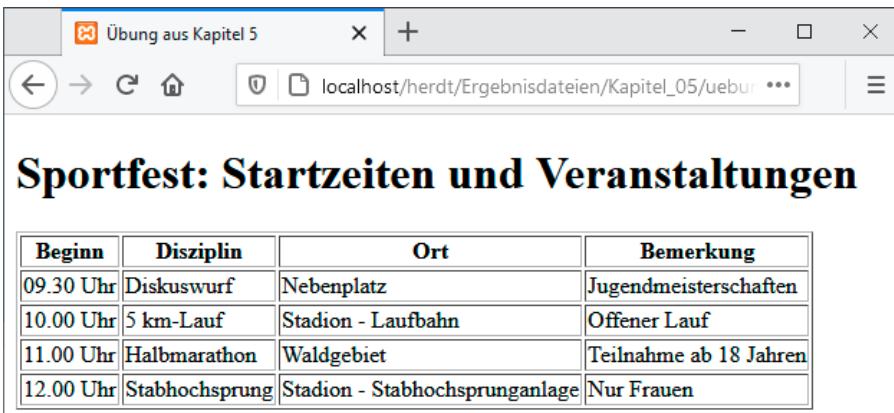
Beispiel-Ergebnisdatei „kennzeichen.php“

Übung 2: Mit mehrdimensionalen Arrays arbeiten

Level		Zeit	ca. 10 min
Übungsinhalte	<ul style="list-style-type: none"> ✓ Mehrdimensionale Arrays ✓ Array-Elemente hinzufügen ✓ Array-Elemente löschen ✓ Nutzen der <code>foreach()</code>-Schleife ✓ HTML-Ausgabe 		
Übungsdatei	--		
Ergebnisdatei	<i>uebung_mehrdimensional.php</i>		

1. Erstellen Sie eine neue Datei unter dem Namen *uebung_mehrdimensional.php*. Erstellen Sie ein mehrdimensionales indiziertes Array mit folgenden Inhalten und geben Sie die Daten anschließend in Tabellenform (mit Überschrift) auf dem Bildschirm aus:

<i>Beginn</i>	<i>Disziplin</i>	<i>Ort</i>	<i>Bemerkung</i>
09:30 Uhr	Diskuswurf	Nebenplatz	Jugendmeisterschaften
10:00 Uhr	5-km-Lauf	Stadion - Laufbahn	Offener Lauf
11:00 Uhr	Halbmarathon	Waldgebiet	Teilnahme ab 18 Jahren
12:00 Uhr	Stabhochsprung	Stadion - Stabhochsprunganlage	Nur Frauen



Sportfest: Startzeiten und Veranstaltungen

<i>Beginn</i>	<i>Disziplin</i>	<i>Ort</i>	<i>Bemerkung</i>
09.30 Uhr	Diskuswurf	Nebenplatz	Jugendmeisterschaften
10.00 Uhr	5 km-Lauf	Stadion - Laufbahn	Offener Lauf
11.00 Uhr	Halbmarathon	Waldgebiet	Teilnahme ab 18 Jahren
12.00 Uhr	Stabhochsprung	Stadion - Stabhochsprunganlage	Nur Frauen

Beispiel-Ergebnisdatei „uebung_mehrdimensional.php“

6

Mit Formularen arbeiten



Beispieldateien: Dateien im Ordner *Kapitel_06*

6.1 Interaktion mit PHP

Formulare einsetzen

Formulare sind die Schnittstelle zwischen Benutzer und PHP. Immer, wenn ein Benutzer interagieren möchte, wird er Daten in ein Formular eintragen. Über Formulare können Daten an den Webserver gesendet werden. PHP kann die Eingaben auswerten und entsprechend die nächste Webseite dynamisch und individuell aufbauen. Unabhängig davon, ob Sie ein Login-, Bestell- oder Kontaktformular verwenden, es handelt sich immer um die Übertragung von Nutzereingaben und deren anschließende Verarbeitung durch PHP.

Formularauswertung mit PHP

Mit PHP können Sie interaktive Webseiten erstellen, bei denen Benutzereingaben aus Formularen durch PHP ausgewertet werden.

Nachdem Sie ein Formular aufgerufen haben, können Sie Daten in das Formular eingeben. Wenn das Formular abgesendet wird, werden die eingegebenen Daten an den Webserver übertragen und stehen PHP für die Verarbeitung zur Verfügung. Abhängig von den Eingaben kann PHP individuelle HTML-Ergebnisseiten generieren und sendet diese an Ihren Browser zurück.

Methoden der Datenübertragung

Das **Hypertext Transfer Protocol (HTTP)**, englisch für Hypertext-Übertragungsprotokoll) ist ein Kommunikationsprotokoll, welches für die Übertragung von Daten zwischen Browser und Webserver verantwortlich ist bzw. die Regeln für den Austausch von Daten festlegt.

HTTP sieht zwei Methoden vor, um Daten an einen Webserver zu senden: **POST** und **GET**. Diese beiden Methoden unterscheiden sich u. a. dadurch, wie Formulardaten an den Server übertragen werden oder in der Menge der Daten, die übertragen werden können.

Auf welche Weise Eingaben eines HTML-Formulars an den Webserver gesendet werden, definieren Sie selbst. Das HTML-Element `form` sieht das Attribut `method` vor. Bei der Angabe von `<form method="POST">` legen Sie fest, dass Formulardaten per POST-Methode übertragen werden, entsprechend `<form method="GET">` für die GET-Methode. Die Werte POST bzw. GET sind **nicht case-sensitiv**, `<form method="post">` und `<form method="get">` sind ebenfalls gültige Schreibweisen. Lassen Sie das `method`-Attribut leer oder weg (absichtlich oder versehentlich), verwenden Browser den Standardwert GET für `method`, Daten werden dann per GET-Methode übertragen.

Die GET-Methode

Die Eingaben werden nach dem Absenden des Formulars an die URL angehängt. Der URL folgt zuerst das Fragezeichen **?** als Trennung von der eigentlichen URL und den Eingaben des Formulars. Hinter dem **?** folgen dann alle Daten aus dem Formular. Diese bestehen aus dem Schlüssel (dem Wert des Formularelement-Attributs `name`) und der eigentlichen Eingabe des Nutzers, getrennt durch ein Gleichheitszeichen **=** (z. B. `vorname=Maxi`). Die einzelnen Schlüssel-Wert-Paare werden jeweils durch ein „kaufmännisches Und“ **&** voneinander getrennt.

Eine URL nach Absenden eines GET-Formulars könnte wie folgt aussehen:

`http://localhost/antwort.php?vorname=Maxi&nachname=Müller&kennwort=123abc`

Eine solche URL könnte auch als Link auf einer Webseite verwendet werden. Links verwenden also die HTTP-GET-Methode zur Datenübertragung. Der Webserver selbst kann nicht unterscheiden, ob die Werte aus einem Formular mit der GET-Methode oder von einem Link mit Parametern stammen.

Merkmale von GET

- ✓ Angabe von `method="GET"` im einleitenden HTML-Tag `<form>`
- ✓ Formulardaten werden sichtbar und unverschlüsselt in der URL übermittelt.
- ✓ Daten sind in der Adresszeile des Browsers veränderbar, ohne dass das Formular erneut ausgefüllt bzw. abgesendet werden muss.
- ✓ Der Aufruf des Skripts mit Angabe der Daten kann als Lesezeichen gespeichert werden.
- ✓ Die Übertragungsmenge gegenüber POST ist wesentlich geringer. Die Begrenzung der Datenmenge wird durch den Browser bestimmt und kann von Browser zu Browser variieren.

Die POST-Methode

Bei der POST-Methode werden Formulardaten als Datenblock bzw. Datenstrom nach dem HTTP-Header gesendet. In der Adresszeile des Browsers werden keine Werte angezeigt, die Formulardaten werden für den Nutzer nicht sichtbar übertragen. Die Menge der Daten, die übertragen werden können, ist bei der POST-Methode wesentlich größer als bei der GET-Methode. Außerdem ermöglicht ausschließlich die POST-Methode den Upload von Dateien (`<input type="file">`), die Übertragung von Dateien ist mit der GET-Methode nicht möglich.

Merkmale von POST

- ✓ Angabe von `method="POST"` im einleitenden HTML-Tag `<form>`
- ✓ Daten können nicht in der Adresszeile des Browsers manipuliert werden.
- ✓ Formulardaten werden nicht in der Protokolldatei des Servers gespeichert.
- ✓ Formulardaten sind nicht im Verlauf des Browsers sichtbar.
- ✓ Die Datenmenge, die übertragen werden kann, ist wesentlich größer als bei der GET-Methode. Per Standard können 40 MB (8 MB vor PHP 8) übertragen werden, der Wert kann angepasst werden (in der PHP-Konfigurationsdatei `php.ini` über den Parameter `post_max_size`, vgl. Installationshinweise im Anhang).
- ✓ Upload von Dateien ist nur über die POST-Methode möglich (hierzu wird im `form`-Tag zusätzlich das Attribut `ENCTYPE="multipart/form-data"` benötigt).

Unterschiede zwischen GET und POST

Der wesentliche Unterschied liegt in der Übertragung in der URL im Gegensatz zu der als Datenstrom. Bei der Übertragung in der URL können Daten in der URL verändert werden, diese bietet die Möglichkeit der Manipulation. Die Möglichkeit, GET-URLs als Favorit zu speichern oder diese per E-Mail zu versenden, birgt die Gefahr, dass diskrete Eingaben wie z. B. Passwörter für andere Nutzer sichtbar werden.

`<input type="hidden">`-Felder (versteckte Formularfelder im HTML für interne Daten zur Weiterverarbeitung) werden bei der GET-Methode in der URL für den Nutzer direkt sichtbar. Allerdings sind `hidden`-Felder auch bei der POST-Methode im HTML-Quelltext auszulesen, was die Möglichkeit der Manipulation bietet.

Ein weiterer Unterschied ist das Verhalten der Browser: Wird eine Webseite nach dem Absenden eines GET-Formulars neu geladen (z. B. per `F5`-Taste), werden die Formulardaten erneut an den Webserver gesendet, hier kann es zur mehrfachen Verarbeitung der gleichen Daten kommen (was z. B. bei einer Bank-Überweisung fatal wäre).

Bei der POST-Methode erscheint im Browser ein Dialogfenster mit der Frage *Um diese Seite anzuzeigen, müssen die von Firefox gesendeten Daten erneut gesendet werden, wodurch alle zuvor durchgeführten Aktionen wiederholt werden (wie eine Suche oder eine Bestellungsaufgabe)* (hier im Firefox 86.0, Meldungen in anderen Browsern vergleichbar). Erst wenn Sie diese Frage bestätigen, wird das Formular nochmal versendet. So erkennen Sie bereits beim Reload einer Seite, welche HTTP-Methode im `form`-Tag definiert wurde.

Aufgrund der Sichtbarkeit der übertragenen Daten kann die POST-Methode als sicherer als die GET-Methode bewertet werden. Aus diesem Grund wird in diesem Buch vorwiegend die POST-Methode in Formularen verwendet.



Allerdings: Die Wahl der POST-Methode bietet nur eine kleinere Verbesserung der Sicherheit. Auch POST-Formulare können manipuliert werden. Ein Seiten-Restart kann dazu führen, dass gleiche Eingaben mehrfach an den Webserver gesendet werden. Diese Risiken müssen über eine durchdachte Logik im PHP abgefangen werden.

6.2 Formulare mit PHP auswerten

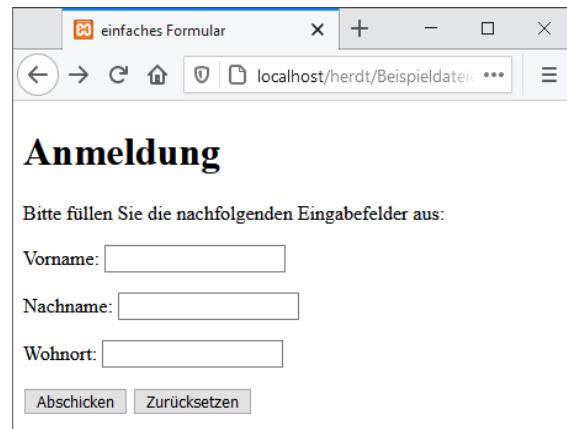
Formulardaten eingeben

Beispiel: *formular.html*

Erstellen Sie zunächst eine HTML-Datei, in der ein einfaches Formular mit Text-Eingabefeldern integriert ist. In diesem Formular soll der Nutzer seinen Vor- und Nachnamen sowie seinen Wohnort angeben. Im Formular sollen Eingaben in Textfelder (*input*-Elemente vom *type="text"*) erfolgen.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>einfaches Formular</title>
  </head>
  <body>
    <h1>Anmeldung</h1>
    <p>Bitte füllen Sie die nachfolgenden Eingabefelder aus:</p>
    ① <form action="formular_auswertung.php" method="POST">
      ② <p>Vorname: <input type="text" name="vorname"></p>
      <p>Nachname: <input type="text" name="nachname"></p>
      <p>Wohnort: <input type="text" name="ort"></p>
      ③ <p><input type="submit" value="Abschicken">
          <input type="reset" value="Zurücksetzen"></p>
    </form>
  </body>
</html>
```

- ① Im HTML wird das Formular über das *<form>*-Tag definiert. Das Skript, das die Formulardaten auswerten und verarbeiten soll, wird über das Attribut *action* festgelegt. In diesem Fall werden die Daten an das PHP-Skript *formular_auswertung.php* gesendet. Die POST-Methode zur Übermittlung der Daten wird mittels *method="POST"* festgelegt. Wichtig ist, dass alle Formularfelder innerhalb des öffnenden und schließenden *form*-Tags liegen. Auch die Schaltfläche *Abschicken* muss innerhalb der *form*-Tags stehen.
- ② Es folgen die einzelnen Texteingabefelder vom *input*-Typ *text*. Falls Sie das *type*-Attribut weg- oder leer lassen, verwenden alle Browser den Standardwert *text* für das HTML-Element *input* und stellen automatisch ein einziges Eingabefeld dar. Das HTML-Attribut *name* der Felder wird mit den Bezeichnern *vorname*, *nachname* und *ort* konfiguriert. Nach dem Absenden des Formulars sind die *name*-Attribute die Schlüssel des *\$_POST*- bzw. *\$_GET*-Arrays, welche für die weitere Verarbeitung zur Verfügung stehen.
- ③ Es folgen die Standardschaltflächen zum *Abschicken* und *Zurücksetzen* der Formulardaten. Die Schaltfläche *Zurücksetzen* ist optional und dient lediglich der leichteren Bedienung des Formulars durch den Benutzer, falls dieser seine Eingaben mit einem Klick löschen möchte.



Beispielformular „formular.html“

Alle Formularelemente, also einzeilige **Eingabefelder** (auch vom Typ `email`, `number`, `password` usw.), **Selectboxen**, **mehrzeilige Textfelder** sowie **Radiobuttons** und **Checkboxen** eines HTML-Formulars müssen durch das Attribut `name` **eindeutig** gekennzeichnet sein. Der Wert des `name`-Attributs des Formularfeldes wird später als Schlüssel im Array `$_POST` bzw. `$_GET` verwendet.

Formulardaten übertragen

Mit Klick auf die Schaltfläche *Abschicken* werden die eingegebenen Formulardaten an den Webserver gesendet. Im Formularattribut `action` geben Sie an, an welches PHP-Skript die Daten zur Verarbeitung gesendet werden sollen (*formular_auswertung.php*). Dieses Skript soll nun erstellt werden.

Formulardaten auswerten

Nach dem Absenden des Formulars stellt der Webserver die Übertragung von Formulardaten fest und stellt diese für das PHP-Skript je nach Übertragungsmethode in dem superglobalen Array `$_POST` bzw. `$_GET` zur Verfügung. Superglobal bedeutet, das Array ist an allen Stellen im PHP-Skript, z. B. auch innerhalb von eigenen PHP-Funktionen, verfügbar (mehr dazu im folgenden Kapitel). Dabei ist der Schlüssel des Arrays der Wert des HTML-Attributs `name`, der Wert selbst ist dann die Eingabe des Nutzers.

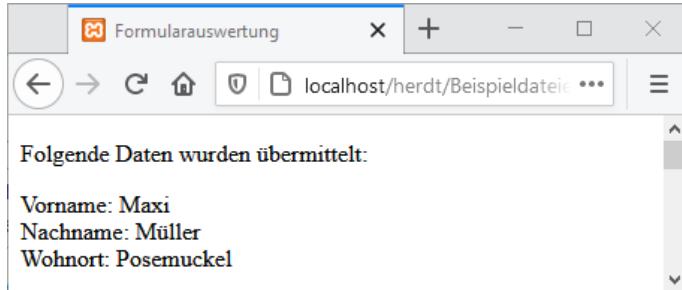
Darüber hinaus hält PHP die übertragenen Werte in dem von PHP definierten Array `$GLOBALS` bereit. Je nach Übertragungsmethode können Sie entweder auf die Arrays `$GLOBALS["_GET"]` bzw. `$GLOBALS["_POST"]` zugreifen. Zusätzlich können Sie die Array-Variable `$_REQUEST` verwenden. In diesem Array stehen die Formulardaten unabhängig von der Übertragungsmethode zur Verfügung. Allerdings sollten Sie immer das `$_POST`- bzw. `$_GET`-Array ansprechen. Das ist guter Stil und ein Aspekt sicherer Programmierung. Damit stellen Sie sicher, dass Formulardaten auch aus dem Formular stammen, das Sie selbst programmiert haben.

Die übermittelten Daten sprechen Sie über die Array-Variable und den Schlüssel an: `$_POST["Schlüssel"]`. Der Wert des HTML-input-Attributs `name` (z. B. `name="vorname"`) wird als Schlüssel der Array-Variablen `$_POST` verwendet, z. B. `$_POST["vorname"]`. Die Eingabe im Formular bildet den dazugehörigen Wert der Array-Variablen, z. B. `$_POST["vorname"] = "Maxi"`.

Beispiel: *formular_auswertung.php*

Sie erstellen ein PHP-Skript, das die Eingaben des HTML-Formulars ausliest und zur Kontrolle am Bildschirm ausgibt.

```
<?php
echo "<p>Folgende Daten wurden übermittelt:</p>";
echo "<p>Vorname: " . $_POST["vorname"] . "<br>";
echo "Nachname: " . $_POST["nachname"] . "<br>";
echo "Wohnort: " . $_POST["ort"] . "</p>";
?>
```



Ausgabe der Beispieldatei „formular_auswertung.php“

Übertragene Formulardaten anzeigen

PHP bietet unterschiedliche Funktionen, mit der Sie sich Arrays im Browser anzeigen lassen können. Diese können Sie auch für die Anzeige der übermittelten Formulardaten verwenden:

```
print_r(Variable);
var_dump(Variable 1[, Variable 2,...]);
```

Erweitern Sie das PHP-Skript um folgende Zeilen. Sowohl `print_r()` als auch `var_dump()` dienen der Darstellung von Array-Variablen im Browser. Das `pre`-Tag, welches Sie über den `echo`-Befehl ausgeben, dient der zeilenweisen Darstellung der Arrays. Ohne das `pre`-Tag werden die Array-Daten in einer Zeile nacheinander und schwerer zu lesen dargestellt.

```
echo "<pre>";
print_r($_POST);
var_dump($_POST);
echo "</pre>";
```

Erweiterung der Beispieldatei „formular_auswertung.php“ um die angegebenen Zeilen

Die Ausgabe von `print_r()` sehen Sie in nebenstehender Abbildung: Die Schlüssel des `$_POST`-Arrays werden in eckigen Klammern angezeigt (sie entsprechen den `name`-Attributen der Formularelemente), daneben die Eingaben, die der Nutzer gemacht hat.

`var_dump()` liefert wie `print_r()` Schlüssel und Werte, darüber hinaus zusätzliche Informationen. Es wird angezeigt, wie viele Einträge das `$_POST`-Array hat, wie viele Zeichen jede einzelne Eingabe hat und von welchem Datentyp der Wert eines Array-Eintrags ist (allerdings handelt es sich im Falle von Formulardaten auch um den Datentyp `string`, auch wenn Ganz- oder Fließkommazahlen eingegeben wurden).



Anzeige der Daten, die aus dem Formular übertragen werden

Verschiedene Formularelemente

Die wichtigsten Formularelemente in HTML sind die Elemente `input`, `select` mit `option`, `textarea` und `button`. Das `input`-Element ist dabei hervorzuheben, da erst durch den Wert des `type`-Attributs das Eingabefeld zu einem speziellen Feld wird, z. B. `type="radio"` wird zum Radiobutton, `type="checkbox"` zur Checkbox, `type="file"` zum Upload-Button, `type="email"` wird zum E-Mail-Feld (Browser validieren die eingegebene E-Mail-Adresse, auf mobilen Geräten wird die Tastatur mit dem @-Zeichen angezeigt), `type="submit"` wird zum Absende-Button.

Mit HTML5 sind weitere `type`-Attributwerte hinzugekommen, die Formulare speziell auf die zu erwartenden Eingaben anpassbar machen, um die Bedienbarkeit zu verbessern (beispielsweise erzeugt `type="date"` ein Eingabefeld, welches einen Auswahlkalender anzeigt).

HTML5-Formularelemente werden von älteren Browsern teilweise nicht unterstützt, wie zum Beispiel dem Internet Explorer vor der Version 10. Aber auch aktuelle Browser unterstützen nicht alle `input`-Attribute, die der HTML5-Standard vorsieht. Was unterstützt wird, finden Sie unter <https://caniuse.com>.

Beispiel: *form-elemente.html*

In dieser HTML-Datei finden Sie eine Reihe von unterschiedlichen Formularelementen. Obwohl die Menge aller Formularelemente überschaubar ist, verändert sich die Darstellung besonders beim `type`-Attribut für das `input`-Feld. Aber nicht nur die Ansicht im Browser ist verschieden, auch bei der Vergabe des `name`-Attributs und bei Übergabe der Eingaben an die Zielseite sind ein paar Dinge zu beachten.

```
① <form action="form-elemente.php" method="get">
②   <p>E-Mail: <input type="email" name="email"></p>
③   <p>Passwort: <input type="password" name="password"></p>
④   <p>Datum: <input type="date" name="datum"></p>
⑤   <p>Farben: <br>
⑥     <input type="checkbox" name="gelb"> gelb
⑦     <input type="checkbox" name="blau"> blau
      <input type="checkbox" name="gruen"> gruen
      <input type="checkbox" name="rot" value="#f00"> rot</p>
⑧   <p>Ampel: <br>
⑨     <input type="radio" name="ampel" value="gruen"> grün
⑩     <input type="radio" name="ampel" value="gelb"> gelb
⑪     <input type="radio" name="ampel" value="rot"> rot</p>
⑫   <p>Speisen: <br>
⑬     <input type="checkbox" name="speisen[] "> Pizza
⑭     <input type="checkbox" name="speisen[] "> Nudeln
⑮     <input type="checkbox" name="speisen[] " value="salat"> Salat</p>
⑯   <p> Eis: <select name="eissorte">
⑰     <option></option>
⑱     <option>Schoko</option>
⑲     <option>Vanille</option>
⑳     <option>Nuss</option>
⑷   </select></p>
⑵   <p> Gemüse: <select name="gemuese">
⑶     <option></option>
⑷     <option value="sorte-1">Bohnen</option>
⑸     <option value="sorte-2">Erbsen</option>
⑹     <option value="sorte-3">Blumenkohl</option>
⑺   </select> </p>
```

```

⑧ <p> Obst: <select name="obst[]" multiple>
    <option>Apfel</option>
    <option>Birne</option>
    <option>Pflaume</option>
    <option>Orange</option>
  </select></p>
⑨ <p>Nachricht:<br><textarea name="memo"></textarea></p>
  <p><input type="submit" value="Abschicken" name="senden"></p>
① </form>

```

- ① Die Übertragungsmethode wurde hier bewusst mit der GET-Methode definiert. Schauen Sie sich nach Absenden des Formulars die URL an, z. B. finden Sie dort die Eingabe des Passwortes im Klartext. Achten Sie darauf, dass alle Formularfelder einschließlich des Absende-Buttons innerhalb des öffnenden und schließenden form-Tags liegen.
- ② Hier werden drei input-Elemente mit unterschiedlichen type-Attributen hinterlegt. Optisch sehen alle drei aus wie ein einzeiliges Eingabefeld. Allerdings unterscheiden Sie sich in der Bedienung durch den Nutzer. Ein input-Element vom type="email" überprüft die Eingabe auf eine gültige Syntax, mit dem type="date" wird je nach Browser ein Kalendersymbol oder, wie hier in Firefox, der Platzhalter TT.MM.JJJJ angezeigt, welcher auf ein Kalenderauswahlfeld hinweisen. Bei allen drei Feldern ist das name-Attribut vergeben, welches als Schlüssel im \$_GET-Array wieder zu finden ist. Die Schlüssel werden bei allen drei Feldern immer übergeben, auch wenn der Nutzer keine Eingabe vorgenommen hat.

Verschiedene Formularelemente

Bitte füllen Sie die nachfolgenden Eingabefelder aus:

E-Mail:

Passwort:

Datum:

Farben:

gelb blau gruen rot

Ampel:

grün gelb rot

Speisen:

Pizza Nudeln Salat

Eis:

Gemüse:

Obst:

Apfel
Birne
Pflaume
Orange

Nachricht:

Abschicken

Ausgabe der Beispieldatei „form-elemente.html“

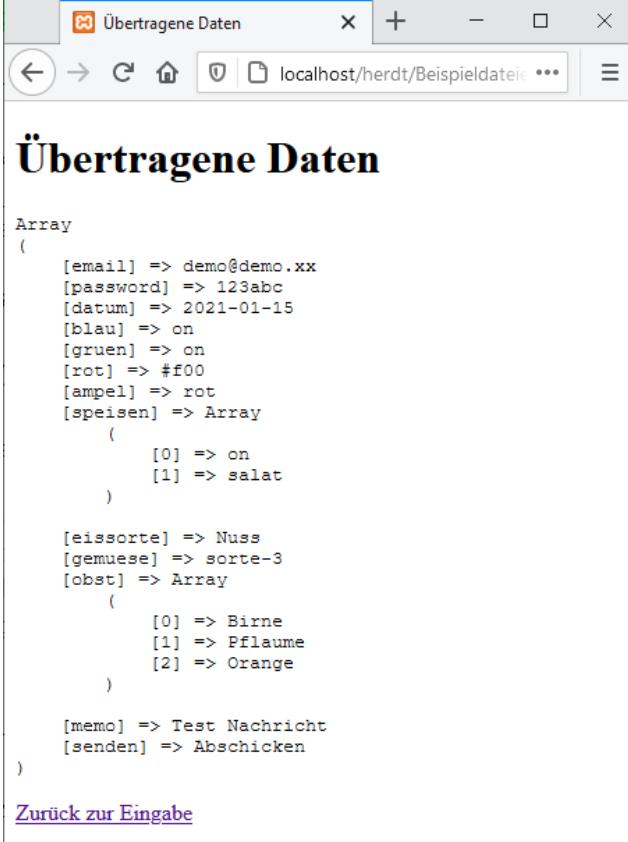
- ③ Über das type-Attribut checkbox erzeugen Sie hier vier Checkboxen. Als name-Attribut vergeben Sie auch hier einen Wert, den Sie als Schlüssel im \$_GET-Array wieder finden. Beachten Sie: Bei der vierten Checkbox ist zusätzlich der value="#f00" angegeben. In dem Fall, und falls die Checkbox ausgewählt wird, wird mit dem Schlüssel rot auch der Wert #f00 übergeben. Bei den anderen drei Boxen fehlt ein value-Attribut. In diesen Fällen ist der Wert zum Schlüssel im \$_GET-Array lediglich on, falls die Checkbox ausgewählt wurde. Mit value zu arbeiten ist z. B. sinnvoll, wenn Sie ein Produkt zur Auswahl anbieten, als value dann aber die Produkt-ID verwenden wollen, und nicht mit der Produktbezeichnung, die zur Ansicht im Browser dient.

Beachten Sie: Ist eine Checkbox nicht ausgewählt, werden weder Schlüssel noch ein möglicher Wert übertragen.

- ④ Über den Wert `radio` für das `type`-Attribut definieren Sie hier Radiobuttons. Damit diese miteinander korrespondieren, also nur ein Radiobutton von den drei auswählbar ist, **müssen** alle dasselbe `name`-Attribut haben. Um die Auswahl im PHP-Code erkennen zu können, **muss** bei einem Radiobutton auch immer ein `value` vergeben werden.

Auch für Radiobutton gilt: Wenn der Nutzer keine Auswahl trifft, werden weder Schlüssel noch Wert übertragen.

- ⑤ Hier werden weitere Checkboxen hinterlegt. Im Unterschied zu den Checkboxen in ③ wird nicht für jede Checkbox ein individuelles `name`-Attribut angegeben, sondern über den gleichen Wert `speisen` und die eckigen Klammern `[]` die Checkboxen gruppiert. Im `$_GET`-Array wird die Auswahl dieser Checkboxen in einem inneren indizierten Array mit dem Schlüssel `speisen` zusammengefasst. Auch hier gilt, wenn kein `value` vergeben wurde, erscheint im `$_GET`-Array lediglich ein `on` zu dem nicht aussagekräftigen Schlüssel des indizierten Arrays (Der Eintrag `$_GET["speisen"] [0]` entspricht der ersten Checkbox im HTML-Code mit dem `name`-Attribut `speisen []`). Erst ein `value` liefert einen verwertbaren Wert.
- ⑥ In dieser Zeile wird eine Selectbox mit dem HTML-Element `select` definiert. Die einzelnen Auswahlmöglichkeiten werden über jeweils ein `option`-Tag hinterlegt. Der ausgewählte Eintrag erscheint mit dem `name` des `select`-Elements als Schlüssel im `$_GET`-Array. Bei Selectboxen wird immer ein Schlüssel übertragen, auch wenn der Nutzer keine Auswahl getroffen hat.
- ⑦ Vergleichbar zu ⑥, allerdings haben hier die `option`-Tags `value`-Attribute. In dem Fall wird nicht der Wert übertragen, den der Nutzer sieht, sondern der Wert von `value`, den Sie hinterlegt haben.
- ⑧ Hier wird eine weitere Selectbox definiert, im Unterschied zu den vorherigen hat das `select`-Element das Attribut `multiple`. Dies erlaubt dem Nutzer, mehrere Optionen aus der Selectbox zu wählen. Damit im `$_GET`-Array alle ausgewählten Einträge übermittelt werden, **muss** das `name`-Attribut des `select`-Elements eckige Klammern `[]` haben, damit die Werte als Array übertragen werden. Wird als `name` ein Schlüssel ohne die Klammern angegeben, wird nur die letzte ausgewählte Option als Wert übertragen.



```

Übertrogene Daten
localhost/herdt/Beispieldatei ...
Array
(
    [email] => demo@demo.xx
    [password] => 123abc
    [datum] => 2021-01-15
    [blau] => on
    [gruen] => on
    [rot] => #f00
    [ampel] => rot
    [speisen] => Array
        (
            [0] => on
            [1] => salat
        )

    [eissorte] => Nuss
    [gemuese] => sorte-3
    [obst] => Array
        (
            [0] => Birne
            [1] => Pflaume
            [2] => Orange
        )

    [memo] => Test Nachricht
    [senden] => Abschicken
)

```

[Zurück zur Eingabe](#)

Ausgabe des `$_GET`-Arrays in der Datei „form-elemente.php“

- ⑨ Hier wird ein mehrzeiliges Eingabefeld definiert. Die HTML-Syntax unterscheidet sich von den `input`-Formularelementen, da nicht nur ein öffnendes `textarea`-Tag benötigt wird, sondern immer auch ein schließendes. Bei `textarea`-Elementen wird ein Schlüssel übertragen, auch wenn der Nutzer keine Eingabe vorgenommen hat.

! HTML5 bietet Formularelemente, die zum Teil mit einer implizierten Validierung einhergehen, z. B. werden E-Mail-Adressen auf Gültigkeit überprüft. Als PHP-Entwickler dürfen Sie sich nicht auf diese Validierung im Browser verlassen. Einerseits kann die implizierte Validierung einfach per `novalidate` im `form`-Tag deaktiviert werden, andererseits können die Daten über ältere Browser übermittelt werden, in denen noch keine HTML5-Elemente und damit eine Validierung nicht unterstützt wird. Die Korrektheit der Eingaben muss immer serverseitig mit PHP überprüft werden.

Formulare mit Checkboxen und Radiobuttons

Checkboxen und Radiobuttons bieten eine Besonderheit, die Sie bei der Auswertung mit PHP berücksichtigen müssen: Diese `name`-Attribute werden aus dem Formular nur übermittelt, wenn sie ausgewählt bzw. angeklickt wurden.

Beispiel: `formular-2.html`

Es wird zunächst eine HTML-Datei erstellt, in der ein Formular mit einer Gruppe von Checkboxen und einer Gruppe von Radiobuttons erstellt wird.

```

① <form action="formular_auswertung-2.php" method="POST">
    <p>Interessen:
        <input type="checkbox" name="interesse[]" value="Kultur"> Kultur
        <input type="checkbox" name="interesse[]" value="Musik"> Musik
        <input type="checkbox" name="interesse[]" value="Natur"> Natur
        <input type="checkbox" name="interesse[]" value="Sport"> Sport
    </p>
② <p>Zahlungsart:
    <input type="radio" name="zahlung" value="bar"> bar
    <input type="radio" name="zahlung" value="Scheck"> Scheck
    <input type="radio" name="zahlung" value="Überweisung"> Überweisung
</p>
<p><input type="submit" name="absenden" value="Abschicken">
    <input type="reset" value="Zurücksetzen"></p>
</form>
```

- ① Um Checkboxen für die Auswertung mit PHP vorzubereiten, muss im Formular jede Gruppe von Checkboxen den gleichen Namen – und zusätzlich folgende eckige Klammern `[]` – erhalten. Der Eingabename `interesse` wird damit als Array-Variable gekennzeichnet, die mehrere Werte aufnehmen kann.

- ② Im Formular wird eine Gruppe von Radio-buttons bereitgestellt. Hier ist für alle drei Radiobuttons dasselbe name-Attribut vergeben, damit diese miteinander korrespondieren. Für jeden Radiobutton wird ein value-Attribut vergeben, dessen Wert bei der Auswahl übergeben wird.

Formular, aus dem Daten übertragen werden

Beispiel: *formular_auswertung-2.php*

```
<?php
    echo "<pre>";
    print_r($_POST);
    echo "</pre>";
    if (!empty($_POST["interesse"])) {
        echo "<p>Folgende Interessen wurden angegeben:<br>";
        echo implode(", ", $_POST["interesse"]) . "</p>";
    }
?>
```

- ① Mit `print_r()` wird ausgegeben, welche Daten aus dem Formular übermittelt wurden. Da im Formular kein Radiobutton ausgewählt wurde, fehlt die Variable `$_POST["zahlung"]` in der Liste der übermittelten Variablen. Die Schaltfläche zum Absenden des Formulars hingegen wird in dieser Liste aufgeführt, da sie im HTML mit einem name-Attribut versehen ist.
- ② Es erfolgt die Prüfung, ob eine Checkbox im Formular ausgewählt wurde. Falls der Nutzer eine Auswahl getroffen hat, fällt diese Prüfung positiv aus, der folgende Anweisungsblock wird dann ausgeführt.
- ③ Über die Array-Funktion `implode()` werden die Werte aus der Variablen `$_POST["interesse"]` ausgelesen, mit Komma getrennt zusammengeführt und als Zeichenkette ausgegeben.

Anzeige der Beispieldatei „formular_auswertung-2.php“

Prüfung auf Existenz und Inhalt übergebener Schlüssel und Werte

Um Formulardaten flexibel auswerten zu können, können Sie mit PHP abfragen, ob

- ✓ Schlüssel vorhanden sind, also übertragen wurden oder
- ✓ Werte leer sind oder einen bestimmten Wert haben.

Mit `isset()` prüfen Sie die Existenz der in den Klammern angegebenen Variablen. Existiert die Variable, wird TRUE zurückgeliefert, ansonsten FALSE. `isset()` wird häufig in Verbindung mit der `if`-Anweisung verwendet, damit ein Anweisungsblock, der bestimmte Werte erwartet, nur dann ausgeführt wird, wenn der Wert auch gesetzt ist. Da folgende Elemente nur Schlüssel und Werte übergeben, wenn diese ausgewählt oder geklickt wurden, empfiehlt sich bei diesen Elementen die Prüfung per `isset()`:

- ✓ Checkboxen
- ✓ Radiobuttons
- ✓ Submit-Schaltflächen

```
isset(Variablen);
if(isset(Variablen)) {
    Anweisungsblock;
}
```

Mit `empty()` prüfen Sie, ob eine angegebene Variable einen Wert enthält. Ist die geprüfte Variable nicht leer und hat sie einen von 0 unterschiedlichen Wert, liefert `empty()` FALSE zurück, ansonsten TRUE.

```
empty(Variablen);
if(!empty(Variablen)) {
    Anweisungsblock;
}
```

Text-Eingabefelder werden bei der Übertragung von Formulardaten immer übermittelt, auch wenn kein Wert eingetragen wurde. In diesen Fällen hilft die Überprüfung mit `empty()`. Dies gehört zu den gängigen Methoden, um zu prüfen, ob Pflichtfelder in Formularen gefüllt wurden.



Leerzeichen in Text-Eingabefeldern werden übertragen und sind als Werte im `$_POST`- bzw. `$_GET`-Array gespeichert. Eine Prüfung mit `empty()` schlägt hier fehl, auch wenn Sie im Browser keinen Eintrag sehen. Die Prüfung eines Leerzeichens beantwortet `empty()` mit FALSE. Hier empfiehlt es sich, die Prüfung um `trim()` zu erweitern. Diese PHP-Funktion entfernt Leerzeichen und Zeilenumbrüche am Anfang und Ende von Zeichenketten. `if empty(trim($_POST["nachname"]))` liefert TRUE, auch wenn dort ausschließlich Leerzeichen eingegeben wurden (mehr zu `trim()` vgl. Abschnitt 9.7).

Mehrere Absende-Schaltflächen

Mit PHP können Sie prüfen, welche Submit-Schaltfläche gedrückt wurde, die ein Formular abgesendet hat. Zu diesem Zweck integrieren Sie in Ihr Formular beliebig viele Submit-Schaltflächen und geben den Schaltflächen eindeutige Namen, z. B.:

```
<input type="submit" name="eins">
<input type="submit" name="zwei">
```

In der auswertenden Datei können Sie gezielt abfragen, welche der Schaltflächen gedrückt wurde. Da eine Submit-Schaltfläche das Formular umgehend übermittelt, kann immer nur genau eine Schaltfläche angeklickt worden sein. Formulare mit mehreren Submit-Schaltflächen sind eine Möglichkeit der Umsetzung, wenn dasselbe Formular verschiedene Aktionen auslösen kann.

```
if(isset($_POST["eins"])) {
    Anweisungsblock, wenn Schaltfläche eins gedrückt wurde
}
if(isset($_POST["zwei"])) {
    Anweisungsblock, wenn Schaltfläche zwei gedrückt wurde
}
```

Beispiel: *form_multi.html*

Es wird zunächst eine HTML-Datei erstellt, die mehrere Submit-Schaltflächen zum Absenden des Formulars hat. In der auswertenden Datei wird geprüft, welche Schaltfläche betätigt wurde, sodass eine entsprechende Aktion ausgeführt werden kann.

```
① <h1>Berechnungen mit zwei Zahlen</h1>
<p>Bitte geben Sie zwei Zahlen ein: </p>
<form action="form_multi-auswertung.php" method="POST">
    <p>Erste Zahl: <input type="text" name="zahl1"></p>
    <p>Zweite Zahl: <input type="text" name="zahl2"></p>
② <p><input type="submit" name="mal" value="Zahlen multiplizieren">
    <input type="submit" name="plus" value="Zahlen addieren"></p>
</form>
```

- ① Über das `action`-Attribut im `form`-Tag wird die Ziel-PHP-Datei festgelegt, an welche die Formulardaten übermittelt werden sollen.
- ② Es werden zwei Submit-Schaltflächen in das Formular integriert. Über die Schaltfläche mit dem `name`-Attribut *mal* sollen die beiden eingegebenen Zahlen miteinander multipliziert werden.
- ③ Über die Schaltfläche mit dem `name`-Attribut *plus* sollen die Zahlen hingegen addiert werden.

Beispiel: *form_multi-auswertung.php*

```
① <?php
② echo "<h1>Rechenergebnis</h1>";
if (isset($_POST["mal"])) {
    $ergebnis = $_POST["zahl1"] * $_POST["zahl2"];
    echo "<p>" . $_POST["zahl1"] . " mal " . $_POST["zahl2"] . " ist
gleich $ergebnis.</p>";
}
③ if (isset($_POST["plus"])) {
    $ergebnis = $_POST["zahl1"] + $_POST["zahl2"];
    echo "<p>" . $_POST["zahl1"] . " plus " . $_POST["zahl2"] . " ist
gleich $ergebnis.</p>";
}
?>
```

- ① Die Überschrift wird über eine echo-Anweisung ausgegeben.
- ② Mithilfe der Funktion `isset()` wird geprüft, ob die Variable `$_POST["mal"]` übermittelt wurde. Wenn die Prüfung TRUE ergibt, wurde die Schaltfläche *Zahlen multiplizieren* zum Absenden des Formulars verwendet. Der folgende Anweisungsblock wird ausgeführt.
- ③ Es wird geprüft, ob die Variable `$_POST["plus"]` vorhanden ist. Wenn die Prüfung TRUE ergibt, wurde die Schaltfläche *Zahlen addieren* verwendet. Der folgende Anweisungsblock wird ausgeführt, die Werte werden addiert und ausgegeben.

The screenshot displays two browser windows. The left window is titled 'Formular mit mehreren Submit' and contains a form with two input fields labeled 'Erste Zahl:' and 'Zweite Zahl:', and two buttons: 'Zahlen multiplizieren' and 'Zahlen addieren'. The right window is titled 'Rechenergebnis' and displays the text '47 plus 11 ist gleich 58'.

Anzeige der Beispieldateien „form_multi.html“ und „form_multi-auswertung.php“ nach Absenden des Formulars über die Schaltfläche „Zahlen addieren“

! Dieses Beispiel setzt voraus, dass der Nutzer eine Schaltfläche drückt. Ein Formular kann jedoch auch über die Return- bzw. Enter-Taste abgesendet werden (wenn sich der Cursor in einem Eingabefeld befindet). Beim Absenden durch Bedienen der Return-Taste wird nur der Wert einer Submit-Schaltfläche übermittelt. Hier verhalten sich die verschiedenen Browser teilweise unterschiedlich.

Formular und Auswertung in derselben PHP-Datei

In der Praxis werden oftmals PHP-Dateien verwendet (Dateiendung `*.php`), die nicht nur das Formular, sondern auch den Code zur Verarbeitung des Formulars beinhalten. In größeren Projekten können Sie so einige Seiten einsparen, außerdem können Sie Hinweise auf Fehler, die Sie bei der Auswertung der Formulardaten ermittelt haben, direkt in das Formular einbauen, was die umständliche Übergabe der Fehler und das erneute Laden der Ausgangsdatei überflüssig macht.

Um mit einem Formular auf dieselbe Datei zu verweisen, in dem das Formular hinterlegt ist, haben Sie über das `action`-Attribut des HTML-Tags `<form>` folgende Möglichkeiten. Notieren Sie im `action`-Attribut:

- ✓ den eigenen Dateinamen, z. B. `formular.php`
- ✓ die PHP-Variable `$_SERVER["SCRIPT_NAME"]` (`$_SERVER` gehört zu den superglobalen Variablen und liefert Informationen über den Webserver. Der Eintrag `SCRIPT_NAME` enthält den Namen der Datei, die aufgerufen ist)
- ✓ `[?]`, also `action="?"` (nicht empfohlen)
- ✓ Leerwert, also `action=""` (nicht empfohlen)

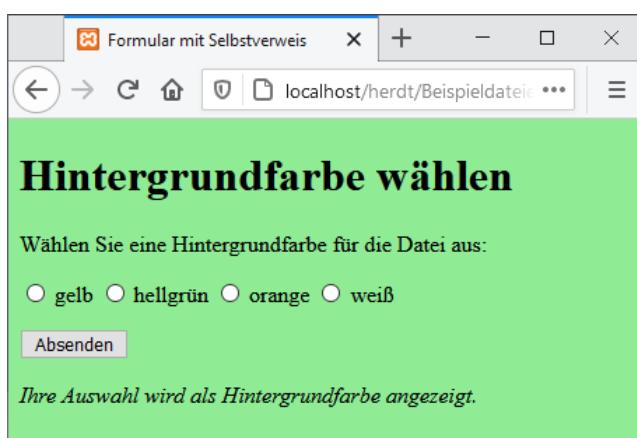
Beispiel: *selbstverweis.php*

Es wird eine PHP-Datei mit einem Formular erstellt. Mit dem Formular können Sie die Hintergrundfarbe der Datei steuern. Durch Absenden des Formulars wird die Seite selbst mit der gewählten Hintergrundfarbe aufgerufen.

```

① <body style="background: <?php if (isset($_POST["hintergrund"])) echo
                      $_POST["hintergrund"]; ?>">
    <h1>Hintergrundfarbe wählen</h1>
    Wählen Sie eine Hintergrundfarbe für die Datei aus:
    <form action="<?php echo $_SERVER["SCRIPT_NAME"]; ?>"
          method="POST">
        <p><input type="radio" name="hintergrund" value="#FFFF00"> gelb
           <input type="radio" name="hintergrund" value="#8FEC95">
           hellgrün
           <input type="radio" name="hintergrund" value="#FFA000"> orange
           <input type="radio" name="hintergrund" value="#FFFFFF">
           weiß</p>
        <p><input type="submit" name="absenden" value="Absenden"></p>
    </form>
    <?php
    if (isset($_POST["absenden"]) && isset($_POST["hintergrund"])) {
        echo "<p><em>Ihre Auswahl wird als Hintergrundfarbe
              angezeigt.</em></p>";
    }
    ?>
</body>
```

- ① Im HTML-Tag `<body>` wird über PHP der Wert der Farbwert für die CSS-Eigenschaft `background` gesetzt und damit die Hintergrundfarbe für diese Datei festgelegt. Dies geschieht in Abhängigkeit von der aus dem Formular übermittelten Variable `hintergrund`. Damit es nicht zu einer PHP-warning im Browser kommt, wird zusätzlich über `isset()` abgefragt, ob die Variable auch gesetzt ist.
- ② Im `action`-Parameter des HTML-Tags `<form>` wird angegeben, dass das Formular „an sich selbst“ gesendet werden soll. Dieselbe Datei mit dem HTML-Formular wird auch zur Auswertung der übertragenen Formulardaten verwendet.
- ③ Es wird geprüft, ob die Variablen `absenden` und `hintergrund` aus dem Formular übermittelt wurden. Die Prüfungen sind verknüpft: Nur wenn beide Prüfungen TRUE zurückliefern (also das Formular abgesendet UND eine Farbe ausgewählt wurde), wird die anschließende `echo`-Anweisung ausgeführt.



Anzeige der Beispieldatei „selbstverweis.php“

6.3 Übungen

Übung 1: Ein Formular für eine Umfrage erstellen

Level		Zeit	ca. 15 min
Übungsinhalte	<ul style="list-style-type: none"> ✓ HTML-Formulare ✓ Formulare mit PHP auswerten ✓ Ausgabe von Formulardaten ✓ Überprüfung auf Formulareingaben 		
Übungsdatei	--		
Ergebnisdateien	<i>uebung_formular.html, uebung_auswertung.php</i>		

Erstellen Sie das nachstehende Formular für die Umfrage und speichern Sie es unter dem Namen *uebung_formular.html*.



The screenshot shows a web browser window with the title "Übung (Kapitel 6)". The page content is as follows:

Bitte füllen Sie das Formular komplett aus

Persönliche Daten

Vorname:

Nachname:

Wohnort:

Was wir wissen wollen

Wie wohnen Sie? Einfamilienhaus Eigentumswohnung Mehrfamilienhaus

Welche TV-Sendungen sehen Sie gern? Dokumentationen Nachrichten Spielfilme Sport

Haben Sie noch eine Nachricht für uns?

Daten speichern **Zurücksetzen**

Ausgabe „uebung_formular.html“

Geben Sie nach dem Absenden des Formulars eine Bestätigungsseite aus. Diese Bestätigungsseite soll die vorher eingegebenen Daten zeigen. Speichern Sie die Datei unter dem Namen *uebung_auswertung.php*. Überprüfen Sie auch, ob eine Nachricht eingegeben wurde und geben Sie *keine* aus, falls *keine* Nachricht gesendet wurde.

Ausgabe der Ergebnisdatei „uebung_auswertung.php“

Übung 2: Informationen über ein Formular anfordern

Level		Zeit	ca. 15 min
Übungsinhalte	<ul style="list-style-type: none"> ✓ HTML-Formulare ✓ Formulare mit PHP auswerten ✓ Ausgabe von Formulardaten 		
Übungsdatei	--		
Ergebnisdatei	<i>uebung_formular2.php</i>		

Erstellen Sie eine neue PHP-Datei (*uebung_formular2.php*) mit dem nachstehenden Formular. Nach dem Absenden des Formulars soll die Auswertung der Formulardaten in derselben Datei vorgenommen werden. Überprüfen Sie, ob die Formularinhalte übergeben werden. Füllen Sie die Textfelder wieder mit den Eingaben und markieren Sie den Radiobutton, welcher ausgewählt wurde, als checked. Bestätigen Sie dem Nutzer, mit welchem Anliegen er das Formular ausgefüllt hat.

Ausgabe der Beispiellösung (Datei „uebung_formular2.php“)

7

Funktionen

 **Beispieldateien:** Dateien im Ordner *Kapitel_07*

7.1 Funktionen erstellen und aufrufen

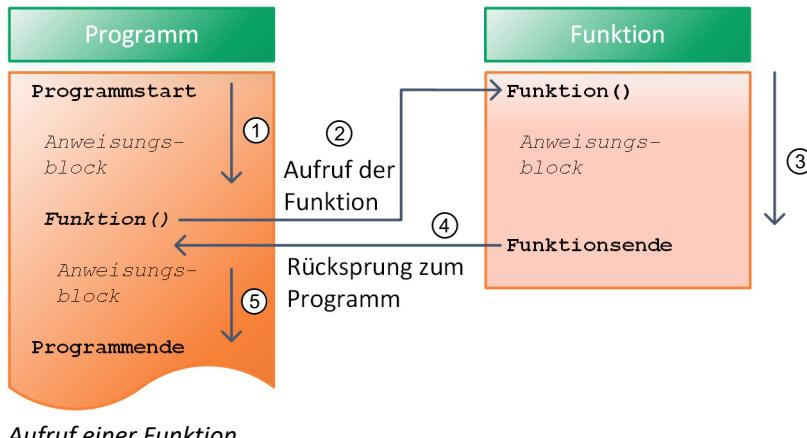
Was sind Funktionen?

Funktionen sind eigenständige Programmteile, die vom Skript beliebig oft aufgerufen und abgearbeitet werden können. Funktionen beinhalten Anweisungen, die innerhalb des Programms oder innerhalb eines Projekts mit mehreren Programmdateien mehrmals benötigt werden. Anstatt die Anweisungen mehrfach im Programm zu codieren, wird die entsprechende Funktion einmalig definiert und an den gewünschten Stellen aufgerufen, um die Anweisungen der Funktion dort auszuführen.

PHP bietet eine Vielzahl vordefinierter Funktionen, um bestimmte Standardaufgaben zu lösen. Funktionen, die Sie selbst zur Lösung Ihrer Aufgaben erstellen, werden **benutzerdefinierte Funktionen** genannt.

Vorteile von Funktionen

- ✓ Immer wiederkehrende Abläufe werden nur einmal programmiert und können danach beliebig oft ausgeführt werden.
- ✓ Der Programmcode wird durch Funktionen strukturiert, lässt sich leichter nachvollziehen, ist dadurch übersichtlicher und einfacher zu pflegen.
- ✓ Änderungen am Programm lassen sich schneller und einfacher durchführen, da eine Änderung nur in der Funktion nötig ist.



Eine Funktion wird erst ausgeführt, wenn sie im Programm aufgerufen wird. Dies geschieht über den Namen der Funktion, gefolgt von einem runden Klammerpaar `()`. In den runden Klammern können je nach Definition ein oder mehrere Parameter stehen, die der Funktion übergeben werden und dort verarbeitet werden. Unabhängig davon, wo eine Funktion im PHP-Code steht, wird der Funktionsblock bei der zeilenweisen Abarbeitung eines PHP-Skripts übersprungen. Die Funktion wird ausschließlich durch einen expliziten Aufruf ausgeführt.

Das eigentliche PHP-Skript wird von oben nach unten abgearbeitet (1). Der Funktionsaufruf (2) erzwingt einen Sprung zur angegebenen Funktion. Jetzt werden die Anweisungen innerhalb der Funktion abgearbeitet (3). Ist das Ende der Funktion erreicht, wird zum Programm zurückgesprungen (4). Das PHP-Skript wird hinter dem Funktionsaufruf weiter zeilenweise ausgeführt (5).

An welcher Stelle im PHP-Skript die Funktion definiert wird, spielt keine Rolle. Daher ist es trotz der zeilenweisen Abarbeitung des Skripts möglich, eine Funktion aufzurufen, die weiter oben steht oder erst weiter unten definiert wird.

Zum guten Programmierstil gehört es jedoch, alle selbst geschriebene Funktionen zu bündeln – entweder am Anfang oder am Ende eines Skripts. Oder diese in separate Dateien auszulagern (mehr dazu am Ende dieses Kapitels). So strukturieren Sie Ihre PHP-Datei und PHP-Code ist damit einfacher und besser zu pflegen.

Eine Funktion erstellen

Syntax und Beschreibung der `function`-Anweisung

```
function name([Parameter]) {
    Anweisungsblock;
}
```

- ✓ Das reservierte Schlüsselwort `function` leitet eine Funktion ein.
name ist die Bezeichnung der Funktion (Funktionsname) und ist frei wählbar. Für den Namen einer Funktion gelten folgende Regeln:
 - ✓ Der Funktionsname darf nur aus Buchstaben, Ziffern und dem Unterstrich `_` bestehen.
 - ✓ Das erste Zeichen muss ein Buchstabe oder ein Unterstrich sein.
 - ✓ Verboten sind Umlaute, das `ß` und alle Sonderzeichen außer dem Unterstrich `_`.

- ✓ Sowohl Groß- als auch Kleinbuchstaben dürfen verwendet werden, allerdings:
 - ✓ Funktionsnamen sind **nicht case-sensitiv**: Bei Funktionsnamen unterscheidet der PHP-Interpreter **nicht** zwischen Groß- und Kleinschreibung.
 - ✓ Der Name darf nicht identisch mit einem sogenannten reservierten Wort (z. B. Befehl aus PHP) sein.
- ✓ In den runden Klammern werden die Bezeichnungen der einzelnen Parameter (auch Über-gabewerte genannt) angegeben, für die beim Aufruf der Funktion Werte übergeben werden können. Im Funktionskopf können beliebig viele (auch keine) Parameter definiert werden. Die einzelnen Parameter werden durch Kommata voneinander getrennt.
- ✓ Das Schlüsselwort `function`, der Funktionsname, die runden Klammern und die Parameter stellen gemeinsam den Funktionskopf dar.

Der Funktionsname sollte einen Bezug zu der Aufgabe haben, welche die Funktion erfüllt. Eine Funktion, die z. B. das Quadrat einer Zahl berechnet, können Sie mit `quadratzahl()` oder `berechnung_quadrat()` benennen.

Besteht ein Funktionsname aus mehreren Begriffen, können die einzelnen Begriffe durch den Unterstrich `_` getrennt werden, z. B. `berechne_quadrat_zahl()`. Alternativ können Sie die CamelCase-Schreibweise verwenden, z. B. `berechneQuadratZahl()`. Die gewählte Schreibweise sollte im Skript einheitlich verwendet werden.

Eine Funktion mit `return`-Anweisung erstellen

Syntax und Beschreibung der `return`-Anweisung

- ✓ Mit der `return`-Anweisung gibt die Funktion einen Wert in den Programmablauf zurück.


```
function name([Parameter]) {
    Anweisungsblock;
    return [$wert];
}
```
- ✓ Sobald die `return`-Anweisung ausgeführt wird, wird eine Funktion verlassen und es erfolgt eine Rückkehr an die aufrufende Stelle. Eventuell folgende Programmzeilen innerhalb der Funktion werden nicht mehr ausgeführt.
- ✓ Die `return`-Anweisung kann auch ohne Rückgabewert ausschließlich zum Verlassen einer Funktion verwendet werden.

Wenn Sie kein `return` definiert haben, geben Funktionen trotzdem einen Wert zurück, und zwar `NULL`. Dieser ist notwendig, damit die aufrufende Stelle weiß, dass die Abarbeitung der Funktion beendet ist.

Datentypen für Parameter und Rückgabewert definieren

PHP 7.0 hat die Definition von Datentypen in Funktionen stark weiterentwickelt. Funktionsparameter können neben `array` und `object` auch auf skalare Werte (`int, string, float, boolean`) festgelegt werden. Neu seit PHP 7.0 ist die Definition des Datentyps für den Rückgabewert.

Die Festlegung der Datentypen hat den großen Vorteil, dass Sie Ihr PHP-Skript besser kontrollieren können. Wird eine Funktion z. B. mit einem *string*-Wert aufgerufen, obwohl Sie den Parameter als *int*-Wert definiert haben, gibt Ihnen PHP eine Fehlermeldung aus.

```
function name( [Datentyp|Datentyp] [Parameter]
                ) : [Datentyp|Datentyp] {
    Anweisungsblock;
}
```

- ✓ Der Datentyp wird vor dem Parameter notiert (z. B. *int, string*). Damit legen Sie fest, mit welchem Datentyp die Funktion aufgerufen werden darf. Die Angabe des Datentyps ist optional. Wird kein Datentyp angegeben, greift die automatische Datentyp-Zuweisung von PHP.
- ✓ Hinter der runden Klammer `()` und vor der geschweiften Klammer `{ }` kann optional der Datentyp für den Rückgabewert angegeben werden (z. B. *int, string*). Der Datentypdefinition wird der Doppelpunkt `:` vorangestellt. Damit legen Sie fest, von welchem Datentyp der Rückgabewert der Funktion ist.
- ✓ Seit PHP 8.0 kann der Datentyp auch mit sogenannten Union Types definiert werden. Die einzelnen Datentypen werden mit einem Pipe-Zeichen `|` voneinander getrennt angegeben. Das bedeutet, Sie sind nicht mehr auf einen Datentyp festgelegt, sondern können eine Gruppe von Datentypen übergeben, z. B. `bool | int | float`. Bei diesem Beispiel wären Boolean-, Integer- und Float-Werte erlaubt. Dies kann sowohl für die Deklaration der Funktionsvariablen als auch für den Rückgabewert eingesetzt werden.
- ✓ Haben Sie z. B. den Datentyp *int* für Integer angegeben, führt ein Rückgabewert vom Datentyp *string* zu einer Fehlermeldung. Dabei greift die automatische Datentyp-Konvertierung von PHP. Kann ein Wert umgewandelt werden (z. B. von *int* nach *string*), kommt es zu einer Fehlermeldung vom Typ **notice**. Kann ein Wert nicht umgewandelt werden (z. B. *array* nach *int*), kommt es zu einem **fatal error**, das Skript bricht ab.

Eine Funktion aufrufen

Syntax und Beschreibung eines Funktionsaufrufs

- ✓ Eine Funktion können Sie von jeder beliebigen Stelle im PHP-Code aufrufen.
- ✓ Eine Funktion wird mit ihrem Namen und folgendem runden Klammernpaar `()` aufgerufen.
- ✓ Die runden Klammern nach dem Funktionsnamen können Parameter zur Übergabe von Werten enthalten. Falls keine Werte übergeben werden, bleiben die Klammern leer.
- ✓ Ob und welche Parameter übergeben werden können, wird durch die Funktionsdefinition bestimmt.
- ✓ Sie können eine Funktion auch aus einer anderen Funktion heraus aufrufen.
- ✓ Eine Funktion kann auch sich selbst aufrufen (rekursiver Aufruf).

```
<?php
    Anweisungsblock
    funktionsname();
    Anweisungsblock
?>
```

Die einzelnen Parameter werden mit Kommas `,` voneinander getrennt. Seit PHP 8.0 darf auch hinter dem letzten Parameter ein Komma `,` stehen. Bislang führte ein `,` hinter dem letzten Parameter zu einem **fatal error**. Dies kann nun nicht mehr passieren, eine Fehlerquelle in PHP ist damit entfernt.

7.2 Mit Funktionen arbeiten

Funktionen ohne Parameter

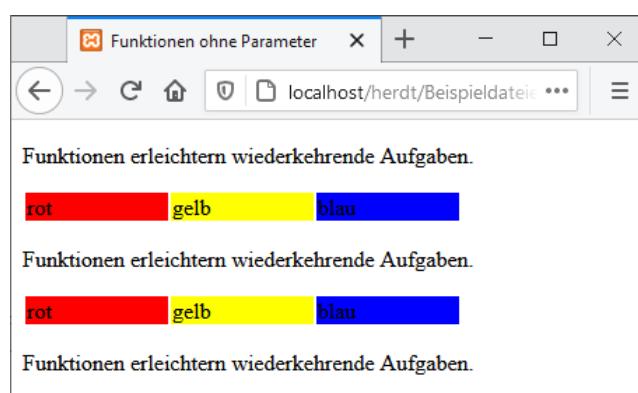
Bei einer Funktion ohne Parameter werden bei jedem Aufruf dieselben Anweisungen ausgeführt.

Beispiel: *funktionen-1.php*

```
<?php
    // Funktionsdefinitionen
①     function textausgabe() {
        echo "<p>Funktionen erleichtern wiederkehrende Aufgaben.</p>";
    }

②     function farbtabelle() {
        echo "<table>
            <tr>
                <td style='background:#FF0000; width:100px;'>rot</td>
                <td style='background:#FFFF00; width:100px;'>gelb</td>
                <td style='background:#0000FF; width:100px;'>blau</td>
            </tr>
        </table>";
    }
    // normaler Programmablauf
③     textausgabe();
④     farbtabelle();
③     textausgabe();
④     farbtabelle();
⑤     TEXTausgabe(); // nicht Case-Sensitiv!
?
>
```

- ① Die Funktion `textausgabe()` wird definiert, in der ein einfacher Text ausgegeben wird.
- ② Es erfolgt die Definition der Funktion `farbtabelle()`, die eine einfache Tabelle mit verschiedenfarbigem Hintergrund für jede Zelle ausgibt.
- ③ Die Funktion `textausgabe()` wird aufgerufen.
- ④ Die Funktion `farbtabelle()` wird ausgeführt.
- ⑤ Die Funktion wird hier teilweise mit Großbuchstaben im Funktionsnamen `TEXTausgabe()` aufgerufen. Da Funktionen nicht case-sensitiv sind, wird die Funktion ebenfalls ausgeführt.



Anzeige der Beispieldatei „funktionen-1.php“

Funktionen mit einem oder mehreren Parametern

Bei einer Funktion mit Parametern können individuelle Werte an die Funktion übergeben werden. Damit kann eine Funktion die gleiche Aufgabe mit variablen Werten ausführen. Hierbei kann ein Parameter eine Zeichenkette, ein Zahlenwert, eine Konstante oder eine Variable sein, aber auch Arrays oder Objekte können Funktionen übergeben werden.

Beispiel: *funktionen-2.php*

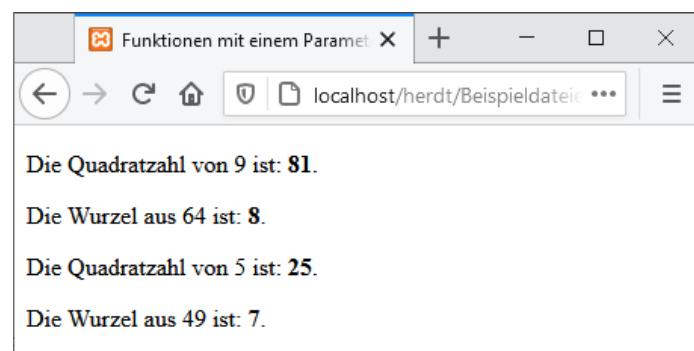
Im folgenden Skript werden mathematische Berechnungen mithilfe von Funktionen mit jeweils einem Parameter durchgeführt.

```
<?php
③    function quadrat($zahl) {
        $ergebnis = $zahl * $zahl;
        echo "<p>Die Quadratzahl von $zahl ist:
              <strong>$ergebnis</strong>.</p>";
    }
④    function wurzel($zahl) {
        $ergebnis = sqrt($zahl);
        echo "<p>Die Wurzel aus $zahl ist:
              <strong>$ergebnis</strong>.</p>";
    }
①    quadrat(9);
②    wurzel(64);
①    quadrat(5);
②    wurzel(49);
?>
```

- ①/② Die Funktionen `quadrat()` und `wurzel()` werden jeweils zweimal aufgerufen.
Der übergebene Parameter (die Zahl in der Klammer) ist bei jedem Aufruf ein anderer.

- ③ Der Wert in der Klammer wird beim Funktionsaufruf der Variablen `$zahl` zugewiesen. Diese Zuweisung geschieht bei jedem Funktionsaufruf erneut. Mit der Variablen `$zahl` wird beim Aufruf bei den Funktionen die Variable `$ergebnis` berechnet und am Bildschirm ausgegeben.

- ④ Aus einer Funktion heraus können andere Funktionen aufgerufen werden, in diesem Fall eine von PHP vordefinierte mathematische Funktion `sqrt()` zur Berechnung der Quadratwurzel.



Anzeige der Datei „funktionen-2.php“

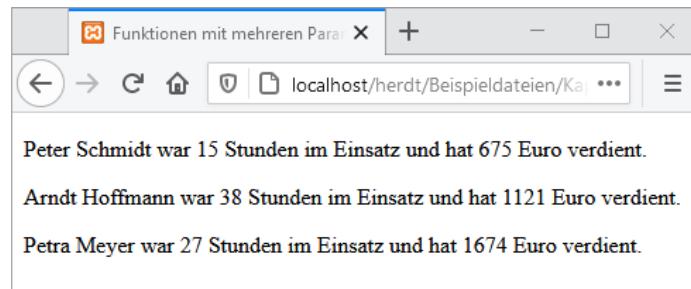
Beispiel: funktionen-3.php

Bei Funktionen mit mehreren Parametern werden diese in der **Reihenfolge** des Aufrufs übergeben. Der erste Parameter im Funktionsaufruf wird dem ersten Parameter in der Funktionsdeklaration übergeben, der zweite dem zweiten usw. Die einzelnen Parameter werden mit Kommata voneinander getrennt.

```

① <?php
    function honorar_berechnen($dozent, $stundenzahl,
                                $honorarsatz) {
        $honorar = $stundenzahl * $honorarsatz;
        echo "<p>$dozent war $stundenzahl Stunden im Einsatz und
              hat $honorar Euro verdient.</p>";
    }
② honorar_berechnen("Peter Schmidt", 15, 45);
② honorar_berechnen("Arndt Hoffmann", 38, 29.5);
② honorar_berechnen("Petra Meyer", 27, 62);
?>
```

- ① Die Funktion `honorar_berechnen()` wurde mit 3 Parametern definiert: `$dozent`, `$stundenzahl`, `$honorarsatz`.
- ② Beim Aufruf der Funktion werden die drei geforderten Parameter durch Kommata getrennt übergeben. Hinter dem letzten Parameter steht ein (Trailing comma), was seit PHP 8.0 erlaubt ist.



Anzeige der Beispieldatei „funktionen-3.php“

Wenn Sie einer Funktion weniger Parameter übergeben als im Funktionskopf definiert sind, erhalten Sie eine entsprechende Fehlermeldung (Ausnahme: Funktionen mit optionalen Parametern). Übergeben Sie einer Funktion mehr Parameter, als im Funktionskopf definiert sind, kommt es zu keiner Fehlermeldung, die überschüssigen Werte sind dann keinem Parameter im Funktionskopf zugewiesen. Allerdings können die überzähligen Werte über die PHP-Funktion `func_get_args()` ermittelt werden.

! Soll Ihr PHP-Skript abwärtskompatibel sein, also auch mit PHP 7 und darunter lauffähig sein, dürfen Sie **kein** Komma hinter dem letzten Parameter setzen.

Optionale Parameter

Standardmäßig müssen Sie einer Funktion mindestens so viele Parameter übergeben, wie im Funktionskopf deklariert sind. Fehlt ein Wert im Funktionsaufruf, gibt PHP eine Meldung der Kategorie **fatal error** aus, das PHP-Skript bricht dann ab.

```

function name(Parameter = Standardwert) {
    Anweisungsblock;
}
```

Sie können eine Variable in der Funktionsdeklaration jedoch mit einem Standardwert versehen. Damit deklarieren Sie diesen Parameter zu einem **optionalen Parameter**. Der Standardwert wird auch als **Vorgabewert** bezeichnet. Für optionale Parameter können Sie, müssen jedoch keine Werte übergeben.

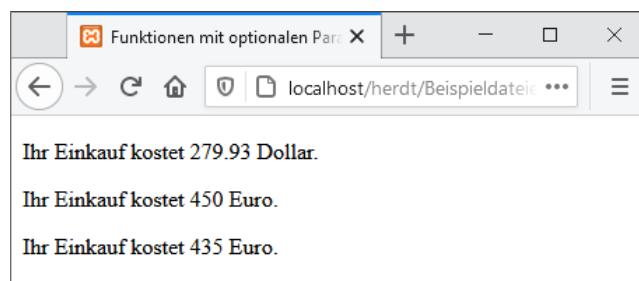
PHP arbeitet dann wie folgt: Wird im Funktionsaufruf ein Wert für den optionalen Parameter übergeben, verwendet die Funktion den Übergabewert für diese Variable, der Standardwert wird dann ignoriert. Wird hingegen **kein** Wert für diesen Parameter übergeben, verwendet die Funktion den Standardwert. Ein fehlender Wert führt **nicht** zu einer Fehlermeldung.

! Vor PHP 8.0 **mussten** Parameter, die einen Wert erwarten, als **erste Werte** angegeben werden. Optionale Werte **mussten** dahinter notiert werden. Diese verpflichtende Reihenfolge ist mit PHP 8.0 entfallen. Damit Ihr PHP-Skript mit älteren PHP-Versionen lauffähig ist, sollte die Reihenfolge – zuerst erforderliche, dann optionale Parameter – beibehalten werden.

Beispiel: *optional.php*

```
<?php
①  function berechne($anzahl, $preis = 45, $waehrung = "Euro") {
    echo "<p>Ihr Einkauf kostet " . ($anzahl * $preis) . " "
    $waehrung.</p>";
}
②  // normaler Aufruf der Funktion
berechne(7, 39.99, "Dollar");
③  // Parameter 2 und 3: Standardwerte werden verwendet
berechne(10);
④  // Parameter 3: Standardwert wird verwendet
berechne(15, 29);
?>
```

- ① Die Funktion `berechne()` wurde mit 3 Parametern programmiert: `$anzahl`, `$preis`, `$waehrung`. Für die letzten beiden Parameter sind Standardwerte definiert worden. Es handelt sich also um optionale Parameter, deren Werte verwendet werden, wenn der zweite und/oder dritte Parameter beim Funktionsaufruf nicht angegeben wird.
- ② Die Funktion wird mit allen definierten Parametern aufgerufen.
- ③ Die Funktion wird mit nur einem Parameter aufgerufen. PHP interpretiert den angegebenen Parameter automatisch als den **ersten** Parameter und weist den Wert der Variablen `$anzahl` zu. Der zweiten und dritten Variablen im Funktionskopf werden die Standardwerte zugewiesen und für die Ausführung der Funktion verwendet.
- ④ Entsprechend erfolgt der Aufruf mit zwei Werten. Die Zahl 29 wird der zweiten Variable `$preis` zugewiesen, der Standardwert wird hier nicht verwendet. Der dritte Parameter erhält wieder den Standardwert.



Anzeige der Beispieldatei „optional.php“

Parameterübergabe per Wert oder per Referenz

Wenn Sie eine Variable per Wert als Parameter an eine Funktion übergeben, dann erhält die Funktion eine **Kopie** der übergebenen Variablen. Die Übergabe der Parameter als Kopie wird auch als **call-by-value** bezeichnet und stellt das Standardverhalten der Parameterübergabe dar. Wenn der Wert einer übergebenen Variablen in der Funktion geändert wird, wird nur die Kopie des Parameters innerhalb der Funktion geändert. Eine Veränderung der Kopie hat keinerlei Rückwirkung auf den Parameter, der vom Haupt-PHP-Skript übergeben wurde, der Variablenwert im Hauptskript bleibt unverändert.

Alternativ gibt es die Möglichkeit, eine Variable per Referenz zu übergeben. Dies erzielen Sie durch ein dem Parameter vorangestelltem **&**-Zeichen. In dem Fall wird nicht nur der Wert innerhalb der Funktion, sondern auch der Wert im Hauptskript verändert, da Sie nicht mit einer Kopie, sondern mit der Variablen selbst arbeiten. Diese Vorgehensweise nennt man **call-by-reference**.

Das Verhalten von call-by-reference wird in folgendem Code-Beispiel deutlich:

Beispiel: *by_reference.php*

```
<?php
①   function quadrat ($value) {
    echo "<p>Die Quadratzahl von $value ist: ";
    $value = $value * $value;
    echo $value . "</p>";
}
②   function quadrat_referenz (&$value) {
    echo "<p>Die Quadratzahl von $value ist: ";
    $value = $value * $value;
    echo $value . "</p>";
}
$zahl = 2;
echo '<p>Ausgangswert von $zahl: <strong>' . $zahl .
'</strong></p>';
echo "<em>call-by-value:</em>";
for ($i = 1; $i <= 3; $i++) {
    quadrat ($zahl);      // call-by-value
}
echo "<p><em>call-by-reference:</em></p>";
for ($i = 1; $i <= 3; $i++) {
    quadrat_referenz ($zahl);    // call-by-reference
}
?>
```

- ① Die Funktion `quadrat ()` wird definiert. Sie berechnet auf Basis des Übergabewertes `$zahl` die Quadratzahl und gibt sie aus. Diese Funktion arbeitet mit der Parameterübergabe als Wert (call-by-value).
- ② Die Funktion `quadrat_referenz ()` berechnet die Quadratzahl von `$zahl` auf exakt die gleiche Weise, arbeitet aber mit call-by-reference, d. h., Parameter werden als Referenz weiterverarbeitet. Die Übergabe per Referenz bewirken Sie durch das **&**-Zeichen im Funktionskopf vor der Variablen `$value`.

Beachten Sie auch, dass der Variablenname im Funktionsaufruf und im Funktionskopf nicht gleich lauten muss. Die Zuweisung der Variablen geschieht durch die Reihenfolge der Parameter, nicht durch deren Namen.

- ③ Die Funktion `quadrat()` wird in einer Schleife dreimal aufgerufen, die Quadratzahl von 2 wird berechnet (`call-by-value`) und ausgegeben. Die drei Funktionsaufrufe ergeben das gleiche Ergebnis 4. Der Wert der Variablen `$zahl` außerhalb der Funktion bleibt unverändert 2. Innerhalb der Funktion wird nur mit einer Kopie der Variablen gearbeitet.
- ④ Die Funktion `quadrat_referenz()` arbeitet hingegen per `call-by-reference` durch das im Funktionskopf verwendete `&`-Zeichen. Hier wird auch die Variable `$zahl` außerhalb der Funktion durch die Berechnung verändert. Da die Funktion auch hier dreimal aufgerufen wird, sehen Sie die Auswirkungen im Ergebnis des Funktionsaufrufs (4, 16, 256).

```
Ausgangswert von $zahl: 2
call-by-value:
Die Quadratzahl von 2 ist: 4
Die Quadratzahl von 2 ist: 4
Die Quadratzahl von 2 ist: 4
call-by-reference:
Die Quadratzahl von 2 ist: 4
Die Quadratzahl von 4 ist: 16
Die Quadratzahl von 16 ist: 256
```

Anzeige der Beispieldatei „by_reference.php“

Rückgabewert per `return`-Anweisung

Über die `return`-Anweisung haben Sie die Möglichkeit, einen Wert an den Funktionsaufruf zurückzugeben, der dort gespeichert und weiterverarbeitet werden kann. Der Rückgabewert kann z. B. das Ergebnis einer Berechnung sein oder ein `TRUE` oder `FALSE`, falls eine Funktion eine bestimmte Überprüfung durchführt.

Dabei ist zu beachten, dass eine `return`-Anweisung nicht nur den Rückgabewert an die aufrufende Stelle zurückgibt, sondern auch die Ausführung der Funktion **sofort** beendet. Alle Anweisungen nach einer `return`-Anweisung werden nicht mehr ausgeführt. `return` kann auch ohne einen konkreten Rückgabewert verwendet werden. Auch dies beendet sofort die Abarbeitung der Funktion.

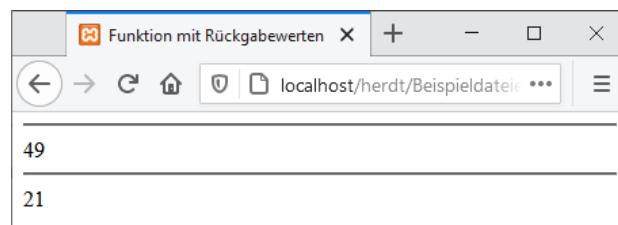
Beispiel: `funktionen-4.php`

```
<?php
    ① function addiere($zahl1, $zahl2) {
        ②     $ergebnis = $zahl1 + $zahl2;
        ③     return $ergebnis;
    }
    ④ addiere(4, 3); // nur Ausführung der Funktion, keine
                      // Ausgabe auf dem Bildschirm
    echo "<hr>";
```

```

⑤ $summe = addiere(30, 19); // der Rückgabewert wird in einer
                           // Variablen gespeichert
    echo $summe;
    echo "<hr>";
    echo addiere(17, 4); // direkte Ausgabe des Rückgabewertes
?>
```

- ① Die Funktion `addiere()` hat die beiden Parameter `$zahl1` und `$zahl2`.
- ② Die beiden Parameter werden innerhalb der Funktion addiert und das Ergebnis in der Variablen `$ergebnis` gespeichert.
- ③ Der Wert der Variablen `$ergebnis` wird mit der Anweisung `return` an die aufrufende Stelle zurückgegeben.
- ④ Die Funktion wird mit den Werten 4 und 3 aufgerufen. Die Funktion wird ausgeführt. Das Ergebnis 7 wird zurückgegeben, vom Funktionsaufruf jedoch nicht weiter verwendet. Das PHP-Skript würde zwar keine Fehlermeldung auswerfen, ein solcher Funktionsaufruf ohne irgendeine Ausgabe oder Weiterverarbeitung des Ergebnisses wäre jedoch als Programmierfehler zu bewerten.
- ⑤ Das Ergebnis eines weiteren Funktionsaufrufs 49 wird der Variablen `$summe` zugewiesen und in der folgenden Programmzeile ausgegeben. Das Speichern des Rückgabewertes einer Funktion in einer Variablen wird häufig verwendet, wenn mit diesem Wert weitere Operationen durchgeführt werden sollen.
- ⑥ Der Funktionsaufruf gibt das Ergebnis 21 zurück. Das Ergebnis wird durch die vorangestellte `echo`-Anweisung sofort ausgegeben. Diese Methode wird oft verwendet, wenn eine Ausgabe, aber keine Weiterverarbeitung des Wertes vorgenommen werden soll.



Anzeige der Datei „funktionen-4.php“

Eine Funktion in PHP kann immer nur **einen** Wert zurückgeben. Falls Sie **mehrere Werte** zurückgeben möchten, speichern Sie diese in einem Array und geben dann das Array zurück. Im Hauptskript können Sie dann auf die einzelnen Array-Einträge zugreifen. In den Code-Beispielen finden Sie die Datei `multi_return.php`, in der Sie eine beispielhafte Umsetzung finden.

Datentyp-Definition in Funktionen verwenden

Sie können im Funktionskopf sowohl den Datentyp für die Werte, welche die Funktion erwartet, als auch für den Wert, der zurückgegeben wird, festlegen. Das reduziert zwar die hohe Fehlertoleranz von PHP (im Vergleich zu anderen Programmiersprachen), Sie haben jedoch eine größere Kontrolle über Ihr PHP-Skript. Wird eine Funktion mit einem Wert mit einem falschen Datentyp aufgerufen, wirft PHP eine Fehlermeldung aus. Bei der Entwicklung entdecken Sie so potenzielle Fehlerquellen.

Beispiel: `datentypen.php`

```

① <?php
    function dividiere(int $zahl1, int $zahl2):int {
```

```

③ echo "<p>Wert von \$zahl1: $zahl1 | "
      . gettype($zahl1) . "</p>";
echo "<p>Wert von \$zahl2: $zahl2 | "
      . gettype($zahl2) . "</p>";
④ $ergebnis = $zahl1 / $zahl2;
echo "<p>Wert von \$ergebnis: $ergebnis</p>";
⑤ return $ergebnis;
}
② $rueckgabe = dividiere(10.5, 15);
// normaler Aufruf der Funktion
⑥ echo "<p>Wert von \$rueckgabe: $rueckgabe</p>";
echo "<p>Datentyp von \$rueckgabe: ".
      gettype($rueckgabe) . "</p>";
?>

```

- ① Hier wird eine Funktion definiert. Den Funktionsparametern ist das Schlüsselwort `int` vorangestellt. Damit legen Sie fest, dass die Funktion nur Werte vom Datentyp `int` verwendet bzw. übergebene Werte in Integer umwandelt. Hinter der schließenden Klammer `)` der Parameterdeklaration ist mit einem vorangestellten Doppelpunkt `:` das Schlüsselwort `int` angegeben. Damit legen Sie fest, dass die Funktion ausschließlich einen Integer zurückgibt bzw. dass der Rückgabewert versucht wird, in einen Integer zu wandeln.
- ② Die Funktion `dividiere()` wird hier mit zwei Parametern aufgerufen. Dabei hat der erste Wert den Datentyp `float`, der zweite den Datentyp `int`. Die implizierte Datenumwandlung ermittelt den Integer 10 aus dem übergebenen Float-Parameter 10.5. Würde hier in String-Parameter übergeben, könnte PHP den Wert nicht in ein Integer umwandeln, es käme zu einem *fatal error* und zum Abbruch des Skripts.
- ③ Hier werden die übergebenen Parameter auf dem Bildschirm angezeigt. PHP hat den Float-Wert 10.5 umgewandelt. Zur Verdeutlichung wird hier per `gettype()` der Datentyp ausgegeben.
- ④ Hier wird die Division der beiden Werte durchgeführt und ausgegeben. Das Ergebnis 0.6666666666667 ist ein Wert vom Datentyp `float`.
- ⑤ Das Ergebnis wird zurückgegeben. Hier greift die Datentyp-Festlegung für den Rückgabewert im Funktionskopf. PHP konvertiert den Wert für die Rückgabe in einen Integer. Ohne mögliche Datentypkonvertierung käme es zu einem *fatal error*. Die Datentypkonvertierung wäre z. B. nicht möglich, wenn Sie den Rückgabewert mit `array` festgelegt hätten.
- ⑥ In der Kontrollausgabe können Sie das Ergebnis aus der Datentypkonvertierung des Rückgabewertes beobachten. Es wird lediglich die Zahl 0 ausgegeben.
- Der Datentyp wird zusätzlich über die PHP-Funktion `gettype()` ausgegeben. Tatsächlich liefert hier PHP einen Integer zurück.

Ausgabe der Datei „datentypen.php“

Union Types für die Datentyp-Definition verwenden

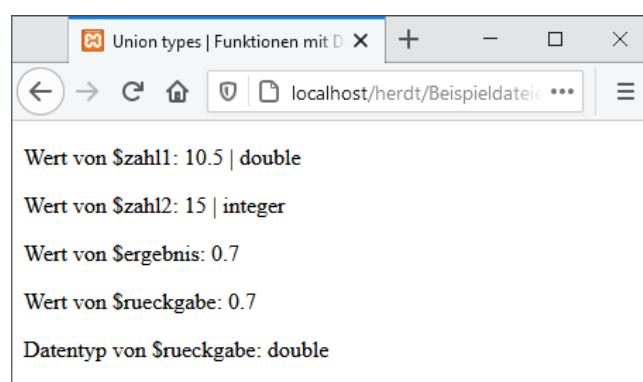
Neu in PHP 8.0 sind Union Types. Bislang konnten Sie die Datentyp-Definition für die Funktionsparameter oder einen Rückgabewert nur mit einem einzelnen Datentyp belegen. Nun können Sie eine Gruppe von Datentypen angeben. Dabei geben Sie die bekannten Datentypen wie `int`, `float`, `string` usw. getrennt mit einer Pipe-Zeichen `|` an, z. B. `int | float`.

Beispiel: `union-types.php`

```
<?php
①    function rechne(int|float $zahl1,
                  int|float $zahl2): int|float {
    echo "<p>Wert von \$zahl1: $zahl1 | " . gettype($zahl1)
. "</p>";
    echo "<p>Wert von \$zahl2: $zahl2 | " . gettype($zahl2)
. "</p>";
    $ergebnis = $zahl1 / $zahl2;
    echo "<p>Wert von \$ergebnis: $ergebnis</p>";
    return $ergebnis;
}
② $rueckgabe = rechne(10.5, 15); // normaler Aufruf der Funktion
echo "<p>Wert von \$rueckgabe: $rueckgabe</p>";
echo "<p>Datentyp von \$rueckgabe: " . gettype($rueckgabe)
. "</p>";
?>
```

- ① Dieser Quelltext entspricht weitestgehend dem Code von `datentypen.php`. Hier sind jedoch zur Datentyp-Deklaration Union Types eingesetzt worden: `int | float`.
- ② Die Funktion wird mit dem `float`-Wert `10.5` und dem `int`-Wert `15` aufgerufen. Da im Funktionskopf über die Union-Type-Angabe `int | float` die möglichen Datentypen für beide Parameter definiert wurden, werden beide Werte mit ihrem Datentyp akzeptiert.

In der Abbildung ist nun zu sehen, dass die Zahlen in der Funktion auch den korrekten Datentyp haben. Auch der Rückgabewert wird nicht, wie im Beispiel zuvor, zu einem Integer umgewandelt, sondern auch als `float` zurückgegeben, da der Rückgabewert per Union Type `:int | float` hinter dem Funktionskopf festgelegt wurde.



Ausgabe der Beispieldatei „union-types.php“

Variadische Funktionen (variadic functions) mit ... verwenden

Variadisch beschreibt die Eigenschaft von Funktionen, dass die Anzahl der Parameter in der Funktionsdeklaration (Funktionskopf) nicht zwingend festgelegt werden müssen. Selbst wenn Sie eine Funktion mit drei Parametern deklariert haben, können Sie fünf Parameter übergeben, ohne dass dies zum PHP-Fehler führt. Über die Funktion `func_get_args()` können nicht deklarierte Variablen ermittelt werden.

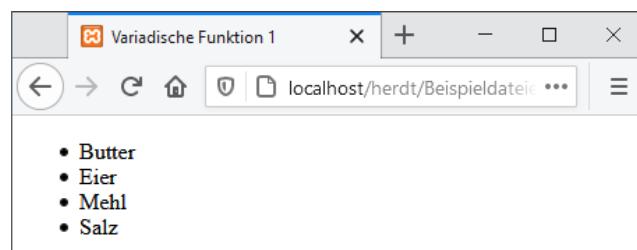
Über den Einsatz des Zeichens für die Auslassungspunkte ... (drei Punkte, auch Ellipse oder Splat-Operator genannt) können Funktionen jetzt mit einer beliebigen Anzahl von Parametern aufgerufen werden. Ein Parameter im Funktionskopf, dem die Auslassungszeichen vorangestellt sind, z. B. `function (...$args)`, nimmt alle Übergabeparameter in einem Array auf. Aber auch beim Aufruf einer Funktion können die Auslassungspunkte verwendet werden, z. B. `meinFunktionsAufruf(...$param)`. Hier kann die Funktion mit einem Array aufgerufen werden, im Funktionskopf werden dann die einzelnen Array-Einträge auf die dort deklarierten Parameter verteilt.

Variadische Parameter verwenden

Beispiel: `variadic-1.php` (Auszug)

```
<?php
    // Funktion mit variadischem Parameter
    ① function zeigeZutaten(...$args) {
        echo "<ul>";
        ② foreach ($args as $val) {
            echo "<li>$val</li>";
        }
        echo "</ul>";
    }
    ③ zeigeZutaten("Butter", "Eier", "Mehl", "Salz");
?>
```

- ① Die Funktion `zeigeZutaten()` wird aufgebaut. Als Parameter wird die Variable `$args` deklariert. Dieser Variablen wird das Auslassungszeichen ... vorangestellt. Damit werden alle vier übergebenen Parameter dem Array `$args` zugewiesen.



Ausgabe der Beispieldatei „variadic-1.php“

- ② Mit einer `foreach`-Schleife wird das Array durchlaufen und jeder einzelne Array-Eintrag als Listen-Eintrag ausgegeben.
- ③ Im Funktionsaufruf `zeigeZutaten()` werden vier Parameter übergeben. Hier könnten beliebig viele Parameter übergeben werden, wobei alle der Funktionsvariablen `$args` als Array zugewiesen werden.

In der Beispieldatei `variadic-1.php` finden Sie eine zweite Funktion: `zeigeZutatenAlt()`. Dort wird die Alternative gezeigt: Dort ist kein Parameter im Funktionskopf deklariert. Der Funktionsaufruf findet ebenfalls mit vier Parametern statt, diese werden in der Funktion über `func_get_args()` ermittelt und im Array `$args` gespeichert. Die weitere Verarbeitung ist mit dem PHP-Code oben identisch.

Bei der Verwendung von variadischen Parametern ist Folgendes zu beachten:

- ✓ Das Auslassungszeichen `...` wird einer Variablen vorangestellt (mit oder ohne Leerzeichen vor der Variablen).
- ✓ Die Variable mit dem Auslassungszeichen `...` **muss** als **letzter** Parameter im Funktionskopf deklariert werden. Dieser wird von den anderen Parametern mit einem Komma getrennt. Folgen nach der Variablen für den variadischen Parameter weitere Variablen, meldet PHP einen Fehler vom Typ *fatal error*.
- ✓ Ein variadischer Parameter kann **nicht** mit einem optionalen Parameter deklariert werden.
- ✓ Die Variable kann nur per call-by-value übergeben werden. Die Übergabe per call-by-reference führt ebenfalls zum *fatal error*.

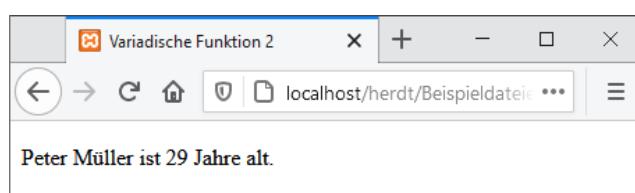
Übergabe einer Parameterliste

Im vorherigen Beispiel wird der Parameter in der Funktionsdeklaration durch das vorangestellte Auslassungszeichen `...` zum variadischen Parameter. In gleicher Weise kann eine Funktion aber auch auf variadische Weise aufgerufen werden.

Beispiel: `variadic-2.php` (Auszug)

```
<?php
    // Funktion mit Parameter für variable Anzahl von Parametern
①    function personInfo($name, $vorname, $alter) {
        echo "<p>$vorname $name ist $alter Jahre alt.</p>";
    }
③    $person = ['Müller', 'Peter', 29];
④    personInfo(...$person);
?>
```

- ① Die Funktion `personInfo()` wird aufgebaut. Die Parameterdeklaration im Funktionskopf entspricht dem Standardaufbau von Funktionen. Es werden drei Variablen deklariert, die jeweils einen Parameter erwarten.



Ausgabe der Beispieldatei „variadic-2.php“

- ② Die übergebenen Parameter werden als Zeichenkette auf dem Bildschirm ausgegeben.
 ③ Hier wird ein Array über die Kurzschreibweise definiert. Dieses enthält drei Einträge.
 ④ Die Funktion `personInfo()` wird mit dem zuvor erstellten Array aufgerufen. Der Array-Variablen wird das Auslassungszeichen `...` vorangestellt. Daran erkennt PHP, dass dieser Parameter nicht an einen einzelnen Parameter im Funktionskopf übergeben werden soll, sondern dass es sich um eine Parameterliste handelt, die auf die einzelnen Parameter im Funktionskopf verteilt werden soll.

Bei der Verwendung des Auslassungszeichens . . . zur Übergabe einer Parameterliste ist Folgendes zu beachten:

- ✓ Werden mehrere Parameter an die Funktion übergeben, darf die Übergabeliste **nur** als letzter Parameter übergeben werden.
- ✓ Dem Parameter wird das Auslassungszeichen . . . vorangestellt und von etwaigen davorstehenden Parametern mit einem Komma getrennt.
- ✓ Der Parameter erwartet ein indiziertes Array oder ein traversables Objekt (wird in diesem Buch nicht weiter vertieft). Eine Zeichenkette oder ein assoziatives Array führen zu einem *fatal error*.

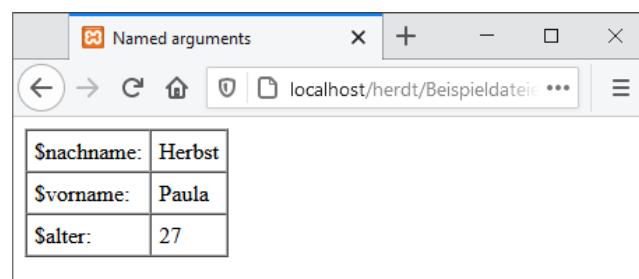
Benannte Parameter (Named Arguments) verwenden

Eine Neuerung in PHP 8.0 sind benannte Parameter. Standardmäßig werden die Parameter, die Sie einer Funktion übergeben, den Variablen im Funktionskopf in gleicher Reihenfolge zugewiesen, wie sie im Funktionsaufruf angegeben sind. Das Verhalten wird in folgendem Beispiel deutlich.

Beispiel: *named-arguments.php*

```
<?php
①  function zeigeDaten($nachname, $vorname, $salter) {
    echo "<table cellpadding=5 cellspacing=0 border=1>";
    echo "<tr><td>\$nachname:<td> $nachname</tr>";
    echo "<tr><td>\$vorname: <td>$vorname</tr>";
    echo "<tr><td>\$salter: <td>$salter</tr>";
    echo "</table>";
}
③  zeigeDaten(vorname: 'Paula', alter: 27, nachname: 'Herbst');
?>
```

- ① Hier wird eine Funktion geschrieben, die 3 Parameter \$nachname, \$vorname und \$salter erwartet.
- ② In einer einfachen HTML-Tabelle werden die Variablennamen sowie die Werte, welche ihnen zugewiesen wurden, ausgegeben.



Ausgabe der Beispieldatei „named-arguments.php“

- ③ Im Funktionsaufruf werden drei Parameter übergeben (hier jedoch jeweils mit dem Namen (ohne Anführungszeichen) der Variablen, wie er im Funktionskopf angegeben ist, gefolgt von einem Doppelpunkt [:]). Damit legen Sie fest, welcher Variablen im Funktionskopf der nachfolgende Wert zugewiesen werden soll.

Die Auswirkung können Sie in der Abbildung sehen: Obwohl als erster Parameter Paula übergeben wird, wird dieser Wert aufgrund des benannten Parameters der 2. Variable im Funktionskopf zugewiesen. Entsprechend die beiden anderen Werte.

Falls Sie einen Namen angeben, der keiner Variablen im Funktionskopf entspricht, erhalten Sie eine Fehlermeldung vom Typ ***fatal error***.



Die Frage, ob sich diese Neuerung in PHP 8.0 im täglichen Einsatz durchsetzen wird, kann man heute noch nicht beurteilen. Ein Vorteil ist, dass mit der Benennung der Variablen schon im Funktionsaufruf deutlich wird, welche Art von Wert erwartet wird. Kritisch ist jedoch zu betrachten, dass man die gewohnte Zuweisung der Reihenfolge im Funktionsaufruf verändern kann, was eine nicht zu unterschätzende Fehlerquelle sein kann.

7.3 Der Gültigkeitsbereich von Variablen

Variablen, die Sie außerhalb einer Funktion im PHP-Programm einsetzen, sind standardmäßig innerhalb von Funktionen nicht definiert, also dort unbekannt. Das Gleiche gilt für Variablen innerhalb von Funktionen, auf die Sie nicht im normalen Programmablauf zugreifen können.

Der Gültigkeitsbereich von Variablen (auch **Namespace** genannt) ergibt sich automatisch aus dem Zusammenhang, in dem sie definiert wurden.

- ✓ **Lokale Variablen** sind nur innerhalb der Funktion gültig, in der sie definiert wurden. Wurde die Variable im normalen Programmablauf definiert (außerhalb einer Funktion), steht sie **nicht** innerhalb von Funktionen zur Verfügung. Dieses Verhalten ist der Standard.
- ✓ Lokale Variablen können zu **globalen Variablen** werden, indem Sie innerhalb einer Funktion mit dem Schlüsselwort `global` bekannt machen, z. B. `global $zahl`. Damit heben Sie für diese Variable die Grenze zwischen Funktion und Programmablauf auf. Eine als `global` definierte Variable steht sowohl in der Funktion als auch außerhalb der Funktion zur Verfügung. Das bedeutet zugleich, dass Veränderungen an der Variablen, sowohl im Hauptskript als auch innerhalb einer Funktion, die Variable immer beeinflussen.
- ✓ **Superglobale Variablen** sind von PHP vordefinierte Variablen, die Ihnen an jeder Stelle des Skripts zur Verfügung stehen. Sie erkennen diese Variablen daran, dass sie mit den Zeichen `$_` beginnen und vollständig aus Großbuchstaben bestehen, z. B. `$_POST`. Superglobale Variablen können Sie nicht selbst definieren. Auch wenn Sie sich an die beschriebene Konvention zur Benennung dieser Variablen halten, erstellen Sie damit keine superglobalen Variablen.

Im folgenden Beispiel werden die verschiedenen Typen von Variablen gegenübergestellt. Hierzu wird ein Bestellformular für Äpfel erstellt. Anhand des PHP-Codes im Auswertungsprogramm wird gezeigt, wann welche Variablen gültig sind.

Beispiel: *formular_funktion.html*

```
<h1>Apfelkauf</h1>
<p>Bitte geben Sie die gewünschte Menge ein und wählen Sie
eine Apfelsorte:</p>
① <form action="funktion_var.php" method="post">
②   <p>Menge (in kg): <input type="text" name="menge"></p>
   <p>Apfelsorte:</p>
```

```

③ <input type="radio" name="sorte" value="Jonagold">
    Jonagold
<input type="radio" name="sorte" value="Gala"> Gala
<input type="radio" name="sorte" value="Elstar">
    Elstar</p>
<p><input type="submit" value="Abschicken"></p>
</form>

```

- ① Es wird ein Formular aufgebaut, welches die POST-Methode zum Versenden verwendet. Das Formular wird an das PHP-Skript *funktion_var.php* gesendet.
- ② Das input-Element mit dem name-Attribut *menge* wird programmiert, dieses kann vom PHP-Skript über die Variable `$_POST['menge']` abgefragt werden.
- ③ Das input-Element vom Typ radio erhält als name den Wert *sorte* und steht nach dem Absenden des Formulars als `$_POST['sorte']` zur Verfügung.



Anzeige Datei „formular_funktion.html“

Beispiel: *funktion_var.php*

```

<?php
①  error_reporting(E_ERROR | E_PARSE); // Keine notice-Fehler melden

②  $bestellnummer = "ABC-12345";
    $bearbeiter = "Maria Schulze";

⑤  function bestellung() {
    global $bestellnummer;           // globale Variable
    echo "<p>Bestellnummer: " . $bestellnummer . "<br>";
    echo "Bearbeiter: " . $bearbeiter . "<br>";
    // lokale Variable - funktioniert hier nicht!
    echo "Die von Ihnen eingegebene Menge: " . $_POST["menge"] . " "
        kg</p>";
    ⑩ switch ($_POST["sorte"]) {
        case "Jonagold":
            $preis = $_POST["menge"] * 1.50;
            break;
        case "Gala":
            $preis = $_POST["menge"] * 1.65;
            break;
        case "Elstar":
            $preis = $_POST["menge"] * 2.00;
            break;
    }
    ⑪ echo "<p>Ausgabe des Preises innerhalb der Funktion: $preis</p>";
}

```

```

③ echo "<p>Die von Ihnen gewählte Sorte: " . $_POST["sorte"] . "</p>";
④ bestellung();
⑫ echo "<p>Ausgabe des Preises außerhalb der Funktion: $preis</p>";
// lokale Variable der Funktion - funktioniert hier nicht!
?>

```

The screenshot shows two browser windows. The left window displays the output of the script while it is executing inside the `bestellung()` function. It shows undefined variables `$bestellnummer` and `$bearbeiter`. The right window shows the output after the function has completed, where these variables have been defined.

Anzeige des Beispielprogramms „funktion_var.php“

Links ist das Error-Reporting für *warning* deaktiviert. Rechts ist das Standard-Error-Reporting von PHP belassen, die *warning*-Meldungen weisen bereits darauf hin, an welchen Stellen die Variablen unbekannt sind.

- ① *warning*-Meldungen werden standardmäßig ausgegeben. In der rechten Abbildung wird durch die Fehlermeldungen bereits deutlich, wo welche Variablen nicht bekannt sind.
Um die Anzeige in der linken Abbildung zu erhalten, sind Fehlermeldungen per `error_reporting (E_ERROR | E_PARSE)` auf schwerwiegende Fehler reduziert.
- ② Die Variablen `$bestellnummer` und `$bearbeiter` werden mit einem Wert belegt.
- ③ Der Wert des **superglobalen Arrays** `$_POST ["sorte"]` wird außerhalb der Funktion `bestellung()` ausgegeben.
- ④ Die Funktion `bestellung()` wird aufgerufen. Da Funktionsblöcke erst durchlaufen werden, wenn die Funktion aus dem Skript aufgerufen wird, werden die Anweisungen innerhalb der Funktion erst ausgeführt, wenn `bestellung()` ⑤ aufgerufen wurde.
- ⑥ Die Variable `$bestellnummer` wird innerhalb der Funktion `bestellung()` ⑤ mithilfe des Schlüsselwortes `global` bekannt gemacht.
- ⑦+⑧ Die Variablen `$bestellnummer` und `$bearbeiter` sollen innerhalb der Funktion ausgegeben werden. Da die Variable `$bearbeiter` innerhalb der Funktion **nicht** mithilfe des Schlüsselwortes `global` bekannt gemacht wurde, wird sie bei der Ausgabe nicht angezeigt ⑧. Sie gilt als nicht definiert.
- ⑨ Der Wert des **superglobalen Arrays** `$_POST ["menge"]` wird innerhalb der Funktion `bestellung()` ausgegeben.
- ⑩ Innerhalb der Funktion `bestellung()` wird der Wert der **lokalen Variablen** `$preis` per `switch`-Anweisung in Abhängigkeit von der Apfelsorte berechnet.
- ⑪ Die **lokale Variable** `$preis` wird innerhalb der Funktion ausgegeben.
- ⑫ Die **lokale Variable** `$preis` ⑪ soll auch außerhalb der Funktion ausgegeben werden. Dies funktioniert allerdings nicht, da es sich um eine lokale Variable handelt, die innerhalb dieser Funktion definiert wurde, aber nicht mit dem Schlüsselwort `global` bekannt gemacht wurde.

7.4 PHP-Dateien einbinden mit `include()` und `require()`

Bislang wurde in allen Übungen mit einer einzelnen PHP-Datei gearbeitet. Wie Sie gelesen haben, optimieren Funktionen den Quelltext dadurch, dass wiederkehrende Aufgaben in Funktionen gekapselt werden und von beliebigen Stellen im Skript aufgerufen werden können. Ein weiterer Schritt, PHP-Code zu verbessern, ist, Funktionen in separate Dateien auszulagern, damit diese nicht nur in einer Datei, sondern von beliebig vielen Dateien verwendet werden können.

Um ausgelagerte Funktionen verwenden zu können, muss die Datei mit den darin enthaltenen Funktionen mit der eigentlichen Skript-Datei verknüpft werden. Mithilfe der `include()`- bzw. `require()`-Anweisung können Sie ausgelagerte PHP-Abschnitte, wie z. B. eigene PHP-Funktionen, in Ihr PHP-Programm einbinden.

Mit `include()` und `require()` arbeiten

Die `include()`- bzw. `require()`-Anweisung bindet eine Datei, deren Name (einschließlich relativer oder absoluter Pfadangabe) als Argument übergeben wird, in den aktuellen Programmcode ein.

Unterschiede zwischen `include()` und `require()`

Beide Anweisungen funktionieren ähnlich. Ein wichtiger Unterschied zwischen den beiden Anweisungen zeigt sich, wenn eine Datei fehlerhaft eingebunden wurde:

- ✓ Falls die angegebene Datei nicht vorhanden ist, führt `include()` zu einer Warnung. Das Skript wird aber weiter ausgeführt. Sollte in der eingebundenen Datei kein PHP-Code sein, der für die weitere Abarbeitung des Hauptskripts notwendig ist, kann dieses auch mit einem fehlerhaften `include()` bis zum Ende laufen.
- ✓ Der Befehl `require()` beendet bei einer fehlenden Datei das Skript sofort mit einer Fehlermeldung.

Syntax und Bedeutung der `include()`- und `require()`-Anweisungen

- ✓ Die `include()`- und `require()`-Befehle erwarten als Parameter die einzubindende Datei. Befindet sich die Datei nicht im selben Ordner, muss der Pfad angegeben werden.
- ✓ Der Aufruf erfordert nicht zwingend die runden Klammern. Dateiname und Pfad können auch in Anführungszeichen hinter dem Schlüsselwort `include` bzw. `require` angegeben werden.
- ✓ Beinhaltet die eingebundene Datei keinen PHP-Code, wird der Inhalt unverändert an den Browser weitergegeben und somit am Bildschirm dargestellt.

```
include("Dateiname");
require("Dateiname");
include ("Pfad/Dateiname");
require ("Pfad/Dateiname");
include "Dateiname";
require "Dateiname";
```

- ✓ Auch Ausgaben per `echo` oder `print_r` in einer eingebundenen Datei werden, soweit keine anderen Maßnahmen ergriffen wurden, direkt im Browser ausgegeben.
- ✓ Eingebundene Dateien, die per `return` einen Rückgabewert liefern, können wie Funktionen mit Rückgabewert behandelt werden. Entweder wird der Rückgabewert direkt per `echo include ("Dateiname")` ausgegeben oder er wird in einer Variablen gespeichert: `$variable = include ("Dateiname")` und kann dann per `echo $variable` ausgegeben werden.
- ✓ Jede Funktionsdefinition in eingebundenen Dateien steht in der aufrufenden Datei zur Verfügung. Die Vorstellung, die eingebundene Datei wird in die aufrufende Datei an die Stelle des Aufrufs „hineinkopiert“, erleichtert das Verständnis der Funktionsweise.

! Wenn `include()` oder `require()` ausgeführt werden, wird die Abarbeitung des Hauptskripts so lange unterbrochen, bis die fremde Datei geladen und verarbeitet wurde. Aus diesem Grund muss die eingebundene Datei vollständig gültiger PHP-Code sein. Ein Leerzeichen oder HTML-Elemente vor dem öffnenden `<?php` oder hinter dem schließenden PHP-Tag `?>` führt oftmals zum Fehler. Als gängige Konvention hat sich deswegen etabliert, das schließende `?>` einfach wegzulassen. Damit vermeiden Sie eventuelle Leerzeichen am Ende der PHP-Datei.

Skripte, die in einen PHP-Code eingebunden werden sollen, können mit der Dateierweiterung `.inc` abgespeichert werden. Je nach Serverkonfiguration wird bei `.inc`-Dateien der PHP-Code jedoch nicht geparsst, sondern der Inhalt direkt im Browser ausgegeben, wenn die Datei direkt im Browser aufgerufen wird, was je nach Inhalt eine Sicherheitslücke darstellen kann. Erst wenn eine `.inc`-Datei in PHP eingebunden ist, wird sie regulär wie PHP geparsst.

Alternativ bietet sich die Dateiendung `.inc.php` an. Daran können Sie erkennen, dass eine solche Datei zur Einbindung vorgesehen ist, sie wird aber beim direkten Aufruf im Browser geparsst. Eine ungewollte Ausgabe des PHP-Codes im Browser wird so vermieden.

Nur einmal laden: `include_once()` und `require_once()`

Die Funktionen `include_once()` und `require_once()` entsprechen im Wesentlichen den Funktionen `include()` und `require()`. PHP prüft allerdings, ob die angegebene Datei bereits eingebunden wurde. Ist das korrekt, wird die Datei nicht nochmals eingebunden.

Durch den Einsatz dieser Funktionen können Sie Fehler vermeiden, die z. B. durch Mehrfach-deklaration namengleicher Funktionen oder nochmalige Wertzuweisungen von Variablen hervorgerufen werden.

Datei mit `require()` einbinden

Sie erstellen eine selbst definierte Funktion `summe()` in einer separaten Datei. Außer der Funktion enthält diese Datei keinen weiteren PHP-Code. Die Datei wird dann von einer zweiten Datei über die `require()`-Anweisung eingebunden, sodass auch dort die Funktion `summe()` zur Verfügung steht.

Beispiel: *require.php*

```
<h1>require: Verwenden einer Funktion aus einer anderen Datei</h1>
<?php
    echo "<p>Ein fremdes PHP-Skript wird eingebunden und die dort
        enthaltene Funktion ausgeführt:</p>";
①    require ("eigene_funktion.inc.php");
    summe(2000, 1376);
    echo "<hr><p>Hier kann das Skript weitere Anweisungen
        enthalten.</p>";
?
>
```

- ① Die `require()`-Anweisung bindet die Datei *eigene_funktion.inc.php* ein.

Datei *eigene_funktion.inc.php*

```
<?php
function summe($zahl1, $zahl2) {
    $ergebnis = $zahl1 + $zahl2;
    echo "<p>Summenbildung: $zahl1 + $zahl2 = " . $ergebnis . "</p>";
}
```

Innerhalb der Datei *eigene_funktion.inc.php* wurde die Funktion `summe()` definiert, die nach dem Einbinden in der aufrufenden Datei zur Verfügung steht.



Damit die Funktion `summe()` in der ausgelagerten Datei zur Verfügung steht, muss die Datei vor dem Funktionsaufruf eingebunden werden. Wird `require()` erst hinter dem Funktionsaufruf verwendet, kennt PHP den ausgelagerten Code noch nicht und es kommt zum Fehler.

require: Verwenden einer Funktion aus einer anderen Datei

Ein fremdes PHP-Skript wird eingebunden und die dort enthaltene Funktion ausgeführt:

Summenbildung: 2000 + 1376 = 3376

Hier kann das Skript weitere Anweisungen enthalten.

Anzeige der Beispieldatei „*require.php*“

Das Einbinden von Dateien per `include` bzw. `require` ist nicht auf PHP-Dateien beschränkt. Sie können z. B. auch `*.txt` oder `*.html`-Dateien einbinden. In den Beispieldateien finden Sie die Datei *include.php*, welche die Datei *einfuegen.txt* einbindet. Das Prinzip und die Syntax sind die gleichen wie bei der Einbindung von PHP-Dateien. Allerdings wird Text bzw. HTML lediglich ausgegeben und nicht von PHP geparsst.

7.5 Übungen

Übung 1: Mit Funktionen arbeiten

Level		Zeit	ca. 10 min
Übungsinhalte	<ul style="list-style-type: none"> ✓ PHP-Funktionen ✓ Optionale Parameter 		
Übungsdatei	--		
Ergebnisdatei	<i>funktion_uebung.php</i>		

1. Erstellen Sie eine PHP-Datei *funktion_uebung.php*, in der Sie zwei Funktionen definieren: `addiere()` und `multipliziere()`. Beide Funktionen sollen Berechnungen gemäß ihren Namen durchführen: Zahlen addieren bzw. multiplizieren.
2. In den Funktionsaufrufen sollen zwei Parameter eingegeben werden müssen, ein definerter dritter Parameter ist optional. (Überlegen Sie, welche Standardwerte Sie für den dritten Parameter vergeben müssen, damit die Funktionen richtige Berechnungen durchführen.) Eine Ausgabe des Ergebnisses soll zusammen mit einem kurzen Text, welche Berechnung mit welchen Zahlen vorgenommen wurde, direkt in den Funktionen erfolgen.
3. Erweitern Sie Ihr Programm um vier Funktionsaufrufe:
 Funktionsaufrufe 1 und 2: `addiere()`, `multipliziere()` mit jeweils drei Parametern: 8, 4 und 2.
 Funktionsaufrufe 3 und 4: `addiere()`, `multipliziere()` mit jeweils zwei Parametern: 8 und 4.



Anzeige der Beispiellösung „funktion_uebung.php“

Übung 2: Funktionen über ein Formular aufrufen

Level		Zeit	ca. 10 min
Übungsinhalte	<ul style="list-style-type: none"> ✓ HTML-Formular ✓ PHP-Funktionen ✓ Einbinden von Dateien per <code>include()</code> ✓ Verwenden der globalen Variable <code>\$_POST</code> 		
Übungsdatei	--		
Ergebnisdateien	<code>funktion.inc.php, form_uebung.php</code>		

1. Kopieren Sie die Funktionsdefinition `addiere()` und speichern Sie diese in eine andere Datei mit dem Namen `funktion.inc.php`.
2. Erstellen Sie ein Formular mit dem Dateinamen `form_uebung.php` mit Eingabefeldern für drei Zahlen und Schaltflächen zum Absenden und Zurücksetzen des Formulars.
3. Die Datei soll sich beim Absenden des Formulars selbst aufrufen. Die Funktion `addiere()` soll in die Datei `form_uebung.php` eingebunden und von hier aus aufgerufen werden, nachdem das Formular versendet wurde.

Übung: Addition mit eingebundener Funktion

Bitte geben Sie zwei oder drei Zahlen ein und senden Sie das Formular ab:

Zahl 1:

Zahl 2:

Zahl 3 (optional):

Absenden

Das Formular wurde abgesendet, das Ergebnis der ...

Addition (Zahlen 45, 317, 512) lautet 874

Anzeige der Beispiellösung „form_uebung.php“

8

Mit Daten aus externen Dateien arbeiten

 **Beispieldateien:** Dateien im Ordner *Kapitel_08*

8.1 Externe Dateien nutzen

Beispiel für die Nutzung von Daten aus externen Dateien

Wenn Sie Daten, welche die Benutzer in ein Formular eingegeben haben, z. B. in einer Excel-Tabelle auswerten oder bearbeiten möchten, können Sie diese außerhalb des PHP-Skripts in einer externen Datei speichern. Über PHP haben Sie die Möglichkeit, externe Dateien zu öffnen, Inhalte auszulesen, aber auch Inhalte in Dateien zu schreiben oder zu ersetzen.

Beispiele für die sinnvolle Nutzung von externen Dateien sind:

- ✓ Protokolldateien schreiben:
Sie können Daten in einer eigenen Datei speichern, um z. B. die Anzahl der Aufrufe einer Webseite zu protokollieren.
- ✓ Dynamische Daten in eine Webseite einbauen:
Sie können beispielsweise einen Hinweis auf das Angebot des Tages in einer externen Textdatei speichern. So brauchen Sie, um die Daten zu verändern, nur die externe Datei zu bearbeiten.
- ✓ Sie können Eingaben aus Formularen speichern, falls Sie keine Datenbank einsetzen können bzw. wollen.

Wichtig bei der Bearbeitung von externen Dateien ist die Art der Datei. PHP kann zwischen dem Zugriff auf Textdateien und binäre Dateien unterscheiden. Textdateien (*.txt) oder Textdateien im Tabellenformat (*.csv) beinhalten Zeilen unterschiedlicher Länge. Sie werden sowohl beim Schreiben als auch beim Lesen zeilenweise bearbeitet. Im Rahmen der PHP-Grundlagen erlernen Sie in diesem Kapitel das Arbeiten mit Textdateien.

8.2 Dateien öffnen, lesen und schließen

Der Zugriff auf externe Dateien besteht immer aus den gleichen Schritten:

- ✓ Datei öffnen,
- ✓ Datei bearbeiten, also Inhalte auslesen oder Inhalte in die Datei schreiben,
- ✓ Datei schließen.

PHP bietet zu diesem Zweck unterschiedliche Funktionen an, welche entweder einen einzelnen dieser Schritte abarbeiten oder alle drei Schritte in einem bewerkstelligen.

Dateien mit `fopen()` öffnen

Bevor Sie auf die Inhalte in Dateien zugreifen können, müssen Sie die entsprechende Datei öffnen. Dazu können Sie die Funktion `fopen()` verwenden.

Syntax und Bedeutung der Funktion `fopen()`

```
fopen("Pfad/Dateiname", "Modus");
$handle = fopen("user.txt", "r")
```

`fopen()` stellt einen Stream (Datenstrom, sozusagen eine Verbindung) zu der angegebenen Datei ① her. Der Rückgabewert von `fopen()` ist ein sogenanntes **Handle** (engl.: Griff oder Henkel) und hat den Datentyp *resource*. Dieses Handle benötigen Sie, um weitere Aktionen mit der Datei durchzuführen. Dieser Rückgabewert wird in der Variablen `$handle` ② gespeichert. Über das Handle steuern Sie auch den Dateizeiger. Dieser bestimmt die Position in der Datei, von der aus Sie Daten lesen oder schreiben.

Voraussetzung für einen erfolgreichen Dateizugriff ist der korrekte Pfad zu der Datei sowie passende Zugriffsrechte auf die Datei. Scheitert das Öffnen der Datei, liefert `fopen()` den Wert `FALSE` zurück.

Modus zum Öffnen von Dateien

Als zweiten Parameter erwartet `fopen()` einen Modus ③. Über den Modus steuern Sie, zu welchem Zweck die Datei geöffnet werden soll, z. B. zum Lesen oder Schreiben. Darüber hinaus steuern Sie über den Modus den Dateizeiger bzw. an welcher Stelle in der Datei dieser steht. Fehlt dieser Modus-„Schalter“, liefert `fopen()` `FALSE` zurück, der Aufbau des Streams ist dann fehlgeschlagen. PHP gibt eine Fehlermeldung vom Typ *warning* aus.

In PHP stehen folgende Modi für `fopen()` zur Verfügung:

Modus	Beschreibung
'r'	Die Datei wird nur zum Lesen (<i>r = read</i>) geöffnet, der Dateizeiger wird auf den Anfang der Datei positioniert.
'r+'	Die Datei wird zum Lesen und Schreiben geöffnet, der Dateizeiger wird auf den Anfang der Datei gesetzt.
'w'	Öffnet eine Datei nur zum Schreiben (<i>w = write</i>). Der Dateizeiger wird auf den Anfang der Datei gesetzt und die Länge der Datei auf 0 Byte. Eventuell vorhandene Daten werden gelöscht. Existiert die Datei nicht, wird sie angelegt.
'w+'	Hier handelt es sich um dieselbe Option wie die Option 'w', nur wird hier die Datei zum Lesen und Schreiben geöffnet.
'a'	Die Datei wird nur zum Schreiben geöffnet, der Dateizeiger wird auf das Ende der Datei gesetzt. Neue Daten ergänzen vorhandene Daten (<i>a = append</i>).
'a+'	Wie Option 'a', jedoch wird die Datei zum Lesen und Schreiben geöffnet.
'x'	Erzeugt und öffnet eine Datei zum Schreiben und setzt den Dateizeiger auf den Dateianfang. Falls die Datei bereits vorhanden ist, liefert <code>fopen()</code> FALSE zurück.
'x+'	Erzeugt und öffnet eine Datei zum Lesen und Schreiben , sonst wie 'x'.
'c'	Öffnet eine Datei zum Schreiben , falls die Datei nicht existiert, wird sie erzeugt. Wenn die Datei existiert, wird sie im Gegensatz zu 'w' nicht gekürzt, vorhandene Daten bleiben erhalten. Der Dateizeiger wird auf den Dateianfang gesetzt.
'c+'	Öffnet die Datei zum Lesen und Schreiben , ansonsten wie 'c'.

Prüfung auf Existenz einer Datei mit `file_exists()`

Bevor Sie eine Datei öffnen, sollten Sie stets prüfen, ob die Datei überhaupt vorhanden ist. Ein Lese-Zugriff per `fopen()` sollte erst gar nicht durchgeführt werden, wenn die Datei nicht vorhanden ist. Bei einem beabsichtigten Schreib-Zugriff sollte dann ein Modus gewählt werden, der das Anlegen der Datei automatisch durchführt. Zu diesem Zweck steht Ihnen `file_exists()` zur Verfügung.

Syntax und Bedeutung der Funktion `file_exists()`

Die Funktion liefert im Erfolgsfall TRUE zurück. Dabei unterscheidet `file_exists(Name);` `file_exists()` nicht nach Dateien oder Verzeichnissen.

Auf Datei mit `is_file()` prüfen

Noch differenzierter prüfen Sie die zu öffnende Datei mit `is_file()`, ob es sich überhaupt um eine Datei handelt. Mit `file_exists()` können Sie schlimmstenfalls doch in einen Fehler laufen, wenn Sie ein gleichlautendes Verzeichnis finden und versuchen, dieses als Datei zu öffnen. Grundsätzlich sollten Sie so konkret wie möglich prüfen, um Fehler gezielt abfangen zu können. Damit verbessern Sie die Qualität Ihres Skriptes.

Syntax und Bedeutung der Funktion `is_file()`

Prüft, ob es sich bei Dateiname um eine Datei handelt. Falls ja, liefert die Funktion TRUE zurück, anderenfalls FALSE.

```
is_file("Dateiname");
```

`is_dir("Ordnername")` ist die entsprechende Funktion, um zu prüfen, ob es sich bei Ordnername um ein Verzeichnis handelt.

Dateien mit `fgets()` lesen

Mit der Funktion `fgets()` können Sie eine Zeile einer Datei auslesen. Sie wird von der Position des Dateizeigers aus gelesen, welche Sie zuvor über den Modus bei `fopen()` bestimmt haben.

Syntax und Bedeutung der Funktion `fgets()`

- ✓ Als ersten Parameter erwartet `fgets()` das Handle, das beim Öffnen der Datei per `fopen()` zurückgeliefert wurde. Im Beispiel wurde das Handle in der Variablen `$handle` ① gespeichert.
- ✓ `fgets()` liest eine Zeile aus einer Datei aus: Eine Zeile entspricht den Daten einer Zeile bis zum ersten Zeilenumbruch `\n`.
- ✓ Geben Sie den Parameter Länge an, wird die Zeile nur bis zu der angegebenen Länge gelesen.
- ✓ `fgets()` liest keine weiteren Daten ein, wenn das Ende der Datei (EOF = *End of File*) erreicht ist.
- ✓ Jeder Aufruf von `fgets()` bewegt den Dateizeiger eine Zeile weiter, bis das Ende der Datei erreicht ist. Wenn Sie `fgets()` erneut aufrufen, wird die nächste Zeile gelesen. Sollen Dateien komplett ausgelesen werden, wird dies in der Regel über eine `while()`-Schleife realisiert.
- ✓ Der Rückgabewert der Funktion `fgets()` ist eine Zeichenkette. Im Beispiel wird der Rückgabewert in der Variablen `$zeile` gespeichert.

```
fgets(Handle[, Länge]);  
$zeile = fgets($handle);
```

①

Steuerungszeichen für das Zeilenende

Die zeilenweise Abarbeitung von `fgets()` basiert darauf, dass das Ende einer Zeile auch erkannt wird. PHP erkennt das Zeilenende an dem Steuerungszeichen `\n`, welches auch beim Schreiben zum Erzeugen eines Zeilenumbruchs verwendet wird.

Allerdings haben die verschiedenen Betriebssysteme unterschiedliche Konventionen für das Zeilenende. Unix-basierte Systeme verwenden das einfache `\n`, Windows die Kombination von `\r\n` (`r` für *return*, `n` für *new line*), Mac-Systeme verwenden nur das `\r`. Gerade beim Schreiben von Dateien können Ergebnisdateien auf den jeweils anderen Systemen fehlerhaft dargestellt werden, wenn nicht das korrekte Zeichen eingesetzt wird.

Prüfung auf Dateiende mit `feof()`

Beim Auslesen von Dateiinhalten über eine `while()`-Schleife benötigen Sie für den Schleifenkopf eine Bedingung, wann die Schleife beendet werden soll. Dies wird in der Regel durch die Funktion `feof()` realisiert.

Syntax und Bedeutung der Funktion `feof()`

Mit der Funktion `feof()` und dem Handle, welchen Sie von `fopen()` erhalten haben, überprüfen Sie, ob der Dateizeiger am Ende der Datei steht.

`feof(Handle);`

`feof()` liefert `TRUE` zurück, wenn der Dateizeiger am Ende der Datei steht, andernfalls `FALSE`. Dies wird als Bedingung für die `while()`-Schleife verwendet: `while (!feof($handle))`.

Das bedeutet, solange das Dateiende noch nicht erreicht ist, gibt `feof()` `FALSE` zurück. Da die Bedingung im Schleifenkopf über das Ausrufezeichen `!` umgekehrt ist, ist die Bedingung erfüllt, da `! FALSE` gleichbedeutend mit `TRUE` ist. Die Schleife wird weiter durchlaufen. Erst wenn das Ende der Datei erreicht ist, gibt `feof()` `TRUE` zurück, `! TRUE` ergibt `FALSE`, die Schleife wird beendet.

Dateien mit `fclose()` schließen

Nachdem Sie eine Datei geöffnet und die Daten ausgelesen haben, sollten Sie die Datei wieder schließen, um sie für andere Prozesse nutzbar zu machen. PHP schließt die Verbindung zur Datei mit Ende des PHP-Skriptes zwar automatisch, allerdings gehört es zum guten Programmierstil, geöffnete Dateien wieder regulär zu schließen, um Ressourcen zu sparen und etwaige Fehlerquellen zu reduzieren.

Syntax und Bedeutung der `fclose()`-Anweisung

- ✓ Mit der Funktion `fclose()` schließen Sie den geöffneten Dateistream. `fclose()` erwartet das Datei-Handle, das zuvor über `fopen()` geliefert wurde.
- ✓ Wurde die Datei erfolgreich geschlossen, gibt die Funktion `fclose()` den Wert `TRUE` zurück, ansonsten `FALSE`.

`fclose(Handle);`
`fclose($handle);`

Beispiel zu „Externe Datei öffnen, lesen und schließen“: *fgets.php*

```

<?php
①  if (is_file("protokoll.txt")) {
②      $handle = fopen("protokoll.txt", "r");
③      if ($handle != FALSE) {
④          while (!feof($handle)) { // Schleife bis zum Dateiende
⑤              $zeile = fgets($handle); // liest aktuelle Zeile
⑥              echo $zeile . "<br>";
⑦          }
⑧          fclose($handle);
⑨      } else {
⑩          echo "<p>Beim Öffnen der Datei trat ein Fehler auf.</p>";
⑪      }
⑫      } else {
⑬          echo "<p>Der angegebene Name ist keine Datei.</p>";
⑭      }
⑮
    ?>

```

- ① Es wird geprüft, ob der angegebene Name *protokoll.txt* eine Datei ist. Bei erfolgreicher Prüfung wird der Anweisungsblock zum Öffnen der Datei durchlaufen, ansonsten eine Fehlermeldung im *else*-Zweig ausgeben.
- ② Die Datei *protokoll.txt* wird mit dem Modus "r" zum Lesen geöffnet. Die Funktion *fopen()* gibt das Handle zurück. Dieses wird in der *resource*-Variablen *\$handle* gespeichert.
- ③ In der *if*-Anweisung wird geprüft, ob die Datei geöffnet werden konnte. Falls ja, wird der *if*-Zweig ausgeführt, anderenfalls der *else*-Zweig.
- ④ Um den kompletten Inhalt der Datei auszulesen, wird eine *while*-Schleife verwendet, die so lange durchlaufen wird, bis das Ende der auszulesenden Datei erreicht ist. Ob der Dateizeiger am Ende der Datei steht, prüfen Sie mit der Funktion *feof()*. Solange das Ende der Datei nicht erreicht ist, gibt *feof()* FALSE zurück, die *while*-Schleife wird weiter ausgeführt. Am Ende der Datei gibt *feof()* TRUE zurück, damit wird die *while*-Schleife dann beendet.
- ⑤ Die aktuelle Zeile wird ausgelesen und in der Variablen *\$zeile* gespeichert. Da *fgets()* zum Auslesen der Zeile verwendet wird, wird der Dateizeiger automatisch auf die nächste Zeile gesetzt, die dann im nächsten Durchlauf der Schleife ausgelesen wird.
- ⑥ Die Variable *\$zeile* wird per *echo* ausgegeben.
- ⑦ Der Stream zur Datei *protokoll.txt* wird nach der *while*-Schleife geschlossen.



Anzeige der Beispieldatei „*fgets.php*“

8.3 Weitere Möglichkeiten zum Lesen von Dateien

Dateien mit `readfile()` oder `file()` lesen

Während die Funktion `fgets()` den Inhalt einer Datei zeilenweise ausliest, können Sie mit den Befehlen `readfile()` und `file()` den gesamten Inhalt einer Datei auf einmal auslesen.

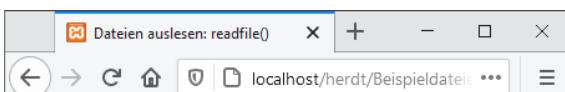
Der Unterschied zwischen den beiden Funktionen `readfile()` und `file()` liegt in der Ausgabe der Daten:

- ✓ `readfile()` liest den Inhalt der Datei vollständig aus und sendet das Ergebnis ohne weitere Bearbeitung und Weiterverarbeitungsmöglichkeit direkt an den Browser. Zeilenumbrüche werden nicht dargestellt, wenn sie nicht als HTML-Tags in der eingelesenen Datei vorhanden sind.
- ✓ `file()` liest den vollständigen Inhalt der Datei in ein Array ein. Hierbei wird jeweils eine Zeile zu einem Eintrag im Array.

```
readfile("Datei");
file("Datei");
```

Beispiel `readfile.php` und `file.php`

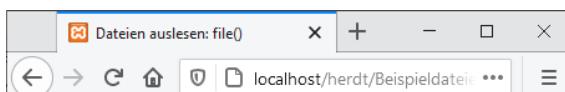
<pre><?php readfile("protokoll.txt"); ?></pre>	<pre><?php \$feld = file("protokoll.txt"); echo "<p>"; \$i = 1; foreach (\$feld as \$zeile) { echo "Zeile " . \$i++ . ": "; echo \$zeile . "
"; } echo "</p>"; ?></pre>
--	--



Auslesen einer Datei per `readfile()`

18.04.2021 16:07:01 Dienst wurde erfolgreich gestartet. 18.04.2021 19:30:31 Statusabfrage: Alles in Ordnung! 18.04.2021 22:47:13 Dienst wurde erfolgreich beendet.

Beispieldatei „`readfile.php`“



Auslesen einer Datei per `file()`

Zeile 1: 18.04.2021 16:07:01 Dienst wurde erfolgreich gestartet.
Zeile 2: 18.04.2021 19:30:31 Statusabfrage: Alles in Ordnung!
Zeile 3: 18.04.2021 22:47:13 Dienst wurde erfolgreich beendet.

Beispieldatei „`file.php`“

Dateien mit `file_get_contents()` lesen

Die Funktion `file_get_contents()` ist die empfohlene Methode, Inhalte einer Datei einzulesen. Dabei werden die Inhalte anders als bei `readfile()` in einem String gespeichert.

```
file_get_contents("Dateiname");
```

Eine Weiterverarbeitung ist so problemlos möglich. Vergleichbar zu `file()` ist nur ein Aufruf notwendig. Das Öffnen und Schließen der Datei wird dabei automatisch von der Funktion übernommen.

Ausgelesene Inhalte verarbeiten

Beim Einlesen von Dateien besteht häufig das Problem, dass in der Datei vorhandene Zeilenumbrüche zwar erkannt werden, bei der Ausgabe im Browser aber nicht als Zeilenwechsel dargestellt werden.

```
nl2br("Zeichenkette [, true/false]");
```

PHP bietet für diese Zwecke die Funktion `nl2br()` (gesprochen: „new line to break“, also „Wandle Zeilenumbrüche in `
`-Tags um“). Die Funktion erfüllt genau diese Aufgabe, sie erkennt Zeilenwechsel in externen Dateien und wandelt diese in HTML-Tags für einen Zeilenumbruch `
` um.

Der zweite Parameter ist optional. Wird dieser nicht angegeben, werden XHTML-konforme `
`-Tags ausgegeben; der Standardwert für den zweiten Parameter ist `TRUE`. Möchten Sie einfache `
`-Tags erhalten, müssen Sie hier `FALSE` angeben.

Beispiel: `file_get_contents.php`

```
<?php
① $inhalt = file_get_contents("protokoll.txt");
② echo nl2br($inhalt, FALSE); // nl2br: Zeilenumbrüche erkennen und
                                // in <br>-Befehle umwandeln
?>
```

- ① Über die Funktion `file_get_contents()` wird der Inhalt der Datei `protokoll.txt` in einen String eingelesen und unter dem Variablennamen `$inhalt` gespeichert. Da HTML5 das Tag `
` in seiner alten Schreibweise zulässt, wird hier der zweite Parameter auf `FALSE` gesetzt.
- ② Die Variable `$inhalt` wird auf dem Bildschirm ausgegeben. Die Funktion `nl2br()` wird verwendet, um Zeilenumbrüche, die in der eingelesenen Datei vorhanden sind, in `
`-Tags umzuwandeln und so auch Zeilenumbrüche in der Ausgabe darzustellen.



Anzeige der Beispieldatei „file_get_contents.php“

Daten in anderen Verzeichnissen auslesen

In den bisherigen Beispielen lagen die eingelesenen Textdateien immer im gleichen Verzeichnis wie die PHP-Datei. In der Praxis ist dies jedoch selten der Fall. Statt dem Dateinamen können Sie auch Pfadangaben in den vorgestellten Funktionen verwenden, entweder relativ, z. B. `fopen ("..../files/protokoll.txt", "r",)` oder absolut. Ein Beispiel für einen absoluten Pfad sehen Sie in folgendem Beispiel.

Beispiel: `pfad.php`

```
<?php
① $datei = "protokoll.txt";
② $pfad =
"C:\\xampp\\htdocs\\Herdt\\Beispieldateien\\Kapitel_09\\";
③ echo "<p>Inhalt der Date " . $pfad . $datei . "</p>";
④ $inhalt = file_get_contents($pfad . $datei);
⑤ echo "<pre>";
print_r($inhalt);
echo "</pre>";
?>
```

- ① Der Dateiname wird hier in einer Variablen `$datei` gespeichert.
- ② Der Pfad wird ebenfalls in einer Variablen `$pfad` gespeichert. Zu beobachten ist, dass hier doppelte Backslashes im Dateipfad verwendet werden.



Anzeige der Beispieldatei „`pfad.php`“

Dies ist für Pfade unter Windows notwendig, da der Backslash `\` in PHP eine Escape-Sequenz und den nachfolgenden Buchstaben ausblenden würde. Das bedeutet, der jeweils erste `\` escaped den folgenden `\`, damit dieser von PHP auch erkannt wird. Unter Linux wird in Pfadangaben der normale Schrägstrich `/` verwendet, die Problematik stellt sich dort nicht.

- ③ Die Variablen `$pfad` und `$datei` werden über den Punkt `.` konkatiniert und hier ausgegeben. In der Abbildung können Sie die escapten Backslashes `\` sehen.
- ④ Die Variablen `$pfad` und `$datei` werden konkatiniert und der Funktion übergeben. Die Inhalte werden über `file_get_contents()` eingelesen.
- ⑤ Hier wird der Inhalt über `echo` und `print_r`-Anweisungen ausgegeben. Das HTML-Tag `<pre>` sorgt dafür, dass die Zeilenumbrüche aus der Textdatei so angezeigt werden, wie sie dort vorkommen.

8.4 In Dateien schreiben

Dateien zum Schreiben öffnen

Wollen Sie Daten in eine Datei schreiben, müssen Sie die Datei zuerst öffnen. Beim Öffnen der Datei können Sie bestimmen, ob Sie den Inhalt der Datei überschreiben oder weitere Inhalte hinzufügen möchten.

Dateien überschreiben

Wenn Sie eine Datei mit der Funktion `fopen()` und dem Modus '`w`' für `write` öffnen, wird die Datei zum Schreiben geöffnet und der Dateizeiger auf den Anfang der Datei verschoben. Gleichzeitig wird die Länge der Datei auf 0 Byte gesetzt. Das bedeutet, dass vorhandene Inhalte gelöscht werden. Wenn die Datei nicht existiert, wird sie angelegt.

Daten in Dateien hinzufügen

Um Daten fortlaufend in eine Datei zu schreiben, verwenden Sie den Befehl `fopen()` mit dem Modus '`a`'. Die Datei wird zum Schreiben geöffnet und der Dateizeiger an das Ende der Datei gesetzt, sodass die neuen Daten hinzugefügt werden. Existiert sie nicht, legt PHP die Datei an.

Daten in Dateien schreiben

Um eine Zeichenkette in eine geöffnete Datei zu schreiben, verwenden Sie die Funktion `fwrite()`.

Syntax und Bedeutung der Funktion `fwrite()`

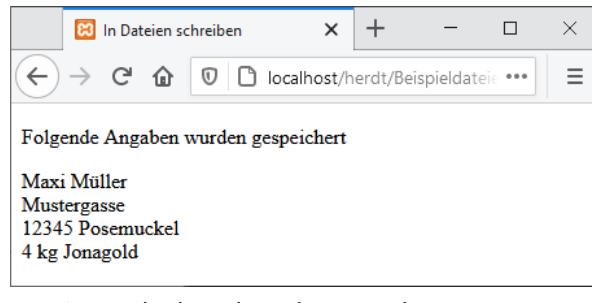
```
fwrite(Handle, Zeichenkette[, Länge]);
```

- ✓ Mit der Funktion `fwrite()` können Sie eine beliebige Zeichenkette in eine Datei schreiben.
- ✓ Die entsprechende Datei wird über das von `fopen()` zurückgegebene Handle angesprochen.
- ✓ Ist der optionale Parameter `Länge` angegeben, wird das Schreiben nach der angegebenen Anzahl Bytes beendet. Geben Sie ihn nicht an, wird die gesamte Zeichenkette geschrieben.
- ✓ `fwrite()` gibt bei Erfolg die Anzahl der geschriebenen Bytes zurück, andernfalls `FALSE`.

Nachdem Daten in die Datei geschrieben wurden, sollte die Datei geschlossen werden, damit sie für andere Zugriffe nicht gesperrt ist.

Beispiel: *bestellung.php*

Die Daten, die der Benutzer in ein Bestellformular für Äpfel (Datei *bestellformular.html*) eingibt, sollen in eine *.csv-Datei (*bestellung_daten.csv*) geschrieben werden. Existiert die *.csv-Datei noch nicht, wird sie angelegt, anderenfalls werden die Daten an das Ende der Datei geschrieben.



Anzeige nach Absendung des Formulars
„bestellformular.html“

```
<?php
① $handle = fopen("bestellung_daten.csv", "a");
② if ($handle == FALSE) {
    echo "<p>Datei konnte nicht zum Schreiben geöffnet werden</p>";
③ die("<p>Das Programm wird beendet.</p>");
}
④ $name = $_POST["name"];
$strasse = $_POST["strasse"];
$ort = $_POST["ort"];
$sorte = $_POST["sorte"];
$menge = $_POST["menge"];
⑤ if (fwrite($handle,
    utf8_decode("$name;$strasse;$ort;$sorte;$menge;\n")) {
    echo "<p>Folgende Angaben wurden gespeichert</p>";
    echo "<p>$name<br>$strasse<br>$ort<br>$menge kg $sorte</p>";
} else {
    echo "<p>Das Schreiben der Daten ist fehlgeschlagen.</p>";
}
⑦ fclose($handle);
?>
```

- ① Die Datei *bestellung_daten.csv* wird mit `fopen()` im Modus 'a' geöffnet, d. h. zum Schreiben und Anhängen neuer Daten. Der Dateizeiger wird an das Ende der Datei gesetzt, sodass die neuen Daten hinzugefügt werden.
- ② Es wird geprüft, ob das Öffnen der Datei erfolgreich war.
- ③ Wenn die Datei nicht erfolgreich geöffnet werden kann, z. B. weil Sie nicht über die nötigen Berechtigungen verfügen, wird mit `die()` das Programm sofort beendet. Die Funktion `die()` kann auch ohne Parameter aufgerufen werden. Wollen Sie eine Meldung über das Beenden des Skripts ausgeben, können Sie `die()` mit einer Zeichenkette aufrufen, welche vor dem Beenden des Skripts ausgegeben wird. Diese Zeichenkette kann auch HTML-Tags enthalten.
- ④ Die Variablen `$name`, `$strasse`, `$ort`, `$sorte` und `$menge` werden mit den übergebenen Formulardaten gefüllt.
- ⑤ Über die Funktion `fwrite()` werden die Werte der Variablen in die geöffnete Datei geschrieben. Hierbei werden die Daten jeweils durch ein Semikolon voneinander getrennt. Dieses wird in CSV-Dateien als Trennzeichen der einzelnen Einträge verwendet. Am Ende einer Zeile wird mit einem "\n" ein Zeilenumbruch erzeugt. Im Erfolgs- bzw. im Fehlerfall wird eine entsprechende Meldung ausgegeben.

- ⑥ Zusätzlich sehen Sie, dass die Zeichenkette von der Funktion `utf8_decode()` umschlossen ist. Um für Windows eine Datei mit dem ISO-8859-1-Zeichensatz zu erstellen, müssen UTF-8-Zeichen konvertiert werden.
- ⑦ Die geöffnete Datei wird geschlossen.

*.csv-Dateien können Sie mit einem Programm wie z. B. *LibreOffice Calc* oder *Microsoft Excel* öffnen und bearbeiten.

	A	B	C	D	E
1	Petra Meyer	Alsterpfad 34a	23456 Hamburg	Jonagold	4
2	Fred Müller	Isarallee 123	87654 München	Gala	2,5
3	Arndt Hoffmann	Mainufer 86	65432 Frankfurt/Main	Elstar	8
4	Maxi Müller	Mustergasse	12345 Posemuckel	Jonagold	4

Ansicht der Datei „bestellung_daten.csv“ in Microsoft Excel

Daten mit der Funktion `file_put_contents()` schreiben

`file_put_contents()` ist eine Funktion, die das Schreiben einer Zeichenkette in eine Datei vereinfacht. Die Funktion fasst die bei `fwrite()` notwendigen Schritte `fopen()`, `fwrite()` und `fclose()` zusammen. Das Öffnen und Schließen der Datei wird dabei automatisch von der Funktion übernommen.

Syntax und Bedeutung der Funktion `file_put_contents()`

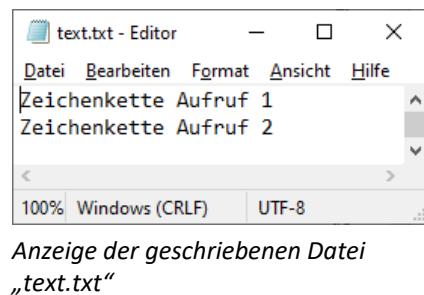
```
file_put_contents(Dateiname, Zeichenkette[, Flag]);
```

- ✓ Die Funktion entspricht `fopen()`, `fwrite()` und `fclose()`.
- ✓ Geben Sie den optionalen Parameter `Flag` nicht an, werden eventuell vorhandene Daten überschrieben. Durch Angabe von `FILE_APPEND` wird eine Zeichenkette bei Aufruf der Funktion am Ende der Datei angefügt.

Beispiel: `file_put_contents.php`

```
<?php
① $file = "text.txt";
② if (file_put_contents($file, "Zeichenkette Aufruf 1\r\n")) {
    echo "<p>Funktionsaufruf 1: erfolgreich.</p>";
}
③ if (file_put_contents($file, "Zeichenkette Aufruf 2\r\n",
                      FILE_APPEND)) {
    echo "<p>Funktionsaufruf 2: erfolgreich.</p>";
}
?>
```

- ① Der Name der Datei, in die geschrieben werden soll, wird festgelegt und der Variablen `$file` zugewiesen.
- ② `file_put_contents()` wird aufgerufen, die Datei `text.txt` wird angelegt und die angegebene Beispiel-Zeichenkette in die Datei geschrieben. Auch hier ist die Voraussetzung, dass Sie Schreibrechte auf das Verzeichnis haben, in dem die Datei angelegt werden soll.



Liefert die Funktion bei erfolgreichem Schreiben `TRUE` zurück, wird auf dem Bildschirm eine Meldung ausgegeben. Damit auch der Windows-Editor die Datei richtig anzeigen kann, wird in diesem Beispiel die Windows-Variante des Zeilenumbruchs "`\r\n`" verwendet.

- ③ Beim nochmaligen Aufruf der Funktion `file_put_contents()` wird das optionale Flag `FILE_APPEND` angegeben. Dies bewirkt, dass die Zeichenkette der bestehenden Datei angefügt wird. Vorhandene Inhalte werden nicht überschrieben.

8.5 Weitere Datei-Funktionen

Dateien per `flock()` sperren

Syntax und Bedeutung der Funktion `flock()`

- ✓ `flock()` sperrt den Zugriff auf die Datei, auf die das Handle verweist. Der Modus ermöglicht verschiedene Sperrzustände der Datei.
- ✓ Darf während Ihres Zugriffs die Datei von anderen nur gelesen werden, setzen Sie den Modus auf `LOCK_SH` (Shared Lock, Lesezugriff).
- ✓ Soll kein anderer Nutzer zeitgleich die Datei nutzen dürfen, setzen Sie die Option `LOCK_EX` (Exclusive Lock). `flock()` wartet, bis die Datei wie angegeben benutzt werden kann. Geben Sie zusätzlich zum `LOCK_EX` den Schalter `LOCK_NB` an (Syntax: `flock($fp, LOCK_EX | LOCK_NB)`), gibt die Funktion den Wert `FALSE` zurück, wenn die Datei bereits von einem anderen Programm gesperrt ist.
- ✓ Möchten Sie die Verriegelung wieder freigeben, rufen Sie `flock()` erneut mit dem Schalter `LOCK_UN` (Unlock) auf.
- ✓ Die Funktion liefert bei Erfolg den Wert `TRUE` zurück bzw. `FALSE`, wenn ein Fehler auftritt.
- ✓ Die Sperre wird automatisch über die Funktion `fclose()` am Ende des PHP-Skripts aufgehoben, falls Sie diese nicht selber aufgehoben haben.

`flock(Handle, Modus);`

! Bei manchen Betriebssystemen ist die Funktion `flock()` auf Prozess-Ebene (Teil einer Applikation) implementiert. Hierbei können Sie sich nicht auf `flock()` verlassen, um Dateien vor dem Zugriff von anderen PHP-Skripten zu schützen.

Zudem wird `flock()` nicht von allen Dateisystemen unterstützt. In solchen Umgebungen erhalten Sie ein `FALSE` zurück, wenn Sie `flock()` verwenden.

Position des Dateizeigers mit **fseek()** setzen

Syntax und Bedeutung der Funktion **fseek()**

In manchen Fällen ist es notwendig, den Dateizeiger innerhalb der Datei neu zu positionieren. Wie oben beschrieben bewegt **fgets()** den Dateizeiger nach dem Lesen eine Zeile weiter. Möchten Sie nach einer Leseaktion eine Schreibaktion durchführen und den zuvor ausgelesenen Inhalt überschreiben, müssen Sie zunächst den Dateizeiger an die richtige Stelle bewegen.

- ✓ Als ersten Parameter verwenden Sie das Handle, welchen Sie zuvor beim Öffnen der Datei erhalten haben.
- ✓ Für den Parameter **Stelle** geben Sie die Anzahl in Bytes, bezogen auf den Dateianfang, der durch den Dateizeiger festgelegt wird, an.
- ✓ Der optionale Parameter **Wie** legt bestimmte Bezugspunkte für die Ermittlung der Position fest: Der Parameter **SEEK_CUR** ermöglicht die Verschiebung auf die aktuelle Position des Dateizeigers plus **Stelle**, **SEEK_END** vom Dateiende und **SEEK_SET** vom Dateianfang.
- ✓ Fehlt der Parameter **Wie**, ist standardmäßig die Option **SEEK_SET** gewählt.

fseek(Handle, Stelle [, Wie]);

8.6 Zugriffszähler für eine Webseite

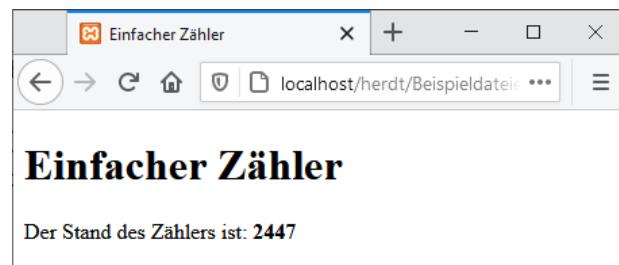
Ein einfaches Beispiel in Verbindung mit Dateizugriffen ist die Einbettung eines Counters (Zugriffszählers) in eine Webseite. Ein Counter zählt die Anzahl der Webseiten-Zugriffe. Dabei wird bei jedem Zugriff der Wert des Zählers um eins erhöht. Das heißt, es wird auf eine bestehende Datei zugegriffen, der Inhalt, also die Zahl wird ausgelesen und um den Wert 1 erhöht. Das Ergebnis der Berechnung überschreibt dann den Inhalt der Datei.

Beispiel: *counter.php*

```

① <p>Der Stand des Zählers ist: <?php counter() ?></p>
<?php
②   function counter() {
③     $name = "counter.txt";
④     $handle = fopen($name, "r+");
⑤     if ($handle) {
⑥       flock($handle, LOCK_EX);
⑦       $count = fgets($handle, 10);
⑧       fseek($handle, 0);
⑨       // Zähler erhöhen und ausgeben
⑩       echo "<strong>" . ++$count . "</strong>";
        fwrite($handle, $count);
        fclose($handle);
      }
    }
?>
```

- ① Im HTML-Text wird die Funktion `counter()` zur Anzeige der Besucherzahl aufgerufen.
- ② Die Funktion `counter()` wird definiert.
- ③ Der Name der Datei, in der die Anzahl der Besucher abgelegt werden soll, lautet `counter.txt` und wird der Variablen `$name` zugewiesen.
- ④ Die Datei `counter.txt` wird über die Funktion `fopen()` zum Lesen und Schreiben geöffnet. Der Zugriff auf diese Datei erfolgt über die Handle-Variable `$handle`. Durch Verwendung des Modus `r+` wird der Dateizeiger an den Anfang der Datei gesetzt.
- ⑤ Damit während der Arbeit mit der Datei kein weiterer Zugriff und somit kein gleichzeitiges Überschreiben der Daten möglich ist, wird die Datei für andere Zugriffe (`LOCK_EX`) gesperrt.
- ⑥ Mit `fgets()` werden die ersten 10 Zeichen der Datei ausgelesen. Der Wert 10 wurde gewählt, da hiermit die Verarbeitung einer Zahl mit 10 Stellen (bis 9999999999) gewährleistet ist.
- ⑦ Damit der neue Wert den alten Wert in der Datei überschreiben kann, wird der Dateizeiger mittels `fseek()` und der Angabe 0 an den Anfang der Datei gesetzt.
- ⑧ Der Wert wird um eins erhöht und am Bildschirm fett formatiert ausgegeben.
- ⑨ Über die Funktion `fwrite()` wird der neue Wert der Variablen `$count` in die Datei geschrieben.
- ⑩ `fclose()` schließt die Datei `counter.txt` und beendet zugleich die Zugriffssperre, die über `flock()` gesetzt wurde.



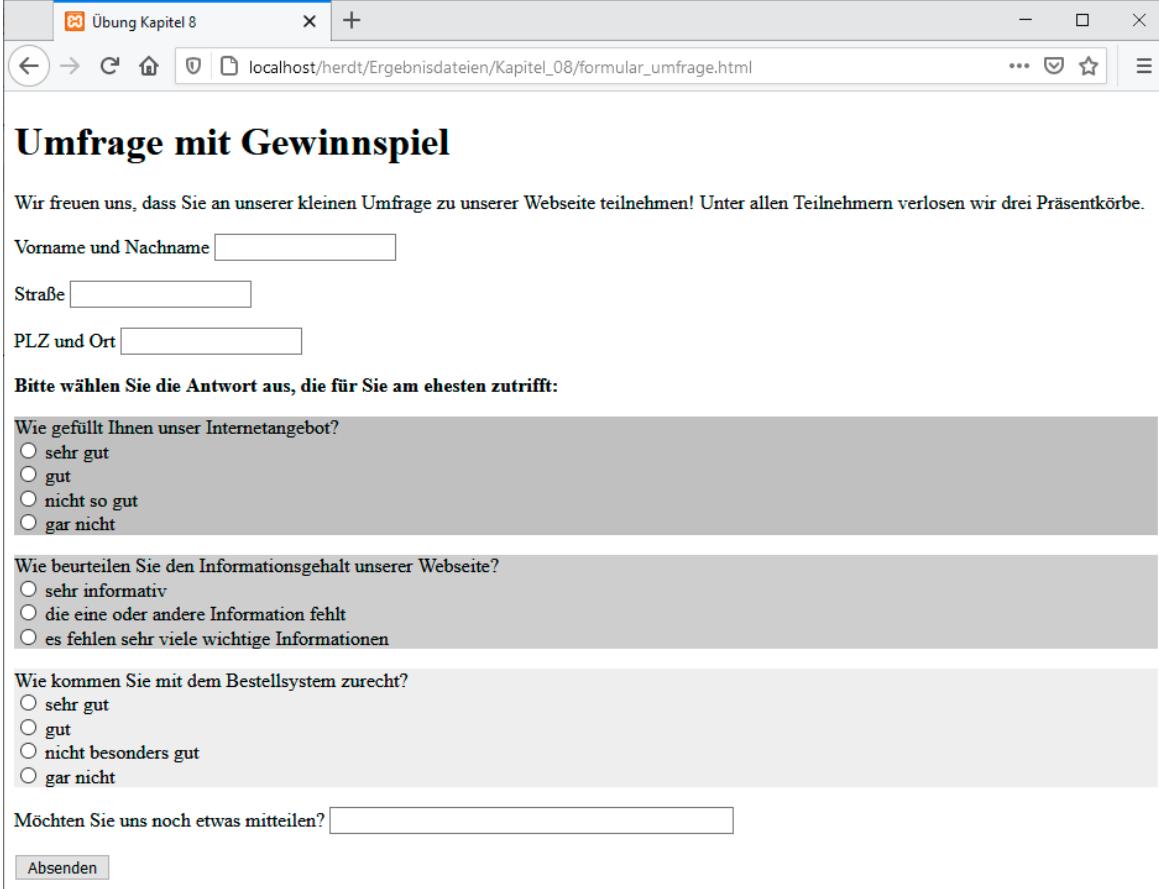
Ausgabe der Beispieldatei „counter.php“

8.7 Übung

Umfrage mit Gewinnspiel

Level		Zeit	ca. 20 min
Übungsinhalte	<ul style="list-style-type: none"> ✓ HTML-Formulare ✓ Dateien öffnen ✓ Daten in Dateien schreiben ✓ Dateien schließen 		
Übungsdatei	--		
Ergebnisdateien	<i>formular_umfrage.htm, umfrage.php, umfrage_daten.csv</i>		

1. Sie möchten eine Umfrage durchführen. Hierfür erstellen Sie ein Formular, in das Daten eingegeben werden. Um die Daten auswerten zu können, schreiben Sie ein PHP-Skript, das die Eingaben in einer CSV-Tabelle speichert.
2. Das Umfrageformular speichern Sie unter dem Namen *formular_umfrage.html*, das auswertende PHP-Skript unter dem Namen *umfrage.php* und die übergebenen Daten werden in der Datei *umfrage_daten.csv* gespeichert.



The screenshot shows a web browser window titled "Übung Kapitel 8". The address bar displays "localhost/herdt/Ergebnisdateien/Kapitel_08/formular_umfrage.html". The page content is as follows:

Umfrage mit Gewinnspiel

Wir freuen uns, dass Sie an unserer kleinen Umfrage zu unserer Webseite teilnehmen! Unter allen Teilnehmern verlosen wir drei Präsentkörbe.

Vorname und Nachname

Straße

PLZ und Ort

Bitte wählen Sie die Antwort aus, die für Sie am ehesten zutrifft:

Wie gefüllt Ihnen unser Internetangebot?

sehr gut
 gut
 nicht so gut
 gar nicht

Wie beurteilen Sie den Informationsgehalt unserer Webseite?

sehr informativ
 die eine oder andere Information fehlt
 es fehlen sehr viele wichtige Informationen

Wie kommen Sie mit dem Bestellsystem zurecht?

sehr gut
 gut
 nicht besonders gut
 gar nicht

Möchten Sie uns noch etwas mitteilen?

Formular „*formular_umfrage.html*“

9

Zeichenketten-Funktionen



Beispieldateien: Dateien im Ordner *Kapitel_09*

9.1 Zeichenketten ausgeben

Daten liegen abhängig von ihrer Herkunft häufig als Rohdaten vor, haben keine besondere Formatisierung und sind für die Ausgabe nicht immer geeignet. PHP bietet vielfältige Funktionen an, mit denen Sie Daten in eine bestimmte optische Form bringen können, die entweder besser lesbar ist oder bestimmten Konventionen der Sprache entspricht (beispielsweise für die Darstellung von Fließkommazahlen, einer Telefonnummer oder eines Datums).

Zeichenketten mit der Funktion `printf()` formatieren

Wenn Sie Daten für die Ausgabe im Browser formatieren möchten, verwenden Sie die Funktion `printf()`. Die Funktion gibt das formatierte Ergebnis direkt im Browser aus, es wird keine `echo`-Anweisung benötigt.

`printf` (*print formatted* = formatierte Ausgabe) ermöglicht Ihnen, durch die Angabe spezieller Optionen festzulegen, wie eine Zahl oder eine Zeichenkette ausgegeben werden soll. Ein Beispiel ist die Ausgabe von Fließkommazahlen. Bei Berechnungen erhalten Sie als Rückgabe mitunter eine Fließkommazahl mit mehreren Nachkommastellen. Sie möchten beispielsweise nur eine oder zwei Stellen nach dem Komma am Bildschirm ausgeben, z. B. 1.780,5 km oder 19,50 €.

Syntax der Funktion `printf()`

```
printf(Formatierung[, Argument1[, Argument2, ...]]);
```

- ✓ Der Parameter Formatierung ist eine Zeichenkette, worin Anweisungen für eine formatierte Ausgabe enthalten sind.
- ✓ Die Formatierung kann keine, eine oder mehrere Anweisungen beinhalten.
- ✓ Jede Formatierungsanweisung besteht aus einem Prozentzeichen (%) gefolgt von einem oder mehreren Elementen, die eine bestimmte Formatierung festlegen. Die Typangabe in der Formatierung ist notwendig, ohne diese erfolgt eine nicht korrekte Ausgabe.

- ✓ Jede einzelne Formatierungsanweisung in der Zeichenkette Formatierung, die mit dem `%` eingeleitet wird, wird von links nach rechts mit den einzelnen Parametern verarbeitet und dann an die definierte Stelle in die Zeichenkette eingesetzt.
- ✓ Innerhalb der Formatierung können auch einzelne Zeichen eingesetzt werden, die bei der Ausgabe mit auf dem Bildschirm erscheinen sollen, z. B. Leerzeichen, Buchstaben bzw. Zahlen oder Sonderzeichen wie beispielsweise `***`.

Formatierungen angeben

Typangabe

Hiermit legen Sie fest, welchen Datentyp das auszugebende Argument haben soll. In dieser Übersicht finden Sie die geläufigsten Optionen. Eine Übersicht aller „Schalter“ finden Sie unter <https://www.php.net/sprintf>.

Option	Erläuterung
b	Die Zeichenkette wird als Ganzzahl angesehen und soll in binärer (<i>binary</i>) Schreibweise ausgegeben werden.
c	Eine Ganzzahl soll als ASCII-Code dargestellt werden, z. B. 65 ergibt A.
d	Eine Ganzzahl erhält führende Füllzeichen, z. B. Nullen: 3:14 Uhr ergibt 03:14 Uhr.
f	Der Parameter soll als Fließkommazahl dargestellt werden.
s	Das Argument wird als Zeichenkette (<i>string</i>) angesehen und als solche ausgegeben.
x	Die Ganzzahl wird in hexadezimaler Form dargestellt. Hexadezimale Ziffern werden als Zahlen von 0 bis 9 und als Kleinbuchstaben a, b, c, d, e, f dargestellt.

Die Typangabe bestimmt den Datentyp, welchen PHP darstellt. Hier greift die automatische Datentypkonvertierung von PHP. Deutlich wird dies in folgendem Code-Beispiel.

Beispiel: typ_angabe.php

```
<?php
①   printf("<p>Ausgabe Typ b: <b>%b</b></p>" , "Hallo");
②   printf("<p>Ausgabe Typ c: <b>%c</b></p>" , "Hallo");
③   printf("<p>Ausgabe Typ d: <b>%d</b></p>" , "Hallo");
④   printf("<p>Ausgabe Typ f: <b>%f</b></p>" , "Hallo");
⑤   printf("<p>Ausgabe Typ s: <b>%s</b></p>" , "Hallo");
⑥   printf("<p>Ausgabe Typ x: <b>%x</b></p>" , "Hallo");
    echo "<hr>";
    $ascii = 83;
⑦   printf("<p>Typ c von $ascii: <b>%c</b></p>" , $ascii);
    $string1 = "Hallo";
    $string2 = "Welt";
⑧   printf("<p>Zwei Parameter: <i>%s</i> <b>%s</b></p>" ,
            $string1, $string2);
?
?>
```

- ① Hier wird die Option `b` angegeben. `printf()` wandelt die übergebene Zeichenkette in eine Zahl um und gibt den binären Wert aus.
- ② Über die Option `c` soll der ASCII Code ausgegeben werden. Da „Hallo“ eine Zeichenkette ist, kommt es zu keiner Ausgabe, der Typ `c` benötigt einen *int*-Wert.
- ③ Über `d` wird hier der String in einen Integer konvertiert, der Wert von „Hallo“ ist 0.
- ④ Hier entsprechend die Konvertierung des Strings in ein *float* bei der Option `f`.
- ⑤ Die Option `s` ist passend für die Zeichenkette, hier wird „Hallo“ korrekt ausgegeben.
- ⑥ Auch bei der Option `x` wandelt PHP den String in einen Integer um, die Ausgabe ist hier ebenfalls 0.
- ⑦ Hier wird das ASCII-Zeichen von der zuvor definierten Variablen ausgegeben. Hier funktioniert die Option `c` korrekt. Wie Sie in diesem Beispielen sehen, können die Parameter direkt im Funktionsaufruf notiert werden, oder eben per Variable übergeben werden.
- ⑧ In dieser Zeile wird die Option `s` für *string* angegeben und es werden zwei Parameter angegeben. Jedes vorkommende `(%)`-Zeichen in der Formatierungsanweisung wird von links nach rechts mit den übergebenen Parametern ersetzt.

```
Ausgabe Typ b: 0
Ausgabe Typ c:
Ausgabe Typ d: 0
Ausgabe Typ f: 0.000000
Ausgabe Typ s: Hallo
Ausgabe Typ x: 0

Typ c von 83: S
Zwei Parameter: Hallo Welt
```

Ausgabe der Beispieldatei „typ_angabe.php“

Füllzeichen und Ausrichtung der Füllzeichen

Über `printf()` können Sie eine Zeichenkette auf eine bestimmte Länge mit beliebigen Zeichen auffüllen. Dabei wird ein String, der kürzer als die Anzahl der angegebenen Zeichen ist, mit dem Füllzeichen aufgefüllt, längere Strings werden nicht ergänzt, aber auch nicht gekürzt. Zusätzlich können Sie festlegen, ob die Zeichen am Anfang oder am Ende der Zeichenkette aufgefüllt werden sollen.

Wie in folgendem Beispiel zu sehen, kann das nützlich sein, wenn Sie Buchungsnummern immer mit einer bestimmten Länge ausgeben möchten.

Beispiel: fuellzeichen.php

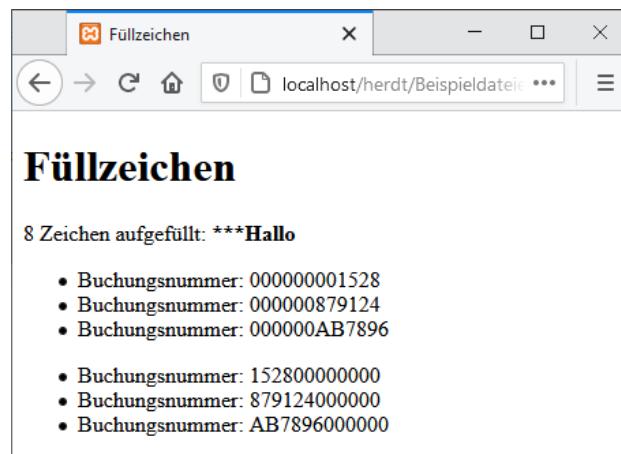
	<code><?php</code>
①	<code>printf("<p>8 Zeichen aufgefüllt: % *8s</p>",</code>
	<code> "Hello");</code>
②	<code>\$buchungsnummern = [1528, 879124, "AB7896"];</code>
	<code>echo "";</code>
③	<code>foreach (\$buchungsnummern as \$eine_nummer) {</code>

```

④    printf("<li>Buchungsnummer: %012s", $eine_nummer);
    }
    echo "</ul>";
    echo "<ul>";
    foreach ($buchungsnummern as $eine_nummer) {
        ⑤    printf("<li>Buchungsnummer: %-012s", $eine_nummer);
    }
    echo "</ul>";
?>

```

- ① Hier wird die Zeichenkette auf eine Länge von 8 Zeichen aufgefüllt. Falls Sie einen Buchstaben oder ein anderes Zeichen übergeben, müssen Sie diesem ein Hochkomma voranstellen. Zahlen geben Sie ohne an.
- ② Es wird über die Kurzschrifweise ein Array mit verschiedenen Buchungsnummern unterschiedlicher Länge definiert.
- ③ Das Array wird über eine `foreach ()`-Schleife durchlaufen.



Ausgabe der Beispieldatei „fuellzeichen.php“

- ④ Die jeweilige Nummer wird als HTML-Listeneintrag ausgegeben. `printf()` füllt die Nummern mit 12 Nullen auf. Per Standard werden die Zeichen links, also vor der Zeichenkette aufgefüllt. Beachten Sie:
Die erste 0 hinter dem `%`-Zeichen definiert das eigentliche Füllzeichen, die Anzahl der aufzufüllenden Stellen folgt dahinter und kann ein oder mehrstellig sein.
- ⑤ In einer weiteren Schleife wird das definierte Array wiederholt durchlaufen. Hier steht vor dem Füllzeichen 0 das Minus-Zeichen `-`. Dieser „Schalter“ bewirkt, dass die Zeichen nicht vorne, sondern am Ende der Zeichenkette aufgefüllt werden.

Nachkommastellen

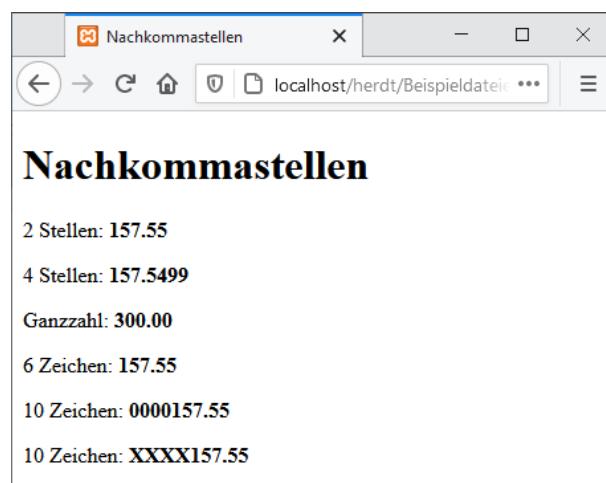
`printf()` bietet ebenfalls die Möglichkeit, die Menge der Nachkommastellen festzulegen. Diese Möglichkeit steht Ihnen in Kombination mit der Option `f` für einen *Float*-Wert zur Verfügung. Dabei wird der Punkt angegeben, daran erkennt PHP, dass hier die Formatierung der Nachkommastellen durchgeführt werden soll. Über die Zahl hinter dem wird die Länge der Nachkommastellen definiert. Sowohl als auch die Längenangabe **muss** vor dem Optionsbuchstaben `f` stehen, ansonsten ist die Ausgabe fehlerhaft. Ganze Zahlen werden mit so vielen Nullen als Nachkommastellen dargestellt, wie angegeben ist.

Die Darstellung der Nachkommastellen und das Auffüllen von Zeichen kann miteinander kombiniert werden. Die Menge der Nachkommastellen wird hinter dem Punkt angegeben, die Länge, auf die der übergebene Parameter aufgefüllt werden soll, vor dem Punkt. Dort wird sowohl das Füllzeichen als auch die Länge an sich definiert. Dabei ist zu beachten, dass das Dezimalzeichen und die Nachkommastellen mitgezählt werden für die Länge, auf welche die gesamte Ausgabe aufgefüllt werden soll.

Beispiel: nachkommastellen.php

```
<?php
①    $zahl1 = 157.549862;
     $zahl2 = 300;
②    printf("<p>2 Stellen: <b>%.2f</b></p>", $zahl1);
③    printf("<p>4 Stellen: <b>%.4f</b></p>", $zahl1);
④    printf("<p>Ganzzahl: <b>%.2f</b></p>", $zahl2);
⑤    printf("<p>6 Zeichen: <b>%06.2f</b></p>", $zahl1);
⑥    printf("<p>10 Zeichen: <b>%010.2f</b></p>", $zahl1);
⑦    printf("<p>10 Zeichen: <b>%'X10.2f</b></p>", $zahl1);
?>
```

- ① Zuerst werden zwei Variablen definiert, die erste mit einem *float*-Wert, die zweite als Ganzzahl.
- ② Mit der Angabe von .2 legen Sie fest, dass 2 Nachkommastellen dargestellt werden sollen. Beachten Sie, dass `printf()` hier den *float*-Wert aufrundet.
- ③ Hier wird der *float* mit 4 Zeichen hinter dem Komma definiert.
- ④ Der Integer wird hier als *float* dargestellt, die Nachkommastellen sind 0.
- ⑤ Hier soll zusätzlich die gesamte Ausgabe auf 6 Zeichen aufgefüllt werden.



Ausgabe der Beispieldatei „nachkommastellen.php“

Da die Ausgabe bereits 6 Zeichen hat und da der Dezimalpunkt sowie die Nachkommastellen mitgezählt werden, findet kein weiteres Auffüllen statt.

- ⑥ In dieser Zeile wird der *float* auf insgesamt 10 Stellen aufgefüllt
- ⑦ Hier wird die Darstellung ebenfalls bis auf 10 Zeichen erweitert, in dem Fall mit dem Buchstaben X, den Sie mit einem Hochkommata maskieren müssen.

9.2 Zahlen formatieren

Zahlen mit der Funktion `number_format()` formatieren

Float-Werte werden in PHP in der englischen Notation mit dem Punkt als Dezimaltrennzeichen und dem Komma als Tausendertrennzeichen dargestellt. Um die Darstellung von Floats an die deutsche Notation anzupassen, steht Ihnen die Funktion `number_format()` zur Verfügung. Neben dem Zeichen, das als Dezimalzeichen angezeigt werden soll, kann ebenfalls das Tausendertrennzeichen definiert werden.

Syntax der Funktion `number_format()`

```
number_format(Zahl, Stellen, "Nachkomma", "Tausender");
```

- ✓ Der Parameter `Zahl` gibt die Zahl an, die formatiert werden soll.
- ✓ Mit dem Parameter `Stellen` geben Sie die Anzahl der Nachkommastellen an.
- ✓ Hat die Ausgangszahl mehr Nachkommastellen als in `Stellen` angegeben, wird die Zahl mathematisch gerundet.
- ✓ `Nachkomma` erwartet die Angabe des Dezimaltrennzeichens für die Nachkommastellen.
- ✓ Der Parameter `Tausender` legt das Tausendertrennzeichen fest.
- ✓ Falls keine Werte für `Nachkomma` und `Tausender` angegeben werden, werden die Standardwerte `,` als Dezimaltrennzeichen und `,` als Tausendertrennzeichen verwendet.



`number_format()` erwartet bei PHP vor der Version 8 entweder zwei oder vier Parameter. Seit PHP 8 können auch drei Parameter angegeben werden. Falls Sie ein Zeichen als Dezimaltrennzeichen angeben und Ihr PHP-Skript mit älteren PHP-Versionen lauffähig sein soll, müssen Sie dann auch immer ein Zeichen für das Tausendertrennzeichen hinterlegen.

Beispiel: `number_format.php`

①	<?php \$zahl = 23456789.7583; echo "<p>Vorher: \$zahl</p>"; echo "<p>Nachher: \$zahl</p>"; ② echo "<p>" . number_format(\$zahl, 2) . "</p>"; ③ echo "<p>" . number_format(\$zahl, 2, ",", ".") . "</p>"; ④ echo "<p>" . number_format(\$zahl, 2, " ", " ") . "</p>"; ?>
---	--

- ① Es wird eine Variable mit einem `float`-Wert deklariert und initialisiert.
- ② Hier wird `number_format()` ohne Werte für Dezimal- und Tausendertrennzeichen aufgerufen. In der Ausgabe sehen Sie, dass die englischen Trennzeichen verwendet werden und dass die Ausgangzahl gerundet wurde.
- ③ Für die deutsche Darstellung werden hier das Komma `,` als Dezimal-, der Punkt `.` als Tausendertrennzeichen übergeben.
- ④ Eine ebenfalls übliche Darstellung von Zahlen ist die Gruppierung der Ziffern in Dreierblöcken. Dies wird hier erreicht, in dem als 4. Parameter ein Leerzeichen gesetzt wird. Um die Breite der Leerzeichen optisch wahrzunehmen, wurde hier die Schriftart auf *monospace* gesetzt. Bei *monospace*-Schriftarten werden alle Zeichen, egal ob `i`, `m` oder ein Leerzeichen, gleich breit angezeigt.



Ausgabe der Beispieldatei „number_format.php“

9.3 Nach Zeichenketten suchen

Nach einer Zeichenkette (String) suchen

Mit den Funktionen `strstr()` und `stristr()` können Sie nach dem ersten Vorkommen eines bestimmten Strings in einer Zeichenkette suchen. Zurückgeliefert werden die Zeichen ab Vorkommen des Strings bis zum Ende des Suchstrings.

Syntax und Bedeutung der Funktion `strstr()`

- ✓ Der erste Parameter `String` gibt die Zeichenkette an, die durchsucht werden soll.
- ✓ Die Zeichenkette, nach der gesucht werden soll, wird im zweiten Parameter `Suchstring` übergeben.
- ✓ Falls der Suchstring nicht im `String` gefunden wird, liefert die Funktion `FALSE` zurück.
- ✓ Ist die Suche erfolgreich, liefert die Funktion standardmäßig als Rückgabewert eine Zeichenkette **ab** dem Vorkommen des Suchstrings **bis zum Ende** des Strings, in dem gesucht wird.
- ✓ Wird der dritte, optionale Parameter `TRUE` angegeben, wird die Zeichenkette **vor** dem ersten Auftreten des gesuchten Suchstrings zurückgegeben.
- ✓ Die Funktion `strstr()` berücksichtigt die Groß- und Kleinschreibung der Zeichen.
- ✓ Die Funktion `stristr()` sucht entsprechend der Funktion `strstr()`, ignoriert jedoch die Groß- und Kleinschreibung der Zeichenketten.
- ✓ Die Funktion kann direkt mit Zeichenketten: z. B. `strstr("MarioC@herdtex.de", "C@")` oder auch mit Variablen aufgerufen werden.

```
strstr (String, Suchstring [, TRUE / FALSE ]);  
stristr(String, Suchstring [, TRUE / FALSE ]);
```

In der Dokumentation auf <https://www.php.net/> werden als Beispielvariablen `$haystack` und `$needle` (Heuhaufen und Nadel) verwendet. Daran ist schnell zu erkennen, welcher der Suchstring ist und worin gesucht wird. Hilfreich ist das auch deswegen, da in unterschiedlichen PHP-Funktionen die Reihenfolge der Parameter nicht einheitlich ist. Manchmal ist der „Heuhaufen“ der erste Parameter, mitunter aber auch der zweite oder dritte.

Beispiel: `strstr.php`

```
<?php  
①   $string = "MarioC@herdtex.de";  
②   $teil_gross = "C@";  
③   $teil_klein = "c@";  
    echo "<p>Vorgabe ist der String: <strong>$string</strong></p>";  
    echo "<p>strstr():<br>Suche nach $teil_gross ergibt: " .  
          strstr($string, $teil_gross);  
    echo "<br>Suche nach $teil_klein ergibt: " .  
          strstr($string, $teil_klein) . "</p>";  
    echo "<p>stristr():<br>Suche nach $teil_klein ergibt: " .  
          stristr($string, $teil_klein) . "</p>";  
?  
>
```

- ① Die Variable `$string` erhält die Zeichenkette, die durchsucht werden soll („Heuhaufen“).
- ② Die zu suchenden Zeichen werden in der Variablen `$teil_gross` und `$teil_klein` angegeben („Nadel“).
- ③ Mit der Funktion `strstr()` werden die Zeichen der Variablen `$teil_gross` innerhalb der Variablen `$string` gesucht. Zurückgegeben werden die Zeichen ab dem gefundenen Suchstring, also `C@herdtx.de`.

The screenshot shows a browser window with the URL `localhost/herdt/Beispieldatei`. The page title is "Nach Zeichenketten suchen". The content displays the string "Vorgabe ist der String: MarioC@herdtx.de". Below it, two examples are shown:

```
strstr():
Suche nach C@ ergibt: C@herdtx.de
Suche nach c@ ergibt:
```

and

```
stristr():
Suche nach c@ ergibt: C@herdtx.de
```

Anzeige der Beispieldatei „strstr.php“

- ④ Die gleichen Zeichen werden in Kleinbuchstaben gesucht. Es erfolgt keine Ausgabe, da die gesuchte Zeichenfolge nicht gefunden wird, weil `strstr()` (ohne das „i“) die Groß- und Kleinschreibung berücksichtigt.
- ⑤ Mit der Funktion `stristr()` können Sie unabhängig von der Groß- und Kleinschreibung nach einer Zeichenkette suchen. Analog zu ④ wird mit der Funktion `stristr()` die Zeichenkette durchsucht, in diesem Fall auch gefunden und das Ergebnis angezeigt.

Nach einem Zeichen suchen

Syntax und Bedeutung der `strrchr()`-Anweisung

Die `strrchr()`-Anweisung findet das letzte Vorkommen eines Zeichens in einer Zeichenkette und liefert die restliche Zeichenkette ab dem gefundenen Zeichen bis zum Ende des Strings zurück.

- ✓ Im ersten Parameter `String` wird die Zeichenkette übergeben, die durchsucht werden soll. Der zweite Parameter `Zeichen` enthält das zu suchende Zeichen.
- ✓ Die Funktion `strrchr()` beginnt, den String von hinten zu durchsuchen, und findet somit das letzte Vorkommen des gesuchten Zeichens.
- ✓ Die Funktion `strrchr()` liefert als Rückgabewert einen String, falls das Zeichen gefunden wird, sonst `FALSE`.
- ✓ Beachten Sie: Auch wenn nach nur **einem** Zeichen gesucht wird, ist der Rückgabewert bei Sucherfolg meistens eine Zeichenkette.
- ✓ Groß- und Kleinschreibung ist zu beachten.

`strrchr(String, Zeichen);`

Das „`r`“ zwischen `str` und `chr` in `strrchr()` steht für *reverse*. Die Suche erfolgt rückwärts, beginnt somit am Ende des Strings. Die Funktion `strchr()` ohne „`r`“ zwischen „`str`“ und „`chr`“ durchsucht den String von vorne. Das erste Vorkommen eines Zeichens wird ermittelt. Mit einiger Erfahrung können Sie aus der Funktionsbezeichnung die Funktionalität der jeweiligen Funktion ableiten.

Beispiel: strrchr.php

```
<?php
$string = "MarioC@herdtex.de";
echo "<p>Vorgabe ist der String: <strong>$string</strong></p>";
$teil_1 = "r";
$teil_2 = "R";
echo "<p>strrchr():<br>Suche nach $teil_1 ergibt: " .
      strrchr($string,
              $teil_1);
echo "<br>Suche nach $teil_2 ergibt: " . strrchr($string,
              $teil_2) . "</p>";
?>
```



Anzeige der Beispieldatei „strrchr.php“

Nach Phrasen in Zeichenketten suchen

PHP 8 liefert eine neue Funktion, um in einer Zeichenkette zu suchen. Dabei konnte man die eigentliche Funktionalität bereits mit anderen Funktionen erreichen, die Benennung der Funktion ist jedoch intuitiver und nicht so kryptisch, wie einige ältere Funktionen.

Syntax und Bedeutung der Funktion str_contains()

- ✓ Die Funktion str_contains() untersucht die Zeichenkette Heuhaufen nach dem Suchstring Nadel.
- ✓ str_contains() gibt TRUE zurück, falls der Suchstring gefunden wurde, ansonsten FALSE.
- ✓ Die Funktion untersucht **case-sensitiv**, sprich: Es wird Groß- und Kleinschreibung berücksichtigt.
- ✓ Falls als Nadel eine leere Zeichenkette übergeben wird, liefert die Funktion TRUE zurück.

```
str_contains(Heuhaufen, Nadel);
```

Am Anfang und am Ende von Zeichenketten suchen

Ebenfalls neu in PHP 8 sind die Funktionen `str_starts_with()` und `str_ends_with()`. Auch hier haben die Entwickler eine sprechende Benennung gewählt, was den Einsatz im Quelltext sehr leserlich macht.

- ✓ Die Funktionen prüfen, ob eine Zeichenkette Heuhaufen mit einem Suchstring Nadel beginnt bzw. endet.
- ✓ Im Erfolgsfall liefern die Funktionen TRUE zurück, andernfalls FALSE.
- ✓ Die Funktionen untersuchen ebenfalls **case-sensitiv**.
- ✓ Falls als Nadel eine leere Zeichenkette übergeben wird, wird TRUE zurückgeliefert.

```
str_starts_with(Heuhaufen, Nadel);
str_ends_with(Heuhaufen, Nadel);
```

Beispiel: *zeichenketten-durchsuchen.php*

```
<?php
①    $aHeuhaufen = ['Am Anfang ist es noch schwer',
                  'Einfacher wird es am Ende'];
    $aNadel = ['Anfang', 'Ende'];
    foreach ($aHeuhaufen as $sSatz) {
        foreach ($aNadel as $sWort) {
            ④        if (str_contains($sSatz, $sWort)) {
                $sErgebins = 'JA';
            } else {
                $sErgebins = 'NEIN';
            }
            echo "<tr><td><code>str_contains</code>
                  <td>$sSatz<td>$sWort<td>$sErgebins</tr>";
            ⑤        if (str_starts_with($sSatz, $sWort)) {
                $sErgebins = 'JA';
            } else {
                $sErgebins = 'NEIN';
            }
            echo "<tr><td><code>str_starts_with</code>
                  <td>$sSatz<td>$sWort<td>$sErgebins </tr>";
            ⑥        if (str_ends_with($sSatz, $sWort)) {
                $sErgebins = 'JA';
            } else {
                $sErgebins = 'NEIN';
            }
            echo "<tr><td><code>str_ends_with</code>
                  <td>$sSatz<td>$sWort<td>$sErgebins</tr>";
        }
    }
?>
```

- ① Es werden zwei Arrays deklariert und initialisiert. In der Variablen \$aHeuhaufen sind zwei Sätze gespeichert, in \$aNadel zwei Suchworte.
- ② Mit einer `foreach`-Schleife werden die Sätze in \$aHeuhaufen durchlaufen.
- ③ In einer weiteren `foreach`-Schleife, welche die Suchworte \$aNadel durchläuft, werden nun die drei neuen PHP-Zeichenketten-Suchfunktionen verwendet.
- ④ In einer `if`-Abfrage wird mit `str_contains()` geprüft, ob das aktuelle \$sWort in dem aktuellen Satz in \$sSatz enthalten ist und je nach Ergebnis in der Variablen \$sErgebnis JA oder NEIN gespeichert und danach in einer HTML-Tabellenzeile ausgegeben.
- ⑤ Hier wird mit `str_starts_with()` geprüft, ob der Satz mit dem Suchwort beginnt und das Ergebnis wird wie in ④ weiterverarbeitet.
- ⑥ Das gleiche geschieht hier mit `str_ends_with()`, in dem Fall die Prüfung, ob die Zeichenkette mit dem Suchwort endet.

The screenshot shows a browser window with the URL `localhost/herdt/Beispieldateien/Kapitel_0 ...`. The page title is "str_contains(), str_starts_with(), str_ends_with()". Below the title is a table comparing three string search functions across three different sentence contexts: 'Am Anfang ist es noch schwer', 'Einfacher wird es am Ende', and 'Einfacher wird es in der Mitte'.

Funktion	Satz	Suchwort	gefunden?
<code>str_contains</code>	Am Anfang ist es noch schwer	Anfang	JA
<code>str_starts_with</code>	Am Anfang ist es noch schwer	Anfang	NEIN
<code>str_ends_with</code>	Am Anfang ist es noch schwer	Anfang	NEIN
<hr/>			
<code>str_contains</code>	Einfacher wird es am Ende	Anfang	NEIN
<code>str_starts_with</code>	Einfacher wird es am Ende	Anfang	NEIN
<code>str_ends_with</code>	Einfacher wird es am Ende	Anfang	NEIN
<hr/>			
<code>str_contains</code>	Einfacher wird es in der Mitte	Mitte	JA
<code>str_starts_with</code>	Einfacher wird es in der Mitte	Mitte	NEIN
<code>str_ends_with</code>	Einfacher wird es in der Mitte	Mitte	JA

Ausgabe der Beispieldatei „zeichenketten-durchsuchen.php“

9.4 Position und Teil einer Zeichenkette ermitteln

Position eines Zeichens ermitteln

Um die erste bzw. letzte Position eines Zeichens in einer Zeichenkette zu bestimmen, verwenden Sie die Funktionen `strpos()` und `strrpos()`.

Syntax und Bedeutung der Funktion `strpos()`

- ✓ Innerhalb der Klammern werden der String sowie das zu suchende Zeichen angegeben.
- ✓ Optional können Sie den Parameter Start angeben. Hiermit beginnt die Suche nach dem Zeichen nicht am Anfang des Strings, sondern erst an der Stelle Start.
- ✓ Die Funktion `strpos()` gibt die erste Stelle an, an der das gesuchte Zeichen gefunden wurde.

```
strpos(String, Zeichen, [Start]);
strrpos(String, Zeichen, [Start]);
```

- ✓ Liefert die Funktion den Wert 0 zurück, handelt es sich um das erste Zeichen im String. Wird der Wert FALSE zurückgeliefert, befindet sich das Zeichen nicht im angegebenen String.
- ✓ Im Unterschied zur Funktion `strpos()` liefert die Funktion `strrpos()` die letzte Position des Zeichens zurück, da mit der Suche von rechts, also vom Ende des Strings begonnen wird.

Teilstring einer Zeichenkette bestimmen

Mit der Funktion `substr()` können Sie einen Teil einer Zeichenkette extrahieren.

Syntax und Bedeutung der `substr()`-Anweisung

- ✓ Geben Sie im Parameter `String` die auszulesende Zeichenkette und im Parameter `Start` die Startposition an, an der das Auslesen der Zeichen (von links) begonnen werden soll.
- ✓ Wenn Sie einen negativen Wert für den Parameter `Start` angeben, dann beginnt das Auslesen der Zeichen von rechts, also vom Ende der Zeichenkette.
- ✓ Optional können Sie den Parameter `Länge` angeben. Geben Sie keine Länge an, werden die Zeichen vom Startpunkt bis zum Ende des Strings zurückgeliefert.

Beispiel: `strpos.php`

Aus einer E-Mail-Adresse soll die dazugehörige Domain ermittelt werden. Dazu suchen Sie in der E-Mail-Adresse nach dem Zeichen `@` und übergeben die nachfolgenden Zeichen an eine neue Variable. Dieser Zeichenkette werden die Zeichen `https://www.` vorangestellt, um die Web-Adresse zu generieren.

```
<?php
① $string = "webmaster@php.net";
$zeichen = "@";
② echo 'E-Mail-Adresse: ' . $string;
③ $pos = strpos($string, $zeichen);
④ if ($pos === FALSE) {
    echo "<p>Zeichen <strong>$zeichen</strong> konnte nicht
        gefunden werden!</p>";
⑤ } else {
    echo "<p>Wert von \$pos: $pos (Position des
        $zeichen-Zeichen)</p>";
⑥ $webadresse = substr($string, $pos + 1);
$webadresse = "https://www." . $webadresse;
echo "<p>Web-Adresse: <a href=\"$webadresse\">
        $webadresse</a></p>";
}
?>
```

- ① Der Variablen \$string wird die E-Mail-Adresse webmaster@php.net übergeben. Das Zeichen @, nach dem gesucht werden soll, wird der Variablen \$zeichen zugewiesen.
- ② Die E-Mail-Adresse wird zur Kontrolle am Bildschirm ausgegeben.
- ③ Über die Funktion strpos() wird die Position gesucht, an der sich das Zeichen @ befindet. Dieser Wert wird in der Variablen \$pos gespeichert.
- ④ In der if-Anweisung wird abgefragt, ob das Zeichen @ gefunden wurde. Durch den Identisch-Operator == == == wird geprüft, ob der Wert der Variablen \$pos gleich FALSE ist und beide den identischen Datentyp aufweisen.
Diese Identisch-Überprüfung ist notwendig, da ein Treffer an der ersten Stelle den Wert 0 zurückgibt. Bei der Gleichheitsprüfung per == == == werden 0 und FALSE gleich behandelt, sowohl if (\$pos == false) als auch if (\$pos == 0) liefern das gleiche Resultat. Die Abfrage würde hier in den Fehler laufen, obwohl das gesuchte Zeichen an Position 0 gefunden wurde.
- ⑤ Der else-Zweig wird ausgeführt, falls das Zeichen @ gefunden wurde.
- ⑥ Mit der Funktion substr() wird die Zeichenfolge nach dem Zeichen @ in der Variablen \$webadresse gespeichert. Somit haben Sie den Namen der Domain aus der E-Mail-Adresse extrahiert.
- ⑦ Der Domain wird die Zeichenkette https://www. vorangestellt.
- ⑧ Der HTML-Code zum Anzeigen eines Hyperlinks wird erstellt und über den Befehl echo am Bildschirm ausgegeben.



9.5 Zählen innerhalb von Zeichenketten

Länge von Zeichenketten bestimmen

Über strlen() bestimmen Sie die Länge der angegebenen Zeichenkette, z. B. zur Prüfung, ob eine Mindestlänge für ein Eingabefeld in ein Formular eingehalten wurde.

Syntax und Bedeutung der strlen() -Anweisung

- ✓ Innerhalb der Klammern wird der String angegeben, dessen Zeichenlänge ermittelt werden soll.
- ✓ Als Ergebnis erhalten Sie die Anzahl der Zeichen bzw. 0, wenn die Zeichenkette leer ist.

strlen(String);

Anzahl des Vorkommens bestimmen

Mit der Funktion `substr_count()` ermitteln Sie, wie oft ein Suchstring in einer Zeichenkette (String) vorkommt.

Syntax und Bedeutung der `substr_count()`-Anweisung

- ✓ Im Parameter `String` wird die Zeichenkette übergeben, innerhalb derer nach der Häufigkeit des Suchstrings durchsucht werden soll.
- ✓ Im Parameter `Suchstring` wird die Zeichenkette angegeben, von der Sie das Vorkommen im String zählen möchten.
- ✓ Als Rückgabewert erhalten Sie die Anzahl, wie oft der Suchstring im String gefunden wurde.
- ✓ Sich überlappende Suchstrings werden nicht mitgezählt, so wird z. B. der Suchstring `ABCABC` nur einmal in der Zeichenkette `ABCABCABC` gezählt.

`substr_count(String, Suchstring);`

Beispiel: `substr_count.php`

In diesem Beispiel soll kontrolliert werden, ob eine Zeichenkette mindestens einmal das Zeichen @ und mindestens einmal den Punkt . enthält.

```
<?php
① $string = "webmaster@herdtx.de";
② $okay = TRUE;
echo "E-Mail-Adresse: " . $string;
③ if (substr_count($string, "@") == 0) {
    echo "<p>Zeichen <strong>@</strong> konnte nicht gefunden
        werden!</p>";
    $okay = FALSE;
}
④ if (substr_count($string, ".") == 0) {
    echo "<p>Zeichen <strong>.</strong> konnte nicht gefunden
        werden!</p>";
    $okay = FALSE;
}
⑤ if ($okay == TRUE) {
    echo "<p>Die E-Mail-Adresse scheint gültig zu sein.</p>";
}
?>
```

- ① Die Variable `$string` beinhaltet die zu untersuchende E-Mail-Adresse. Sie können sie zum Testen der Abfragen beliebig ändern.
- ② Die Variable `$okay` wird angelegt, um das Testergebnis zu speichern. Zu Beginn wird sie auf den Wert `TRUE` gesetzt.
- ③ Im `if`-Zweig wird über die Funktion `substr_count()` überprüft, ob sich im Wert der Variablen `$string` das Zeichen @ befindet. Ist der Rückgabewert 0, wurde das Zeichen nicht gefunden, so wird eine entsprechende Meldung ausgegeben. Der Wert der Variable `$okay` wird auf `FALSE` gesetzt.

- ④ In der zweiten if-Anweisung wird nach dem Zeichen `@` gesucht. Wird dieses nicht gefunden, wird analog zu dem vorhergehenden Zweig eine Meldung ausgegeben und der Wert der Variable `$okay` auf `FALSE` gesetzt.
- ⑤ Die letzte Bedingung prüft, ob die Variable `$okay` den Wert `TRUE` hat und somit die E-Mail-Adresse die Zeichen `@` und `.` enthält. Im Browser wird eine Erfolgsmeldung ausgegeben.



Anzeige Beispieldatei „substr_count.php“



Diese Überprüfung einer E-Mail-Adresse dient lediglich zur Verdeutlichung der Funktion `substr_count()`. Die Kriterien für eine gültige E-Mail-Adresse sind weitaus umfangreicher. Für den Einsatz in der Praxis ist dies kein ausreichender Ansatz.

9.6 Zeichenketten vergleichen

Zeichenketten binär miteinander vergleichen

Mit den Funktionen `strcmp()` und `strcasecmp()` können Sie zwei Zeichenketten miteinander vergleichen.

Syntax und Bedeutung der `strcmp()`-Anweisung

- ✓ Die Funktion vergleicht die einzelnen Zeichen von `String1` und `String2` anhand ihres ASCII-Codes. Beispielsweise ist das Zeichen `A` (ASCII-Code: 65) kleiner als das Zeichen `B` (ASCII-Code: 66).
- ✓ Die zwei Zeichenketten, die binär miteinander verglichen werden sollen, werden innerhalb der Klammern angegeben.
- ✓ Bei dem Vergleich wird nach Groß- und Kleinschreibung unterschieden.
- ✓ Als Rückgabe erhalten Sie einen Wert `< 0`, wenn `String1 < String2` ist, oder einen Wert `> 0`, wenn `String1 > String2` ist, oder den Wert `0`, wenn beide Zeichenketten gleich sind.
- ✓ Die Funktion `strcasecmp()` lässt im Unterschied zur Funktion `strcmp()` die Groß- und Kleinschreibung außer Acht.

```
strcmp(String1, String2);
strcasecmp(String1, String2);
```

9.7 Zeichenketten modifizieren

Mit folgenden Funktionen können Sie Zeichenketten unterschiedlich umwandeln, wie z. B. eine Zeichenkette wiederholen, Leerzeichen aus einer Zeichenkette entfernen oder Zeichenketten in Groß- bzw. Kleinschreibung umändern.

Zeichenketten wiederholen

Syntax und Bedeutung der `str_repeat()`-Anweisung

- ✓ Der angegebene String wird mehrmals aneinandergesetzt.
- ✓ Wie oft die Zeichenkette wiederholt werden soll, geben Sie über den Parameter Anzahl an.
Beispiel: Eine mehrfache Ausgabe am Bildschirm erreichen Sie beispielsweise über die Angabe von `echo str_repeat('-', 80);`.

```
str_repeat(String, Anzahl);
```

Leerraum oder andere Zeichen entfernen

Syntax und Bedeutung der Anweisungen

- ✓ Werden diese drei Funktionen ohne den optionalen Parameter Zeichenliste aufgerufen, werden folgende Leer- und Steuerungszeichen (Escape-Sequenzen) entfernt:
 - ✓ "\n" (Zeilenumbruch)
 - ✓ "\t" (Tabulator)
 - ✓ "\r" (Wagenrücklaufzeichen)
 - ✓ "\0" (NULL-Byte)
 - ✓ "\v" (vertikaler Tabulator)
- ✓ Die Funktion `trim()` entfernt angegebene Zeichen bzw. die Leerräume am Anfang **und** Ende einer Zeichenkette.
- ✓ `ltrim()` hat dieselbe Funktion, entfernt diese Zeichen jedoch nur **am Anfang** einer Zeichenkette (das „l“ steht hier für *left*).
- ✓ Die Funktion `rtrim()` löscht angegebene Zeichen oder Leerräume **am Ende** („r“ vor dem `trim` steht für *right*) eines Strings.
- ✓ Über den 2. optionalen Parameter Zeichenliste können Sie eine Liste von weiteren Zeichen angeben, die ebenfalls gelöscht werden sollen.

```
trim (String[, Zeichenliste]);
ltrim(String[, Zeichenliste]);
rtrim(String[, Zeichenliste]);
```

`trim('https://www.php.net/ ', '/')` löscht nicht nur die Leerzeichen **vor** dem `https`, auch der Slash **am Ende** der URL wird hiermit entfernt.

Mit der Anweisung `trim($satz, '.,!?:')` würden alle Satzzeichen am Anfang und am Ende des Wertes `$satz` entfernt. Dabei wird jedes Zeichen einzeln berücksichtigt, die Kombination der Zeichen spielt keine Rolle.

- ✓ Diese Funktionen bieten sich an, um Leerzeichen bei der Eingabe von Formularen zu entfernen oder wenn Sie Textdateien auslesen, in denen Zeilenumbrüche sind.

Buchstaben innerhalb einer Zeichenkette umwandeln

Syntax und Bedeutung der Anweisungen

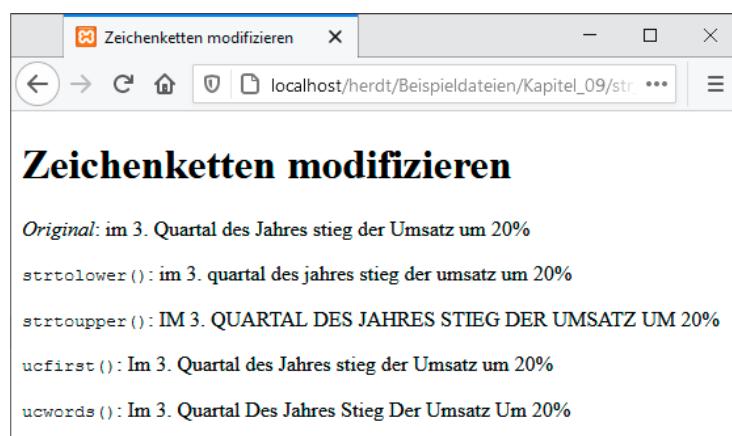
- ✓ Die Funktion `strtolower()` wandelt alle Zeichen der Zeichenkette in Kleinbuchstaben um.
- ✓ Die Funktion `strtoupper()` wandelt alle Zeichen der Zeichenkette in Großbuchstaben um.
- ✓ Mit der Funktion `ucfirst()` setzen Sie den ersten Buchstaben der Zeichenkette in einen Großbuchstaben um, sofern es sich hierbei um ein Zeichen des Alphabets handelt (uc am Anfang des Funktionsnamens steht hier für *UpperCase*).
- ✓ Die Funktion `ucwords()` setzt den ersten Buchstaben aller Worte in einen Großbuchstaben um. Auch hierbei muss es sich jeweils um ein alphabetisches Zeichen handeln.

```
strtolower(String);
strtoupper(String);
ucfirst(String);
ucwords(String);
```

Beispiel: str_umwandeln.php

```
<?php
① $text = "im 3. Quartal des Jahres stieg der Umsatz um 20%";
  echo "<p><em>Original</em>: $text</p>";
② $kl = strtolower($text);
  echo "<p><code>strtolower()</code>: $kl</p>";
③ $gr = strtoupper($text);
  echo "<p><code>strtoupper()</code>: $gr</p>";
④ $uf = ucfirst($text);
  echo "<p><code>ucfirst()</code>: $uf</p>";
⑤ $uw = ucwords($text);
  echo "<p><code>ucwords()</code>: $uw</p>";
?
>
```

- ① Der Variablen `$text` wird eine Zeichenkette zugewiesen, die über die verschiedenen Funktionen modifiziert werden soll.
- ② Der Beispieltext wird mit der Funktion `strtolower()` in Kleinbuchstaben umgewandelt.
- ③ Der Beispieltext wird mit der Funktion `strtoupper()` in Großbuchstaben umgewandelt.
- ④ Das erste Zeichen des übergebenen Wertes, also hier des ganzen Satzes, und damit nur das des ersten Wortes des Beispieltextes, wird mit der Funktion `ucfirst()` in einen Großbuchstaben umgewandelt.
- ⑤ Das erste Zeichen jedes Wortes des Beispieltextes wird mit der Funktion `ucwords()` in einen Großbuchstaben umgewandelt.



Ansicht der Beispieldatei „str_umwandeln.php“

Zeichen oder Zeichenfolgen innerhalb einer Zeichenkette austauschen

Manchmal ist es notwendig, spezielle Zeichen oder Zeichenfolgen auszutauschen, z. B. bei unerwünschten Sonderzeichen oder um Zeichenfolgen eine spezielle Formatierung zuzuweisen.

Syntax und Bedeutung der Funktion `strtr()`

```
strtr(String, Zeichen_von [, Zeichen_nach]);
```

- ✓ Die `strtr()`-Funktion bearbeitet den `String`, indem Zeichen aus `Zeichen_von` in die entsprechenden Zeichen aus `Zeichen_nach` umgesetzt werden. Als Ergebnis erhalten Sie die umgewandelte Zeichenkette.
- ✓ Die Zeichen, die ausgetauscht werden sollen, geben Sie bei der `strtr()`-Funktion in Anführungszeichen hintereinander an (z. B. `strtr("Buch", "u", "a") ;`).
- ✓ Es wird jedes einzelne Zeichen in `Zeichen_von` durch das Zeichen an entsprechender Stelle in `Zeichen_nach` ersetzt. Beispiel: die Anweisung `strtr("Buch", "ab", "cd") ;` ersetzt jedes a mit einem c und jedes b mit einem d, unabhängig davon, ob die Zeichen zusammenstehen.
- ✓ Die Länge der Zeichen in `Zeichen_nach` muss der Länge von `Zeichen_von` entsprechen. Sind `Zeichen_von` und `Zeichen_nach` von unterschiedlicher Länge, so werden die überzähligen Zeichen ignoriert, unabhängig davon, welcher Parameter weniger Zeichen hat. Beide Parameter werden zuvor auf die gleiche Länge gekürzt.
- ✓ Alternativ kann `strtr()` mit nur zwei Parametern aufgerufen werden. In dem Fall muss der zweite Parameter ein assoziatives Array sein. Der Schlüssel eines Array-Eintrags entspricht dann dem `Zeichen_von` und wird mit dem Wert des Array-Eintrags ausgetauscht, z. B. `strtr("Buch", array("u" => "esu")) ;`. Beim Aufruf mit einem Array muss die gemeinsame Länge der Parameter nicht übereinstimmen, es wird jeder gefundene Schlüssel mit dem vollständigen Wert ersetzt.

Syntax und Bedeutung der Funktion `str_replace()`

```
str_replace(String_von, String_nach, String);
```

- ✓ Über die Funktion `str_replace()` können Sie ganze Wörter oder Textpassagen vertauschen. Diese müssen nicht die gleiche Länge haben.
- ✓ Anders als bei `strtr()` wird hier nur eine ganze Zeichenkette ersetzt, nicht die einzelnen Zeichen der Parameter.
- ✓ Der erste Parameter `String_von` ist die Suchphrase, der zweite Parameter `String_nach` ist der Wert, den die Suchphrase ersetzen soll. Der dritte Parameter `String` ist die Zeichenkette, auf den dieses Ersetzen angewendet werden soll.
- ✓ Die Zeichen, die die Fundstellen ersetzen, können Buchstaben oder Zahlen sowie vollständige HTML-Tags beinhalten.

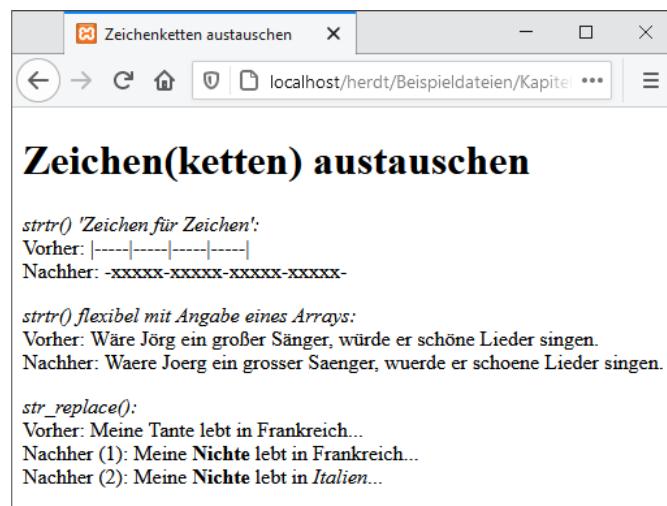
Bei der Angabe des zu ersetzenen Strings wird die Groß- und Kleinschreibung beachtet.

Beispiel: str_tauschen.php

```

<?php
    echo "<p><em>strr() 'Zeichen für Zeichen':</em>" ;
①   $string = "-----|-----|-----|-----|";
    echo "<br>Vorher: $string";
②   echo "<br>Nachher: " . strr($string, '-' | ', 'x-') . "</p>";
    $string = "Wäre Jörg ein großer Sänger, würde er schöne Lieder
              singen.";
    echo "<p><em>strr() flexibel mit Angabe eines Arrays:</em>" ;
    echo "<br>Vorher: $string <br>";
③   $umlaute = array("ä" => "ae", "ö" => "oe", "ü" => "ue", "ß" => "ss");
    echo "Nachher: " . strr($string, $umlaute) . "</p>";
    echo "<p><em>str_replace():</em>" ;
    $string = "Meine Tante lebt in Frankreich...";
    echo "<br>Vorher: $string";
④   $string = str_replace("Tante", "<strong>Nichte</strong>", $string);
    echo "<br>Nachher (1): " . $string;
⑤   $string = str_replace("Frankreich", "<em>Italien</em>", $string);
    echo "<br>Nachher (2): " . $string . "</p>";
?
  
```

- ① Die Variable \$string wird mit Zeichen gefüllt.
- ② Die Funktion strr() wird aufgerufen, das Zeichen **-** wird durch **X** und das Zeichen **I** wird durch **-** ersetzt. Das Ergebnis wird ausgegeben.
- ③ Ein Array wird definiert, das Zeichenkettenpaare enthält.
- ④ Die Schlüsselwerte werden in einem Beispielsatz gesucht und durch die dazugehörigen Werte des entsprechenden Array-Elements ausgetauscht.
- ⑤ Mit der Funktion str_replace() soll das Wort *Tante* durch das fett gedruckte Wort *Nichte* ersetzt werden.
- ⑥ Die Funktion str_replace() wird ein weiteres Mal aufgerufen. Auch in diesem Fall wird eine Zeichenkettenersetzung dazu verwendet, um den Ländernamen auszutauschen und zusätzlich das HTML-Tag em einzufügen.



Anzeige der Beispieldatei „str_tauschen.php“

9.8 Mit Arrays und Zeichenketten arbeiten

Eine Zeichenkette in ein Array umwandeln

Wenn Sie eine Zeichenkette an einem bestimmten Trennzeichen aufteilen möchten und die Teilstrings, die Sie dadurch erhalten, in einem Array speichern möchten, verwenden Sie die Funktion `explode()`.

Syntax und Bedeutung der Funktion `explode()`

```
explode(Trennzeichen, String [, Limit]);
```

- ✓ Die Funktion `explode()` ermöglicht es, eine Zeichenkette anhand eines Trenzeichens, das Sie selbst bestimmten können, aufzuteilen und als einzelne Elemente in einem indizierten Array zu speichern. So können Sie beliebige Wertelisten mit demselben Trennzeichen, z. B. alle Wörter eines Satzes per Trennung über das Leerzeichen, einfach in einem Array speichern.
- ✓ Mit der optionalen Angabe von `Limit` können Sie die maximale Anzahl der zurückgelieferten String-Teile angeben. Besitzt eine Zeichenkette mehr Trennzeichen, als im `Limit` angegeben wird, enthält das letzte Element des Arrays den Rest der Zeichenkette.
- ✓ Wird der Parameter `Limit` negativ angegeben, werden alle String-Teile zurückgegeben, mit Ausnahme der der Zahl `Limit` entsprechenden Anzahl von rechts gezählt.
`explode(", " , "a,b,c,d,e" , -2);` beispielsweise liefert ein Array ohne die letzten beiden Elemente, die mit einem Komma getrennt sind, also ohne d und e.

Aus einem Array eine Zeichenkette erzeugen

Mit der Funktion `implode()` können Sie die Elemente eines Arrays zu einer Zeichenkette verbinden.

```
implode(Verbindungszeichen, Array);
```

- ✓ Mit dieser Funktion können Sie die einzelnen Elemente eines Arrays in einer Zeichenkette zusammenführen.
- ✓ Zwischen den einzelnen Elementen steht jeweils das Verbindungszeichen, das Sie selbst festlegen.

Beispiel: `ex-implode.php`

In diesem Beispiel werden die Funktionen `explode()` und `implode()` verwendet.

①	<pre><?php \$vorgabestring = "Elstar; Gala; Jonagold; Boskoop; Delicious"; echo "<p>Vorgabe-Zeichenkette: \$vorgabestring</p>"; echo "<p>explode()</p>";</pre>
---	---

```

② $ausgabe = explode(";", $vorgabestring);
③ $laenge = count($ausgabe);
echo "<p>";
④ for ($i = 0; $i < $laenge; $i++) {
    echo "Suchstring $i: $ausgabe[$i]<br>";
}
echo "</p>";
echo "<pre>";
⑤ print_r($ausgabe);
echo "</pre>";
echo "<p><strong>implode()</strong></p>";
⑥ $ergebnis = implode(" * ", $ausgabe);
echo "<p>Die Zeichenkette lautet: <br>$ergebnis</p>";
?>

```

- ① Der Variablen \$vorgabestring wird eine Zeichenkette mit Begriffen zugewiesen, die durch Semikolons voneinander getrennt sind.
- ② Mit der Funktion explode() wird diese Zeichenkette am Semikolon aufgeteilt und in der Variablen \$ausgabe als indiziertes Array gespeichert. Dabei werden Leer- und Steuerungszeichen wie z. B. \n von den einzelnen Strings entfernt. Das Trennzeichen selbst wird nicht mit in das Ergebnis-Array übernommen.
- ③ Mithilfe der Array-Funktion count() wird die Länge des Arrays, also die Anzahl der Array-Elemente, festgestellt.
- ④ Die Elemente des Arrays werden einzeln mit einer for-Schleife am Bildschirm ausgegeben.
- ⑤ Der Inhalt des Arrays \$ausgabe wird zusätzlich mit dem speziellen Befehl print_r() am Bildschirm ausgegeben. Wird print_r() für ein Array aufgerufen, so werden die Indizes und Werte des Arrays angezeigt.
- ⑥ Der Variablen \$ergebnis werden mit der Funktion implode() die einzelnen Array-Elemente des Arrays \$ausgabe als Zeichenkette, getrennt durch die Zeichenkette " * ", zugewiesen und ausgegeben.

Vorgabe-Zeichenkette: Elstar; Gala; Jonagold; Boskoop; Delicious

explode()

Suchstring 0: Elstar
Suchstring 1: Gala
Suchstring 2: Jonagold
Suchstring 3: Boskoop
Suchstring 4: Delicious

Array

```
( [0] => Elstar
  [1] => Gala
  [2] => Jonagold
  [3] => Boskoop
  [4] => Delicious
)
```

implode()

Die Zeichenkette lautet:
Elstar * Gala * Jonagold * Boskoop * Delicious

Anzeige der Beispieldatei „ex-implode.php“

9.9 Übungen

Übung 1: Mit Zeichenkettenfunktionen arbeiten

Level		Zeit	ca. 15 min
Übungsinhalte	<ul style="list-style-type: none"> ✓ Zeichenkettenformatierung mit <code>printf()</code> ✓ Zeichenketten mit <code>explode()</code> in Arrays umwandeln ✓ Zeichenketten mit <code>implode()</code> aus Arrays generieren ✓ Zeichenketten mit <code>str_replace()</code> austauschen 		
Übungsdatei	--		
Ergebnisdatei	<code>uebung.php</code>		

1. Nach einer Berechnung erhalten Sie eine Fließkommazahl mit diversen Nachkommastellen, z. B. 78,123456789. Formatieren Sie die Ausgabe, sodass drei Stellen vor dem Komma und fünf Nachkommastellen ausgegeben werden.

2. Gegeben sind folgende Variablen:

```
$string1 = "Beachten Sie das Angebot für die ";
$string2 = "folgende Kalenderwoche: ";
$string3 = " ";
$string4 = "Bananen, 5 Kilo für nur 5.- Euro!";
```

Geben Sie die Variablen mithilfe der Funktion `printf()` in einer formatierten Zeichenkette aus.

Setzen Sie hierfür die Länge der Variablen `$string3`, die mit dem festgelegten Zeichen gefüllt wird, auf 5 Zeichen. Trennen Sie die vier Variablen mit der Zeichenkette voneinander ab.

3. Fügen Sie die einzelnen Variablen in einer neuen Variablen `$string` aneinander. Die Zeichenkette, die in der Variablen `$string` gespeichert wurde, soll mithilfe der Funktion `explode()` am Zeichen " " (Leerstelle) getrennt werden und die einzelnen Elemente des entstehenden Arrays sollen ausgegeben werden. Anschließend verwenden Sie die Funktion `implode()`, um das Array in eine Zeichenkette zu speichern. Verwenden Sie als Trennzeichen das Zeichen und geben Sie das Ergebnis aus.

4. Ersetzen Sie die Zeichenkette *das Angebot* durch die fett gedruckte Zeichenkette **unser Sonderangebot** und speichern Sie das Ergebnis in der Variablen `$string5`. Tauschen Sie die Zeichenkette *Bananen* durch die Zeichenkette *Alle Obstsorten* aus, speichern Sie das Ergebnis in der Variablen `$string6`. Geben Sie das Ergebnis im Browser aus.

Übung 2: Suche nach der „Nadel im Heuhaufen“

Level		Zeit	ca. 15 min
Übungsinhalte	<ul style="list-style-type: none"> ✓ Arbeiten mit Formularen ✓ Textsuche per substr_count() ✓ Zeichenketten mit str_replace() austauschen 		
Übungsdatei	--		
Ergebnisdatei	<i>textsuche.php</i>		

1. Entwerfen Sie eine Datei (*textsuche.php*) mit einem einfachen HTML-Formular, in das Sie einen Beispieltext in ein mehrzeiliges Eingabefeld und einen Suchbegriff in ein Eingabefeld eintragen können. Die Auswertung der Suche soll in derselben Datei geschehen und Informationen über die Anzahl der Treffer beinhalten (Funktion substr_count()), wenn der Suchbegriff eingegeben wurde.
2. Um eine Fehlermeldung zu vermeiden, prüfen Sie per isset(), ob die \$_POST-Variable für den Suchbegriff vorhanden ist.
3. Zudem soll der durchsuchte Text mit markierten Fundstellen unterhalb des Formulars ausgegeben werden (Funktion str_replace()).

Orientieren Sie sich mit Ihrer Lösung an der nachfolgenden Abbildung einer Beispilllösung.

The screenshot shows a web browser window titled "Übung Kapitel 9 - Textsuche". The address bar shows "localhost/herdt/Ergebnisdateien/Kapitel_09/textsuche.php". The main content area has a heading "Begriff in einer Textpassage suchen". Below it is a text area containing two paragraphs about gummy bears. A "Originaltext:" label is followed by a scrollable text area. Below that is a "Suche nach:" input field and a "Zeichenkette suchen" button. The text area below the search results contains highlighted matches for "Gummibärchen".

Freilebende Gummibärchen gibt es nicht. Man kauft sie in Packungen an der Kinokasse. Dieser Kauf ist der Beginn einer fast erotischen und sehr ambivalenten Beziehung Gummibärchen-Mensch. Zuerst genießt man. Dieser Genuss umfasst alle Sinne.

Man wühlt in den Gummibärchen, man fühlt sie. Gummibärchen haben eine Konsistenz wie weichgekochter Radiergummi. Die Tastempfindung geht auch ins Sexuelle. Das bedeutet nicht unbedingt, daß das Verhältnis zum Gummibärchen ein geschlechtliches wäre, denn prinzipiell sind diese geschlechtsneutral.

Originaltext:

Suche nach:

Zeichenkette suchen

Suche nach "Gummibärchen": 6 Mal gefunden.

Freilebende Gummibärchen gibt es nicht. Man kauft sie in Packungen an der Kinokasse. Dieser Kauf ist der Beginn einer fast erotischen und sehr ambivalenten Beziehung Gummibärchen-Mensch. Zuerst genießt man. Dieser Genuss umfasst alle Sinne.

Man wühlt in den Gummibärchen, man fühlt sie. Gummibärchen haben eine Konsistenz wie weichgekochter Radiergummi. Die Tastempfindung geht auch ins Sexuelle. Das bedeutet nicht unbedingt, daß das Verhältnis zum Gummibärchen ein geschlechtliches wäre, denn prinzipiell sind diese geschlechtsneutral.

Nun sind Gummibärchen weder wabbelig noch zäh; sie stehen genau an der Grenze. Auch das macht sie spannend. Gummibärchen sind auf eine aufreizende Art weich. Und da sie weich sind, kann man sie auch ziehen. Ich mache das sehr gerne.

Ich sitze im dunklen Kino und ziehe meine Gummibärchen in die Länge, ganz ganz langsam. Man will sie nicht kaputt machen, und dann siegt doch die Neugier, wieviel Zug so ein Bärchen aushält. (Vorstellbar sind u.a. Gummibärchen-Expander für Kinder und Genesende).

Textsuche in einem Beispieltext

10

Datum und Uhrzeit

 **Beispieldateien:** Dateien aus Ordner *Kapitel_10*

10.1 Datum und Zeit ermitteln

In vielen PHP-Skripten sind Datums- und Zeitberechnungen notwendig. PHP verwendet dabei das Datum und die Uhrzeit des Servers, auf dem das Programm läuft.

Datum und Zeit auslesen

Informationen zu Datum und Zeit erhalten Sie über die Funktion `getdate()`. Diese Funktion liefert das Ergebnis in einem assoziativen Array zurück.

Syntax der `getdate()`-Anweisung

- ✓ `getdate()` gibt ein assoziatives Array zurück.
Über von PHP vordefinierte Schlüssel können Sie auf bestimmte Zeit- bzw. Datumsinformationen zugreifen.
Die nachfolgende Tabelle gibt Ihnen einen Überblick, welche Werte im Rückgabe-Array zu finden sind.
- ✓ `getdate()` kann ohne Parameter aufgerufen werden. In dem Fall erhalten Sie die aktuellen Datums- und Zeitwerte des Webservers, also die Informationen zum Zeitpunkt des Aufrufs.
- ✓ `getdate()` kann optional mit einem Parameter `Zeitstempel` aufgerufen werden.
Der Parameter `Zeitstempel` erwartet die Anzahl der Sekunden, die seit dem 01.01.1970 vergangen sind. Sämtliche Zeitangaben in PHP beruhen auf diesem Datum, das auch als Beginn der UNIX-Epoche bezeichnet wird. Aus diesem Grund wird der Zeitstempel als **UNIX-Timestamp** oder kurz **Timestamp** bezeichnet. `getdate()` gibt dann ein Array zurück, in dem die Werte, bezogen auf den Timestamp, enthalten sind.

`getdate([Zeitstempel]);`

Schlüssel	Erklärung	Rückgabewerte
seconds	Sekunde der aktuellen Uhrzeit	0 bis 59
minutes	Minute der aktuellen Uhrzeit	0 bis 59
hours	Stunde der aktuellen Uhrzeit	0 bis 23
mday	Tag des aktuellen Monats	1 bis 31
wday	Numerischer Tag der Woche	0 = Sonntag, 1 = Montag, ... 6 = Samstag
mon	Monat als Zahl	1 = Januar, 2 = Februar, ... 12 = Dezember
year	Jahreszahl	1970 bis 2038
yday	Numerischer Tag des Jahres	0 bis 365
weekday	Ausgeschriebener Wochentag (in Englisch)	Sunday, Monday, ..., Saturday
month	Ausgeschriebener Monat (in Englisch)	January, February, ..., December
0	Sekunden seit 01.01.1970	0 bis ...

Beispiel: *aktuell.php*

Die Informationen im Rückgabe-Array von der Funktion `getdate()` zum jetzigen Zeitpunkt (ohne Aufruf eines Zeitstempels) sollen angezeigt werden. Dazu wird jedes Element des assoziativen Arrays angesprochen.

```
<?php
① $jetzt = getdate();
② echo "<pre>";
③ print_r($jetzt);
④ echo "</pre>";
⑤ echo "<p>Stunde: " . $jetzt["hours"];
⑥ echo "<br>Minute: " . $jetzt["minutes"];
⑦ echo "<br>Sekunde: " . $jetzt["seconds"];
⑧ echo "<br>Tag der Woche: " . $jetzt["wday"] . " = " . $jetzt["weekday"];
⑨ echo "<br>Tag des Monats: " . $jetzt["mday"];
⑩ echo "<br>Tag des Jahres: " . $jetzt["yday"];
⑪ echo "<br>Monat: " . $jetzt["mon"] . " = " . $jetzt["month"];
⑫ echo "<br>Jahr: " . $jetzt["year"];
⑬ echo "<br>Zeitstempel: " . $jetzt["0"] . "</p>";
?>
```

- ① Über die Funktion `getdate()` wird die aktuelle Zeit- und Datuminformation des Web-servers ausgelesen und in der Variablen `$jetzt` gespeichert. Dabei handelt es sich um ein assoziatives Array.
- ② Mithilfe der Funktion `print_r()` wird die Array-Variablen `$jetzt` im Browser ausgegeben.

- ③ Die Stunden-, Minuten- und Sekundenwerte der aktuellen Zeit werden über die entsprechenden, vordefinierten Schlüssel im Array `$jetzt` angesprochen und ausgegeben.
- ④ Der Wert des Array-Eintrags `$jetzt["wday"]` liefert einen numerischen Wert, welcher den Wochentag repräsentiert, in diesem Fall der Wert 5 für Freitag (Friday). Beachten Sie, dass die Zählung der Wochentage bei 0 = Sonntag beginnt. Hinter der Zahl für den Wochentag wird über den Array-Schlüssel `weekday` der Name des Wochentags ausgegeben. Mit der PHP-Standardinstallation erhalten Sie die englischen Bezeichnungen.
- ⑤ Über den Schlüssel `mday` erhalten Sie den Tag des Monats als Zahlenwert.
- ⑥ Der Wert `$jetzt["yday"]` liefert einen numerischen Wert. Dieser entspricht dem n -ten Tag im Jahr. Auch hier beginnt die Zählung bei 0, der 1. Januar hat also für den Schlüssel `yday` den Wert 0.
- ⑦ Über die Schlüssel `mon` und `month` erhalten Sie den Zahlenwert sowie den englischen Namen des Monats.
- ⑧ Hiermit erhalten Sie das Jahr des aktuellen Datums.
- ⑨ Als letzte Ausgabe wird der aktuelle Zeitstempel in Sekunden seit dem 01.01.1970 ausgegeben. Falls Sie `getdate()` mit einem Zeitstempel aufgerufen haben, finden Sie den Wert hier wieder.

```

Array
(
    [seconds] => 27
    [minutes] => 37
    [hours] => 10
    [mday] => 15
    [wday] => 5
    [mon] => 1
    [year] => 2021
    [yday] => 14
    [weekday] => Friday
    [month] => January
    [0] => 1610703447
)

Stunde: 10
Minute: 37
Sekunde: 27
Tag der Woche: 5 = Friday
Tag des Monats: 15
Tag des Jahres: 14
Monat: 1 = January
Jahr: 2021
Zeitstempel: 1610703447

```

Daten des 15.01.2021 um 10:37:27 Uhr

10.2 Datum und Zeit formatieren

Mit der `date()`-Funktion können Sie ein Datum für die Ausgabe formatieren.

Syntax und Bedeutung der `date()`-Anweisung

- ✓ Die `date()`-Anweisung erwartet als Parameter Formatanweisungen, welche Datumssegmente angezeigt werden sollen. In der nachfolgenden Tabelle finden Sie eine Übersicht der wichtigsten Angaben und deren Auswirkungen.
- ✓ Der Parameter Format wird als Zeichenkette in Anführungszeichen `" "` angegeben.

`date(Format [, Zeitstempel]);`

- ✓ In der Zeichenkette können Sie Punkte `.`, Doppelpunkte `:` oder Leerzeichen verwenden, welche in Ihrer Formatierung dargestellt werden sollen. Aber auch Buchstaben und andere Zeichen sind möglich. Die Buchstaben, die als Formatanweisung in PHP implementiert sind, werden „übersetzt“, nicht bekannte Buchstaben bleiben als solche erhalten.
- ✓ `date()` kann mit nur dem ersten Parameter der Formatanweisungen aufgerufen werden. In dem Fall erhalten Sie den formatierten Datumsstring des aktuellen Zeitpunkts zurück.
- ✓ Der optionale zweite Parameter `Zeitstempel` ermöglicht Ihnen, ein spezielles Datum anzugeben. Dieser Zeitstempel entspricht den Sekunden seit dem 01.01.1970.

Formatanweisungen (Auswahl)

Von einer langen Liste möglicher Formatanweisungen, die Sie im Parameter `Format` angeben können, finden Sie die wichtigsten in nachfolgender Tabelle. Über die einzelnen Format-Zeichen steuern Sie, welche Elemente des Datums genutzt und in welchem Format diese formatiert werden sollen.

Eine vollständige Übersicht aller Format-Zeichen finden Sie unter <https://www.php.net/manual/de/function.date.php>.

Format-Zeichen	Resultat	Beispiel
<code>d</code>	Tag des Monats, zweistellig und mit führender Null	01 bis 31
<code>G</code>	Stunde im 24-Stunden-Format ohne führende Null	0 bis 23
<code>H</code>	Stunde im 24-Stunden-Format mit führender Null	00 bis 23
<code>i</code>	Minuten mit führender Null	00 bis 59
<code>j</code>	Tag des Monats ohne führende Null	1 bis 31
<code>m</code>	Zahl des Monats mit führender Null	01 bis 12
<code>n</code>	Zahl des Monats ohne führende Null	1 bis 12
<code>r</code>	Formatierte Ausgabe nach RFC 822/2822	Mon, 15 Mar 2021 10:31:13 +0100
<code>s</code>	Sekunden mit führender Null	00 bis 59
<code>Y (großes Y)</code>	Jahr als vierstellige Zahl	2021
<code>y (kleines y)</code>	Jahr als zweistellige Zahl	21

Beispiel: date.php

In diesem Beispiel sollen die verschiedenen Angaben der Formatanweisungen sichtbar gemacht werden.

```
<?php
① echo "<p>" . date("d.m.y");
② echo "<br>" . date("d.m.Y", time() + 86400);
③ echo "<br>Tag: " . date("d.m.Y") . ", Uhrzeit: " .
    date("H:i:s");
④ echo "<br>" . date("j.n.y");
⑤ echo "<br>" . date("r") . "</p>";
?>
```

- ① Mit der Formatanweisung `d.m.y` werden Tag, Monat und Jahr jeweils zweistellig formatiert. Über die `echo`-Anweisung geben Sie die formatierte Zeichenkette im Browser aus.
- ② In diesem Funktionsaufruf mit der Formatanweisung `d.m.Y` wird die Jahresangabe vierstellig dargestellt. Als zweiter Parameter wird hier ein Zeitstempel übergeben, welcher im Funktionsaufruf berechnet wird.



Anzeige der Beispieldatei „date.php“

Über `time()` (vgl. Abschnitt 10.5) wird der aktuelle Zeitstempel ermittelt, zu diesem Wert werden 86400 Sekunden hinzugefügt. 86400 ist das Ergebnis aus $24 * 60 * 60$ (Stunden * Minuten * Sekunden), also der Sekundenanzahl eines Tages. Der übergebene Zeitstempel entspricht also genau einem Tag in der Zukunft. Um den Aufruf nachvollziehbarer zu gestalten, können Sie auch `date ("d.m.Y", time() + 24 * 60 * 60)` im PHP-Code schreiben. Da Punkt- vor Strichrechnung gilt, werden zuerst die Sekunden pro Tag berechnet und dann zum aktuellen Zeitstempel hinzugefügt.

- ③ Hier werden zusätzlich zu Datum und Zeit weitere Zeichenketten ausgegeben.
- ④ Die Funktion `date()` wird aufgerufen. Die Ausgabe von Tag und Monat erfolgt ohne führende Nullen, die des Jahres erfolgt zweistellig.
- ⑤ Die Anweisung `r` gibt standardmäßig das Datum und die Uhrzeit nach den im Standard RFC 822/2822 festgelegten Regeln aus. Diese Formatierung wird beispielsweise als Angabe von Uhrzeit und Datum im Mailheader, dem nicht sichtbaren Kopfbereich einer E-Mail, verwendet.

10.3 Datumsangabe an Sprache anpassen

Englische Monatsbezeichnungen manuell übersetzen

Bei der Ermittlung des aktuellen Datums erhalten Sie die englischen Tages- und Monatsbezeichnungen. Auf einer deutschsprachigen Webseite soll das Datum jedoch in deutscher Sprache ausgegeben werden. Das nachfolgende Beispiel wandelt mithilfe eines selbst erstellten Arrays die englischen Bezeichnungen in die entsprechenden deutschen Namen um.

Beispiel: *monat_dt.php*

Über ein Array sollen beim Auslesen des aktuellen Datums die deutschsprachigen Bezeichnungen der Monate ausgegeben werden.

```
<?php
① $monat = array(1 => "Januar", "Februar", "März", "April",
                 "Mai", "Juni", "Juli", "August", "September",
                 "Oktober", "November", "Dezember");
② $datum = getdate();
echo "<p><strong>Englischsprachige Bezeichnung des Monats</strong>";
echo "<br>Heute ist der " . $datum["mday"] . ". ";
③ echo $datum["month"] . " " . $datum["year"] . ".</p>";
echo "<p><strong>Deutschsprachige Bezeichnung des Monats</strong>";
echo "<br>Heute ist der " . $datum["mday"] . ". ";
④ echo $monat[$datum["mon"]]. " " . $datum["year"] . ".</p>";
?>
```

- ① Die Namen der Monate werden im Array \$monat definiert. Für Januar, den ersten Eintrag, wird der Index 1 vergeben. Diese Zuweisung wird vorgenommen, damit die Monatnamen die dazugehörigen Monatszahlen als Index erhalten. Da bei indizierten Arrays für das nächste Element immer der nächst höhere Index verwendet wird, reicht es hier, lediglich den ersten Array-Eintrag mit dem Index 1 zu versehen. Die Indexe 2–12 werden dann vom PHP-Interpreter automatisch und den Monaten entsprechend vergeben.



Englische und deutsche Monatsbezeichnungen
(Beispieldatei „monat_dt.php“)

- ② In der Array-Variablen \$datum werden die Datumswerte gespeichert.
- ③ Über den Array-Schlüssel month wird der englische Name des Monats ausgegeben.
- ④ Damit ein bestimmtes Element des Arrays \$monat über den Index angesprochen werden kann, benötigen Sie den Monat als Zahl. Die Zahl liefert Ihnen \$datum["mon"]. Es folgt die erneute Ausgabe des aktuellen Datums. Diesmal wird der Wert aus dem von Ihnen angelegten Array \$monat mit dem Wert von \$datum["mon"] angesprochen und angezeigt.

10.4 Länder- und Spracheinstellungen ändern

Einstellungen für deutsche Sprache festlegen

PHP unterstützt bei Länder- und sprachspezifischen Informationen eine Reihe verschiedener Sprachen. Dies bedeutet, Sie können beispielsweise die deutsche Bezeichnung von Monatsnamen direkt ausgeben lassen. Dazu müssen Sie über die Funktion `setlocale()` die entsprechende Sprache einstellen.

Syntax und Bedeutung der `setlocale()`-Anweisung

- ✓ Der erste Parameter Kategorie legt fest, auf welche Angaben sich die Umstellung der Sprache auswirken soll. Folgende Angaben sind möglich:

```
setlocale(Kategorie, Sprache);
```

LC_ALL	alle Einstellungen
LC_COLLATE	für den Vergleich von Zeichenketten
LC_CTYPE	betrifft Klassifizierungen und Umwandlung von Zeichen
LC_MONETARY	für Währungsfunktionen
LC_NUMERIC	für das Dezimal-Trennzeichen bei Zahlen
LC_TIME	betrifft Zeit- und Datumsformatierungen
LC_MESSAGES	Systemmeldungen (abhängig von der PHP-Installation)
✓	Der zweite Parameter legt die anzuwendende Sprache und Region fest und wird in Hochkommata angegeben. Folgende Werte sind zulässig (Auszug aus einer Liste von mehr als 50 möglichen Werten): Nachfolgende Kurzbezeichner wurden bisher in PHP integriert:
de_DE	Deutschland
de_AT	Österreich
de_CH	Schweiz
fr_FR	Frankreich
en_GB	Großbritannien
en_US	USA

Die Funktion `setlocale()` ist systemabhängig. Sollten die Werte für die anzuwendende Sprache nicht greifen, verwenden Sie alternativ die dreibuchstabigen Sprachcodes nach ISO 639, z. B. `deu`, `fra` oder `eng`. Alternativ kann auch die Schreibweise '`deu_deu`' zum Erfolg führen.

Nachfolgend wird die deutsche Sprache für alle Werte eingestellt.

```
setlocale(LC_ALL, "de_DE");
```

Datum und Zeit sprachspezifisch ausgeben

Um aktuelle Zeit- und Datumswerte anhand der eingestellten lokalen Informationen nutzen zu können, steht Ihnen die Funktion `strftime()` zur Verfügung.

Ähnlich der Funktion `date()` können Sie über `strftime()` ein Datum für eine Ausgabe formatieren. Hierbei ist jedoch zu beachten, dass sich die Formatanweisungen der beiden Funktionen grundlegend unterscheiden.

Syntax der `strftime`-Anweisung

- ✓ Die Anweisung erwartet als ersten Parameter die Formatanweisungen.
- ✓ Ohne die Angabe des zweiten Parameters `Zeitstempel` liefert die Funktion die aktuelle Datums- und Zeitinformation zurück.

`strftime(Format [, Zeitstempel]);`

Ausgewählte Formatanweisungen

Mit diesen Formatanweisungen bestimmen Sie, welche Elemente des Datums genutzt werden sollen. Alle Formatanweisungen sind in Anführungszeichen „“ anzugeben.

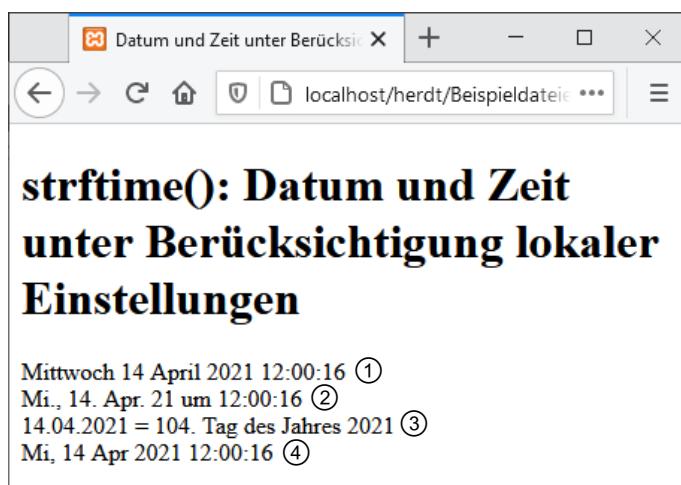
Format-Zeichen	Resultat	Beispiel
%a	Abgekürzter Tag der Woche	Mon
%A	Bezeichnung für den Wochentag	Monday
%b	Abgekürzter Monatsname	Mar
%B	Name des Monats	March
%d	Tag des Monats mit führender Null	01 bis 31
%H	Stunde im 24-Stunden-Format mit führender Null	00 bis 23
%j	Tag des Jahres	1 bis 366
%m	Zahl des Monats mit führender Null	01 bis 12
%M	Minuten mit führender Null	00 bis 59
%S	Sekunden mit führender Null	00 bis 59
%x	Datumswiedergabe ohne Zeit	03/15/21
%X	Zeitwiedergabe ohne Datum	10:48:21
%Y (großes Y)	Jahr als vierstellige Zahl	2021
%y (kleines y)	Jahr als zweistellige Zahl	21
%Z	Sommerzeit	Mittteleuropäische Zeit
%D	Wie %m/%d/%y	03/15/21
%T	Wie %H:%M:%S	10:48:21

Eine vollständige Übersicht aller Format-Zeichen finden Sie unter <https://www.php.net/manual/de/function.strftime.php>.

Beispiel: *strftime.php*

Analog zum Beispiel der `date()`-Funktion werden die Datums- und Zeitangaben mit der `strftime()`-Funktion gezeigt. Sie kann im Gegensatz zur Funktion `date()` die lokalen Einstellungen über `setlocale()` berücksichtigen.

```
<?php
    setlocale(LC_ALL, "deu");
    echo "<p>";
①    echo strftime("%A %d %B %Y %H:%M:%S", time()) . "<br>";
②    echo strftime("%a., %d. %b. %y um %X", time()) . "<br>";
③    echo strftime("%x = %j. Tag des Jahres %Y", time()) . "<br>";
④    echo strftime("%a, %d %b %Y %X", time()). "</p>";
?>
```



Anzeige der Beispieldatei „*strftime.php*“

10.5 Zeitfunktionen

Aktuelle Zeit mit der Funktion `time()` bestimmen

Die Zeit, die seit dem 01.01.1970 um 00:00:00 Uhr (Greenwich-Zeit; GMT Greenwich Mean Time) vergangen ist, wird auch **UNIX-Timestamp** (oder einfach **Zeitstempel**) genannt und wird in Sekunden berechnet. Dies ist für die weiteren Zeit- und Datumsfunktionen von Bedeutung, da alle weiteren Datums- und Zeitberechnungen auf dieser Angabe beruhen.

Syntax der Funktion `time()`

- ✓ Diese Funktion ist ohne die Angabe eines Parameters zu verwenden. `time();`
- ✓ Da diese Funktion den aktuellen Zeitstempel zurückgibt, können Sie auch durch einfache Addition bzw. Subtraktion von Sekunden andere Zeitstempel als den aktuellen generieren. So erhalten Sie z. B. durch die Berechnung: `time() - 7 * 86400` (1 Tag = 86400 Sekunden) den Zeitstempel für den Zeitpunkt vor genau einer Woche.

UNIX-Timestamp eines Datums mit der Funktion `mkttime()` bestimmen

Um den Zeitstempel, also die Sekunden vom 01.01.1970 bis zum angegebenen Datum, zu erhalten, verwenden Sie die `mkttime()`-Funktion.

Syntax und Bedeutung der Funktion `mkttime()`

```
mkttime([Stunde [, Minute] [, Sekunde] [, Monat] [, Tag] [, Jahr]]);
```

- ✓ Die Funktion erwartet optional die sechs angegebenen Parameter.
- ✓ Einzelne Parameter können Sie von rechts nach links weglassen, also zuerst `Jahr`, dann `Tag`, `Monat` usw. PHP verwendet bei fehlenden Werten den aktuellen Datums- oder Zeitwert des Webservers. Dies bedeutet, dass beim Aufruf von `mkttime()` ohne Parameter dasselbe Ergebnis zurückgeliefert wird wie bei `time()`.
- ✓ Möchten Sie einen Wert nicht angeben, ersetzen Sie ihn durch die Zahl 0. Dies kann sinnvoll sein, wenn Sie nur mit dem Datum arbeiten wollen. In dem Fall setzen Sie die Parameter `Stunde`, `Minute` und `Sekunde` auf den Wert 0.

Beispiel: `date_diff.php`

Anhand eines vorgegebenen Datums wird die Differenz zum aktuellen Datum berechnet.

```
<?php
① $tag = 15;
$monat = 1;
$jahr = 1969;
② $start = mkttime(0, 0, 0, $monat, $tag, $jahr);
③ $diff = time() - $start;
④ echo "<p><strong>" . (floor($diff / 86400)) . " Tage</strong> liegen zwischen ";
⑤ echo "heute (" . date("d.m.Y") . ") und dem " .
      date("d.m.Y", $start) . ".</p>";
?>
```

- ① Die Variablen werden mit den Angaben eines Datums gefüllt, hier: 15.1.1969.
- ② Aus diesen Angaben werden über die Funktion `mkttime()` die bis dato vergangenen Sekunden berechnet und in der Variablen `$start` gespeichert. Da Stunden-, Minuten- und Sekundenwerte für dieses Beispiel keine Rolle spielen, werden sie jeweils mit dem Wert 0 angegeben.



Ausgabe der Differenz zweier Daten
(Beispieldatei „date_diff.php“)

- ③ Aus dem aktuellen Zeitstempel `time()` und dem Zeitstempel der Variablen `$start` wird die Differenz zum jetzigen Zeitpunkt berechnet. Der Wert liegt in Sekunden vor.
- ④ Um aus den Sekunden einen Wert in Tagen zu berechnen, wird dieser durch 86400 dividiert. Der Wert 86400 entspricht der Anzahl der Sekunden pro Tag ($24 * 60 * 60$). Die zusätzlich verwendete Funktion `floor()` runden eine Fließkommazahl auf die nächst kleinere Ganzzahl ab, z. B. `floor(139.35)` ergibt 139.
- ⑤ Das aktuelle sowie das im Beispiel definierte Datum wird über die Funktion `date()` formatiert ausgegeben. Da diese Funktion als optionalen Parameter den Zeitstempel in Sekunden erwartet, wird der bereits gespeicherte Zeitstempel über die Variable `$start` übergeben.

Genaue Zeitstempel und Zeitspannen mit der Funktion `microtime()` berechnen

Je nach Anwendungsfall kann es nicht ausreichen, Zeitstempel lediglich in Sekunden zur Verfügung zu haben. Für geringere Zeiteinheiten steht in PHP die Funktion `microtime()` zur Verfügung. `microtime()` liefert den aktuellen Zeitstempel in Mikrosekunden zurück. Wegen ihrer größeren Genauigkeit wird die Funktion häufig zur Berechnung von Zeitspannen verwendet, z. B. zur Berechnung der Ausführungszeit von komplexeren Programmen.

Syntax der Funktion `microtime()`

- ✓ Diese Funktion kann ohne die Angabe eines Parameters verwendet werden und liefert dann den aktuellen Zeitstempel in der Form *Mikrosekunden Sekunden*.
- ✓ Wird der Parameter `TRUE` angegeben, erfolgt die Ausgabe des Zeitstempels in Mikrosekunden als Gleitkommazahl. Ansonsten wird der Standardwert `FALSE` verwendet.

`microtime([TRUE/FALSE]);`

Beispiel: `microtime.php`

Die Funktion wird mit und ohne Parameter aufgerufen. Zusätzlich wird die Ausführungszeit einer Befehlsfolge berechnet.

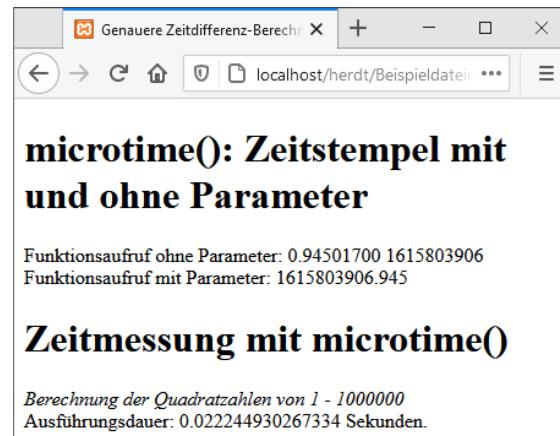
```
<h1>microtime(): Zeitstempel mit und ohne Parameter</h1>
<?php
① echo "Funktionsaufruf ohne Parameter: " . microtime() .
    "<br>";
echo "Funktionsaufruf mit Parameter: " . microtime(TRUE);
?>
<h1>Zeitmessung mit microtime()</h1>
<?php
    echo "<p><em>Berechnung der Quadratzahlen von
        1 - 1000000</em><br>";
    $start = microtime(TRUE);
    ③ for ($i = 1; $i <= 1000000; $i++) {
        ④ echo "$i: " . sqrt($i) . "<br>";
    }
}
```

```

⑤ $ende = microtime(TRUE);
⑥ echo "Ausführungsduer: " . ($ende - $start) . "
      Sekunden.</p>" ;
?>

```

- ① Die Funktion `microtime()` wird mit und ohne Parameter aufgerufen und am Bildschirm ausgegeben.
- ② Der Variablen `$start` wird der aktuelle Zeitstempel vor Ausführen der Schleife ③ in Mikrosekunden zugewiesen.
- ③ In einer Schleife erfolgt die Berechnung und Ausgabe der Quadratwurzeln der Zahlen von 1 bis 1000000.
- ④ Der Variablen `$ende` wird der aktuelle Zeitstempel nach Ausführung der Schleife ③ in Mikrosekunden zugewiesen.
- ⑤ Die Variable `$start` wird von der Variablen `$ende` subtrahiert, um die Ausführungsduer der Schleife zu berechnen.



Ausgabe der Beispieldatei „microtime.php“ (ohne die 1000000 echo-Anweisungen ④)

Zeitstempel einer englischen Datumsangabe

Die Funktion `strtotime()` wandelt ein beliebiges Datum, das in englischer Schreibweise angegeben werden muss, in einen Zeitstempel um.

Syntax und Bedeutung der Funktion `strtotime()`

- ✓ Im Parameter Datumsangabe wird eine englische Datumsangabe als Zeichenkette übergeben.

`strtotime(Datumsangabe [, Zeitstempel]);`

Der Zeitstring, der `strtotime()` übergeben wird, kann sowohl eine Zeichenkette im englischen Dateiformat sein, es können aber auch sprechende Rechenoperationen angegeben werden. Die Varianten des Zeitstring sind vielfältig. Die nachfolgende Tabelle zeigt einige Beispiele, weitere Beispiele finden Sie unter <https://www.php.net/manual/de/function.strptime.php>.

<code>strtotime("now");</code>	Gibt die aktuelle Zeit des Servers aus.
<code>strtotime("24 March 2021");</code>	Gibt den Zeitstempel für den 24. März 2021 um 00:00:00 Uhr aus.
<code>strtotime("+1 day");</code>	Fügt dem aktuellen Datum einen Tag hinzu.
<code>strtotime("-1 week");</code>	Zieht von der aktuellen Zeit genau eine Woche ab.
<code>strtotime("+1 week 2 days 4 hours 2 minutes");</code>	Rechnet zur aktuellen Zeit eine Woche, zwei Tage, vier Stunden und zwei Minuten hinzu.

Funktion	Zeitstempel	Datumsausgabe des Zeitstempels
strtotime("now")	1615804425	Mon, 15 Mar 2021 11:33:45 +0100
strtotime("24 March 2021")	1616540400	Wed, 24 Mar 2021 00:00:00 +0100
strtotime("+1 day")	1615890825	Tue, 16 Mar 2021 11:33:45 +0100
strtotime("-1 week")	1615199625	Mon, 08 Mar 2021 11:33:45 +0100
strtotime("+1 week 2 days 4 hours 2 minutes")	1616596545	Wed, 24 Mar 2021 15:35:45 +0100

Anzeige der verschiedenen Parameter (Beispieldatei „strtotime.php“)

10.6 Datumsangaben überprüfen

Datums- bzw. Zeitangabe auf Gültigkeit überprüfen

Um eine Datumsangabe auf Gültigkeit zu überprüfen, benutzen Sie die Funktion `checkdate()`.

Syntax und Bedeutung der `checkdate()`-Anweisung

- ✓ Die Anweisung erwartet als Parameter den Monat, den Tag sowie das Jahr. Achten Sie bei der Übergabe der Daten auf die richtige Reihenfolge.
- ✓ Monat kann einen Wert zwischen 1 und 12 besitzen.
- ✓ Der Tag ist jeweils abhängig vom Monat.
- ✓ Das Jahr kann einen Wert zwischen 1 und 32767 besitzen.
- ✓ Als Rückgabewert liefert die Funktion den Wert TRUE, falls das Datum existiert, sonst FALSE.

`checkdate(Monat, Tag, Jahr)`

Beispiel: `checkdate.php`

Im Beispiel wird ein Formular erstellt, das die Eingabe eines Datums in ein HTML-Formular vorsieht. In derselben Datei wird mit PHP überprüft, ob das eingegebene Datum gültig ist.

```

① <form action="<?php echo $_SERVER["PHP_SELF"]; ?>">
    method="post">
    Geben Sie ein beliebiges Datum im Format TT.MM.JJJJ ein:
    <input type="text" name="datum" size="10" maxlength="10">
    <input type="submit" name="absenden" value="Prüfen">
</form>
<?php

```

```

② if (isset($_POST["absenden"])) {
    $data = explode(".", $_POST["datum"]);
    ④ if ((!checkdate($data[1], $data[0], $data[2]))
        or ( count($data) != 3)) {
        echo "<p>" . $_POST["datum"] . " ist kein korrektes
        Datum!</p>";
    } else {
        echo "<p>Das Datum " . $_POST["datum"] . " ist
        korrekt!</p>";
    }
}
?>

```

- ① Die Angabe von `$_SERVER["PHP_SELF"]` im `action`-Parameter des HTML-Formular-Tags `form` legt fest, dass die Formulardaten an die Datei `checkdate.php` selbst zur Auswertung gesendet werden.
- ② Es erfolgt die Prüfung auf Existenz der Variablen `$_POST["absenden"]`. Der Array-Schlüssel `absenden` ist in der globalen `$_POST`-Variable nur dann vorhanden, wenn die Schaltfläche *Prüfen* gedrückt und damit die Variable `absenden` aus dem Formular übermittelt wurde. Ergibt die Prüfung `TRUE`, wurde das Formular versendet und der Anweisungsblock wird ausgeführt. Andernfalls ist das Formular noch nicht versendet worden, der Anweisungsblock wird übersprungen.
- ③ Das vom Nutzer eingegebene Datum wird über die Zeichenkettenfunktion `explode()` an den Dezimalstellen getrennt, die einzelnen Segmente werden von PHP als indiziertes Array `$data` gespeichert.
- ④ Zum Testen des Datums werden über die einzelnen Einträge des Arrays `$data` übergeben. Wird die Eingabe korrekt nach dem Muster `TT.MM.JJJJ` vorgenommen, ist nach dem Einsatz von `explode()` der Monat das zweite Array-Element (mit dem Index 1), der Tag ist der erste Eintrag im Array (Index 0), das Jahr der dritte Eintrag (Index 2). Je nach Rückgabe der `checkdate()`-Funktion und der Elementanzahl des Arrays `$data` wird eine entsprechende Mitteilung im Browser ausgegeben. Werden der Funktion nichtnumerische Werte übergeben, generiert diese einen *Fatal error*.



Ausgabe der Beispieldatei „checkdate.php“ nach Eingabe des 31.04.2021 und Klick auf den Prüfen-Button

10.7 Übungen

Übung 1: Mit Datums- und Zeitangaben arbeiten

Level		Zeit	ca. 15 min
Übungsinhalte	<ul style="list-style-type: none"> ✓ Verwenden der <code>date()</code>-Funktion ✓ Formatierungen für <code>date()</code> umsetzen ✓ Lokale Datumseinstellungen ✓ Datumsformatierung per <code>strftime()</code> ✓ Datumsüberprüfung mit <code>getdate()</code> 		
Übungsdatei	--		
Ergebnisdatei	<code>date_time.php</code>		

1. Geben Sie mithilfe der Funktion `date()` folgende Datums- und Zeitangaben aus.
Die Angaben werden automatisch durch das aktuelle Datum ersetzt.

03.04.21

03-04-2021

03.04.2021 - 13:51:02

04/03/21 - 01:51 PM

2021-04-03

13:04 Uhr

2. Die Funktion `date()` liefert die Tagesnamen standardmäßig in englischer Sprache zurück.
Lassen Sie sich den aktuellen Wochentag mittels `setlocale()` und `strftime()` in deutscher Sprache ausgeben.
3. Lesen Sie für den aktuellen Zeitstempel den Wochentag aus. Verwenden Sie die Fallauswahl `switch`, um für alle möglichen Wochentage eine beliebige Bildschirmausgabe festzulegen, z. B. *Heute ist Dienstag*.

Übung 2: Zeitmessung durchführen

Level		Zeit	ca. 10 min
Übungsinhalte	<ul style="list-style-type: none"> ✓ Verwendung der Funktion <code>microtime()</code> ✓ Vergleich unterschiedlicher Programmieransätze ✓ Einsatz von PHP-Funktionen 		
Übungsdatei	--		
Ergebnisdatei	<code>vergleich.php</code>		

Ein Bekannter behauptet, dass mit PHP Quadratzahlen am schnellsten berechnet werden, wenn man eine Funktion zur Berechnung von Quadratzahlen erstellt und diese aufruft. Da Sie sich nicht sicher sind, ob eine direkte Berechnung oder der Aufruf einer Funktion schneller ist, wollen Sie beide Varianten vergleichen.

1. Erstellen Sie eine Datei (`vergleich.php`) und berechnen Sie mit einer `for`-Schleife die Quadratzahlen für die Zahlen 1 bis 1000000 – einmal über Aufruf einer selbst definierten Funktion und einmal durch direkte Berechnung. Verwenden Sie die Funktion `microtime()`, um die Ausführungszeit beider Berechnungsarten zu vergleichen.
2. Geben Sie das Ergebnis des Vergleichs auf dem Bildschirm aus.

Ausführungszeit: Funktionsaufruf - direkter Aufruf

Ausführungszeit (10000 x Aufruf Funktion): 0.043164968490601 Sekunden.

Ausführungszeit (10000 x direkte Ausführung): 0.011668920516968 Sekunden.

Anzeige der Beispiellösung (Datei „vergleich.php“)

11

Sessions



Beispieldateien: Dateien aus Ordner *Kapitel_11*

11.1 Mit Sessions arbeiten

Grundlagen zu Sessions

Eine Datenübertragung im Internet erfolgt über das Protokoll HTTP. HTTP ist allerdings ein **zustandsloses Protokoll**, das heißt, mehrere Anfragen, also Seitenaufrufe – auch desselben Seitenbesuchers – sind grundsätzlich voneinander unabhängige Aktionen. Daten, die beispielsweise ein Benutzer in ein Formular eingegeben hat, werden in Variablen nur bis zum auswertenden PHP-Programm weitergegeben. Bereits auf der übernächsten aufgerufenen Webseite sind die Daten nicht mehr vorhanden. Der Webserver kann also, wenn keine weiteren Maßnahmen ergriffen wurden, **nicht** feststellen, welche Seitenaufrufe vom selben Benutzer kommen.

Für diverse Abwicklungen auf Webseiten benötigen Sie jedoch genau eine Funktionalität, über die Sie einen **Besucher wiedererkennen** und mit ihm verbundene Daten zwischenspeichern können. Sie wollen z. B. über mehrere Webseiten Eingaben ermöglichen und dann als Bestellung absenden (Shopsystem). Oder Sie möchten, dass sich auf Ihren Webseiten Benutzer einloggen können und möchten auf jeder weiteren Seite prüfen, ob der Benutzer durch seine Anmeldung autorisiert ist, die jeweilige Seite aufzurufen.

Gerade bei sogenannten personalisierten Seiten, die auf den Benutzer „reagieren“, ist es notwendig, bestimmte Daten, z. B. den Login-Status, von Seite zu Seite mitzuführen bzw. weiterzugeben.

PHP unterstützt diese Anforderungen mithilfe von Sessions (Sitzungen). Eine Session bietet die Möglichkeit, Daten während eines Webseitenbesuchs über mehrere Seiten zwischenspeichern und für jede einzelne Seite verfügbar zu halten. Der Nutzen einer Session liegt darin, dass während einer Session Daten gespeichert und diese von jeder weiteren PHP-Seite weiterverwendet werden können. Eine Session endet,

- ✓ wenn der Benutzer den Browser schließt,
- ✓ wenn eine Webseite mit entsprechenden Befehlen aufgerufen wird,
- ✓ nach einer definierten Zeitspanne
- ✓ oder – je nach Programmierung – auch beim Verlassen der Webseiten, auf denen die Session definiert wurde.



Die Standarddauer einer Session beträgt 1440 Sekunden (**24 Minuten**). Mit jedem neuen Seitenaufruf während der Sitzung wird diese erneut gesetzt. Nach 1440 Sekunden Inaktivität wird die Sitzung des Seitenbesuches beendet. Zum Beenden einer Session reicht das Schließen eines einzelnen Tabs des Browsers nicht aus. Die Session wird erst beendet, wenn der **Browser komplett geschlossen** wird.

Session-IDs: Identifikation einer Session

Damit Informationen zu einem Besucher innerhalb einer Website seitenübergreifend verwaltet werden können, erhält ein Besucher beim ersten Zugriff auf mit Sessions definierte Webseiten eine zufällige, eindeutige 32-stellige ID, die **Session-ID**. Diese wird bei jedem Aufruf einer Seite an den Webserver gesendet, woran dieser den Besucher wiedererkennt. Anhand der Session-ID können gespeicherte Daten zugeordnet werden.

Solange der Besucher auf den Webseiten verweilt, auf denen die Session gestartet wurde, werden auf dem Webserver benutzerbezogene Daten in einer **Session-Datei** gespeichert. Dieses können z. B. Formulareingaben des Nutzers sein, aber auch Daten, die anhand von Login-Daten aus Datenbanken oder anderen Quellen ermittelt wurden. Bei der Session-Datei handelt es sich um eine Textdatei, die auf dem Webserver in einem Ordner für temporäre Dateien abgelegt wird. Bei der Standardeinstellung der im Buch verwendeten XAMPP-Installation handelt es sich unter Windows um das Verzeichnis `C:\xampp\tmp`. Unter macOS befindet sich das entsprechende Verzeichnis unter `/xampp/temp` (*in der gemounteten virtuellen Maschine*).

Es handelt sich hierbei um ein serverseitiges Session-Cookie. Cookie ist die Bezeichnung für die Textdatei, die Informationen über einen Vorgang sammelt. In jedem Skript, das zur Session gehört, wird die Session-ID zur abermaligen Authentifizierung benötigt. Die Übertragung der Session-ID von Skript zu Skript wird als **Durchschleifen** bezeichnet. Das Durchschleifen der Session-ID ist einfach zu realisieren, da diese Funktionalität intern von PHP verwaltet wird. Dazu werden die Daten der Session temporär auf dem Webserver gespeichert. Das Durchschleifen einer Session-ID findet nur innerhalb der PHP-Skripte statt, die eine Session mit der Funktion `session_start()` fortsetzen. Durch Aufruf einer Seite mit der Dateinamenerweiterung `*.htm(l)` oder einer `*.php`-Seite, auf der `session_start()` nicht aufgerufen wurde, verlässt der Nutzer also die Session.

Als Gegenstück zum Session-Cookie speichert der Browser einen Browser-Cookie mit der Session-ID ab. Die Session-ID wird mit jedem Seitenaufruf an den Server gesendet. Damit wird die konkrete Verknüpfung zwischen Client und Server hergestellt.

Falls Nutzer Cookies im Browser deaktiviert haben, funktionieren Sessions nicht korrekt. Theoretisch wäre es möglich, die Session-ID händisch von Seite zu Seite „durchzuschleifen“, indem man sie an alle Links anhängt und mit allen Formularen überträgt. Da aber im Grunde jede Webseite auf Session-Cookies basiert, ist das tatsächlich nur eine theoretische Überlegung.



Mit der Einführung der DSGVO in 2018 bekommen immer mehr Webseiten sogenannte Cookie-Banner, auf denen die Besucher über die Verwendung von Cookies informiert werden. Die DSGVO zielt vor allem auf Cookies ab, die Drittanbieter setzen, um Nutzerdaten, z. B. für Marketingzwecke, abzugreifen. Die durch Sessions gesetzten Cookies, wie in diesem Kapitel beschrieben, gehören zu den notwendigen Cookies einer Webseite. Sie gehören zu einer funktionierenden Webseite wie das Wasser zum Waschen.

11.2 Session starten bzw. fortsetzen

Wenn Sie mit einer Session arbeiten wollen, gehen Sie wie folgt vor:

- ✓ Alle Dateien, die innerhalb der Session erreichbar sein sollen, müssen korrekte PHP-Dateien sein. Die Dateiendung muss `*.php` lauten. Früher wurden Endungen wie `*.php4` oder `*.php5` verwendet, derweil hat sich `*.php` als Standard etabliert, von dem man auch nicht abweichen sollte. Welche Dateiendung als PHP-Datei erkannt wird, ist in der `php.ini` konfiguriert.
- ✓ Jede dieser Dateien beginnt mit Öffnen eines PHP-Blocks (`<?php`) in der ersten Zeile der Datei, gefolgt von der Funktion `session_start()`, um eine Session zu starten bzw. eine laufende Session fortzusetzen. Das gilt auch für Dateien, die ausschließlich HTML-Tags enthalten.

Um eine Session zu starten, verwenden Sie die Funktion `session_start()`. Beim Aufruf des Programms wird eine Session gestartet und eine eindeutige Session-ID erzeugt. Auf die Session-ID können Sie bei Bedarf mit der Funktion `session_id()` zugreifen.

Syntax und Bedeutung der Funktion `session_start()`

- ✓ Bei Aufruf der Funktion `session_start()` wird eine neue Session gestartet oder eine aktuelle Session wieder aufgenommen. `session_start();`
Ist die vom User übermittelte Session-ID aktiv, wird diese ID von PHP automatisch erkannt und damit die bestehende Session fortgeführt. Wird keine Session-ID erkannt, generiert PHP eine neue Session-ID und startet damit eine neue Session.
- ✓ Jede Webseite, die zu einer Session gehören soll, muss den Befehl `session_start()` beinhalten, ansonsten kann keine Session gestartet bzw. fortgesetzt werden.
- ✓ Die Funktion erwartet keine Parameter.
- ✓ Rückgabewert der Funktion ist `TRUE` (bei Erfolg) bzw. `FALSE` (im Fehlerfall).

! Sie müssen `session_start()` aufrufen, bevor Sie eine Ausgabe im Browser vornehmen. Eine `echo`-Anweisung in PHP oder ein HTML-Tag vor dem Aufruf dieser Funktion führen zu einer Fehlermeldung ("*header already sent*"). Denselben Effekt haben Leerzeichen außerhalb von PHP-Blöcken. Auch ein versehentlich eingefügtes Leerzeichen hinter dem schließenden PHP-Tag einer inkludierten PHP-Datei, die vor dem Aufruf von `session_start()` eingebunden wird, kann zu dieser Fehlermeldung führen. Um Schwierigkeiten zu vermeiden, sollten Sie als Erstes ganz oben in der Datei einen PHP-Block einfügen und danach `session_start()` aufrufen.

Standardmäßig lautet der Name der Session `PHPSESSID`, der in der Datei `php.ini` definiert ist. Sie können eine Session zur Laufzeit auch mit einem anderen Namen versehen. Über diesen Namen können Sie diese Session im späteren Verlauf wieder ansprechen.

Syntax und Bedeutung der Funktion `session_name()`

Möchten Sie bei Aufruf einer Webseite einen eigenen Session-Namen definieren, rufen Sie die Funktion `session_name()` **vor** dem Starten der Session mit `session_start()` auf.

```
session_name(Bezeichnung);
```

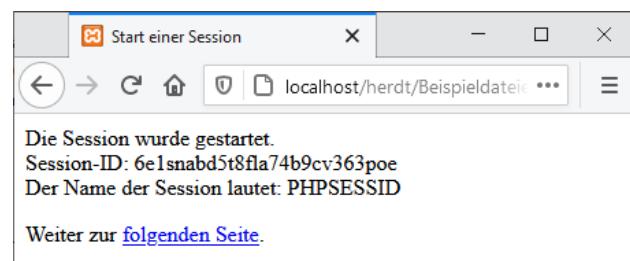
Damit wird der Wert der Session mit dem angegebenen Namen erstellt bzw. übernommen.

! Achtung: Der Parameter Bezeichnung in der Funktion `session_name()` darf nicht nur aus Zahlen bestehen, sondern muss mindestens einen Buchstaben enthalten.

Beispiel `start.php`

```
<?php
① session_start();
② $id = session_id();
echo "<!DOCTYPE html>";
echo "<html><head>";
echo "<meta charset=\"UTF-8\">";
echo "<title>Start einer Session</title></head>";
echo "<body>Die Session wurde gestartet.<br>";
echo "Session-ID: " . $id;
③ echo "<br>Der Name der Session lautet: " . session_name();
echo "<p>Weiter zur <a href='formular.php'>folgenden
Seite</a>.</p>";
④ echo "</body></html>";
```

- ① Mit dem Befehl `session_start()` wird eine neue Session gestartet oder eine gestartete Session fortgesetzt.
- ② Der Variablen `$id` wird die aktuelle Session-ID zugewiesen.
- ③ Über die Funktion `session_name()` wird der Name der aktuellen Session ausgegeben.
- ④ Da hier bereits die komplette HTML-Ausgabe geschehen ist und gerade im Zusammenhang mit Sessions auch die Ausgabe von Leerzeichen Probleme bereiten können, wird der schließende PHP-Tag `?>` weggelassen. Damit vermeiden Sie eine versehentliche Ausgabe von Leerzeichen am Dateiende. Die PHP-Datei und die Abarbeitung des Skripts enden damit.



Anzeige der Beispieldatei „start.php“

11.3 Daten in einer Session speichern

PHP hilft Ihnen bei Start und Betrieb einer Session. Die automatisch angelegte Session-Datei kann gewünschte Informationen aufnehmen und für die Dauer der Session verfügbar halten. Das Speichern von Daten in einer Session hingegen geschieht nicht automatisch, sondern hängt von der Programmierung ab. Um Daten in der Session-Datei zu speichern, verwenden Sie die superglobale Array-Variable `$_SESSION`.

Mit dieser Zuweisung erreichen Sie, dass das Element unter dem Array-Schlüssel "benutzer" in die Session-Datei eingetragen wird.

```
$_SESSION["Element"] = Wert;  
z.B. $_SESSION["benutzer"] = "Maxi";
```

Damit steht es in der Session als Element der Array-Variablen `$_SESSION` zur Verfügung. Am Ende eines PHP-Skripts werden üblicherweise alle Variablen aus dem Arbeitsspeicher des Servers gelöscht. So auch die Werte aus `$_SESSION`. Setzt das als nächstes aufgerufene Skript die Session fort, werden die Daten aus der Session-Datei in die Variable `$_SESSION` eingelesen. Die Inhalte stehen somit wieder zur Verfügung, nachdem Sie `session_start()` aufgerufen haben. Dieser Vorgang wiederholt sich von Skript zu Skript bis zum Ende der Session.

Beispiel: *formular.php*

In Beispiel wird ein Formular aufgebaut, in das Nutzer ihre Daten eingeben können und das nach Absenden des Submit-Buttons weiterverarbeitet wird. Im auswertenden Skript werden diese Daten an Variablen übergeben, auf die Sie innerhalb einer Session von jeder anderen Seite zugreifen können.

```
<?php  
①   session_start();  
?><!DOCTYPE html>  
<html>  
  <head><meta charset="UTF-8">  
    <title>Formular zur Eingabe der Daten</title></head>  
  <body>  
    <h1>Session: Angaben zur Person</h1>  
    <p>Bitte füllen Sie die nachfolgenden Eingabefelder aus:</p>  
    ②    <form action="auswertung.php" method="POST">  
      <p>Vorname: <input type="text" name="vorname"></p>  
      <p>Nachname: <input type="text" name="nachname"></p>  
      <p>Wohnort: <input type="text" name="ort"></p>  
      <p><input type="submit" value="Abschicken"></p>  
    </form>  
  </body>  
</html>
```

- ① Eine Session wird initialisiert. In diesem Beispiel läuft bereits eine Session (falls Sie die Seite von der Seite `start.php` aufgerufen haben), also wird die bestehende Session fortgesetzt. Die Session muss lückenlos fortgesetzt werden, deshalb erhält auch die Datei, die nur ein HTML-Formular beinhaltet, die Dateinamenerweiterung `*.php` und initialisiert eine Session. Da der PHP-Block zur Initialisierung der Session notwendig ist, muss es eine PHP-Datei sein. In HTML-Dateien können Sie diesen PHP-Block nicht einsetzen.
- ② Ab hier wird ein einfaches Formular aufgebaut. Als Ziel-Skript des Formulars wird über das `action`-Attribut die PHP-Datei `auswertung.php` definiert, in der die Auswertung des Formulars vorgenommen werden soll.

Session: Angaben zur Person

Bitte füllen Sie die nachfolgenden Eingabefelder aus:

Vorname:

Nachname:

Wohnort:

Abschicken

Ausgabe der Beispieldatei „formular.php“

Beispiel: `auswertung.php`

```
<?php
① session_start();
echo "<!DOCTYPE html><html>";
echo "<head><meta charset=\"UTF-8\">";
echo "<title>Daten ins Session speichern</title></head>";
echo "<body><h1>Daten in der Session speichern</h1>";
echo "<p>Sie haben folgende Daten im Formular eingetragen:</p>
② echo "<br>Vorname: " . $_POST["vorname"];
echo "<br>Nachname: " . $_POST["nachname"];
echo "<br>Ort: " . $_POST["ort"] . "</p>";
③ $_SESSION["vorname"] = $_POST["vorname"];
$_SESSION["nachname"] = $_POST["nachname"];
$_SESSION["ort"] = $_POST["ort"];
$_SESSION["zeit"] = time();
echo "<p>Folgende Daten sind nun in der Session
gespeichert: </p>";
echo "<pre>";
④ print_r($_SESSION);
echo "</pre>";
echo "<p>Weiter zur <a href='auslesen.php'>folgenden
Seite</a>.</p></body></html>";
```

- ① Die laufende Session wird fortgesetzt.
- ② Die Daten aus dem übermittelten Formular werden über die Array-Variable `$_POST` ausgegeben.

- ③ In der Array-Variablen `$_SESSION` werden die Einträge mit den Array-Schlüsseln `vorname`, `name` und `ort` (Daten aus dem Formular) sowie zusätzlich die Variable `zeit`, die den aktuellen Zeitstempel enthält, gespeichert. Damit werden die Angaben in der aktuellen Session-Datei im Ordner `C:\xampp\tmp` (Windows) bzw. `/lampp/temp` (Mac) gespeichert und für den Zugriff über die Variable `$_SESSION` während der gesamten Session bereitgestellt.
- ④ Zur Überprüfung der in der Session gespeicherten Variablen wird mit der Funktion `print_r()` die Variable `$_SESSION` ausgelesen und angezeigt.

```

Daten in der Session speichern

Sie haben folgende Daten im Formular eingetragen:
Vorname: Maxi
Nachname: Müller
Ort: Posemuckel

Folgende Daten sind nun in der Session gespeichert:

Array
(
    [vorname] => Maxi
    [nachname] => Müller
    [ort] => Posemuckel
    [zeit] => 1616760629
)

```

Weiter zur [folgenden Seite](#).

Ausgabe der Beispieldatei „auswertung.php“

Session-Datei anzeigen lassen

Session-Variablen können vom Client nicht manipuliert werden. Nachdem sie erzeugt und gespeichert wurden, existieren sie nur im Datenspeicher des Servers und können somit nur vom Skript gelesen werden. PHP liest diese Textdatei am Anfang jeder Session ein. Bei jeder Wertzuweisung an die Session-Variable `$_SESSION` speichert PHP die geänderten oder neu hinzugekommenen Session-Daten wieder ab.

- ▶ Wechseln Sie unter Windows in den Ordner `C:\xampp\temp` bzw. `/lampp/temp` unter macOS.
- ▶ Öffnen Sie die Datei mit der aktuellen Session-ID mit Notepad++ oder einem einfachen Texteditor (unter macOS benötigen Sie root-Rechte, um die Datei zu öffnen). Der Dateiname der Session-Datei besteht aus dem Präfix `sess_`, gefolgt von der Session-ID (z. B. `sess_6e1snabd5t8fla74b9cv363poe`).

Die Datei hat folgenden Inhalt (in Form eines sogenannten serialisierten Arrays):

```
vorname|s:4:"Maxi";nachname|s:7:"Müller";ort|s:10:"Posemuckel";zeit|i:1616760629;
```

Die einzelnen Variablen werden nach folgendem Schema abgelegt:

```
Name | Datentyp [:Länge] :Variablen-Inhalt;
```

Anzeige der aktuellen Session-ID

Session-Datei mit gespeicherten Daten

11.4 Daten einer Session abrufen

Die gespeicherten Werte einer Session können Sie über den Namen der jeweiligen Variablen (dem Array-Schlüssel in `$_SESSION`) direkt ansprechen bzw. über die Funktion `foreach()` komplett auslesen. Da es sich bei der Variablen `$_SESSION` um eine Array-Variable handelt, können Sie alle Funktionen zur Weiterverarbeitung der Daten anwenden, die Sie aus dem Kapitel „*Arrays*“ kennen.

Beispiel: *auslesen.php*

Die Werte, die Sie im vorigen Beispiel in der Session-Datei gespeichert haben, möchten Sie auf einer neuen Seite auslesen. In dem Beispiel werden zwei Möglichkeiten aufgezeigt.

```
<?php
① session_start();
?><!DOCTYPE html>
<html>
    <head><meta charset="UTF-8"><title>Auslesen der
Session-Daten</title></head>
    <body>
        <h1>Auslesen der Session-Daten</h1>
        <?php
            echo "<p><em>Folgende Variablen wurden gespeichert
('foreach'):</em><br>";
        ② foreach ($_SESSION as $key => $value) {
            echo $key . " : " . $value . "<br>";
        }
        echo "<p><em>Direktes Ansprechen der Variablen:</em><br>";
        ③ echo "Vorname aus der Session-Datei: " .
            $_SESSION["vorname"] . "<br>";
        echo "Nachname aus der Session-Datei: " .
            $_SESSION["nachname"] . "<br>";
        echo "Ort aus der Session-Datei: " . $_SESSION["ort"] . "<br>";
        echo "Zeitstempel aus der Session-Datei: " .
            $_SESSION["zeit"] . "</p>";
        echo "<p>Weiter zur <a href='session_destroy.php'>folgenden
Seite</a>.</p>";
        ?
    </body>
</html>
```

- ① Die laufende Session wird fortgesetzt.
- ② Über die `foreach()`-Schleife wird jedes Element des Arrays `$_SESSION` ausgelesen. Mit den Standardvariablen `$key` und `$value` können Sie den Schlüssel und den Wert jeder Session-Variablen am Bildschirm anzeigen lassen.
- ③ Zusätzlich wird jede Variable einzeln anhand des Namens angesprochen und ausgegeben.

Auslesen der Session-Daten

Folgende Variablen wurden gespeichert ('foreach'):
 vorname: Maxi
 nachname: Müller
 ort: Posemuckel
 zeit: 1616760629

Direktes Ansprechen der Variablen:
 Vorname aus der Session-Datei: Maxi
 Nachname aus der Session-Datei: Müller
 Ort aus der Session-Datei: Posemuckel
 Zeitstempel aus der Session-Datei: 1616760629

Weiter zur [folgenden Seite](#).

Anzeige der Beispieldatei „*auslesen.php*“

11.5 Session-Daten und Session löschen

Neben dem Speichern und Auslesen der Daten können Sie diese auch komplett löschen und die Session beenden. Dies ist z. B. notwendig, wenn in einem Online-Shop die Bestellung ausgeführt wurde und somit der Warenkorb des Kunden gelöscht wird.

Löschen der Session-Daten

Wenn Sie alle Session-Variablen löschen wollen, also komplett die Inhalte der Array-Variablen `$_SESSION` sowie den Inhalt der Session-Datei, dann verwenden Sie die folgende Anweisung:

- ✓ Mit dieser Anweisung initialisieren Sie die Session-Variable `$_SESSION = array();` neu und leeren damit das vorhandene Array bzw. Sie ersetzen das bestehende Session-Array durch ein leeres Array. Die Session läuft nach dieser Anweisung weiter.

Wollen Sie einzelne Variablen der Session löschen und die restlichen Daten unberührt lassen, verwenden Sie die Funktion `unset()`:

- ✓ Sie geben als Parameter `Variable` die genaue Bezeichnung der Variablen an, die Sie löschen möchten. `unset(Variable);`
- Falls Sie aus den vorherigen Beispielen in diesem Kapitel den Vornamen aus der Session entfernen möchten: `unset($_SESSION["vorname"]);`

Löschen der Session mit der Funktion `session_destroy()`

Wenn Sie die komplette Session beenden wollen, reicht es nicht aus, die Session-Daten zu löschen. Die Session läuft weiter. Durch Ausführen der Funktion `session_destroy()` wird die Session-Datei gelöscht. Beim nächsten Versuch, Session-Daten zu speichern, oder beim Versuch, eine Session zu initialisieren, meldet PHP zurück, dass es die dazugehörige Session-Datei nicht (mehr) findet. In diesem Fall wird eine neue Session gestartet. Eine neue Session-ID wird erzeugt und die dazugehörige Session-Datei angelegt.

- ✓ Mit `session_destroy()` beenden Sie die aktuelle Session. Dabei wird die Session-Datei im Ordner `C:\xampp\tmp` bzw. `/xampp/temp` gelöscht. `session_destroy();`

Beispiel: `session_destroy.php`

Die Daten der bisherigen Session sollen gelöscht und anschließend die gesamte Session beendet werden.

```

<?php
①    session_start();
?><!DOCTYPE html>
<html>
    <head> <meta charset="UTF-8">
        <title>Session-Daten und Session löschen</title></head>
    <body>
        <h1>Session-Daten und Session löschen</h1>
        <?php
            echo "<pre>";
②        print_r($_SESSION);
            echo "</pre>";
③        unset($_SESSION["vorname"]);
            echo "<pre>";
④        print_r($_SESSION);
            echo "</pre>";
⑤        $_SESSION = array();
            print_r($_SESSION);
            echo "<p>Die Session mit der ID " . session_id() .
                " wurde ";
            if (session_destroy()) {
                echo "erfolgreich gelöscht.";
            } else {
                echo "nicht gelöscht.";
            }
            echo "</p>";
        ?>
    </body>
</html>

```

- ① Die aktuelle Session wird fortgesetzt.
- ② Mit der Funktion `print_r()` geben Sie zur Kontrolle die Session-Variablen mehrmals am Bildschirm aus. Die Ausgabe des `pre`-Tags vor und hinter dem `print_r()` dient der zeilenweisen Ausgabe im Browser.
- ③ Mit der Funktion `unset()` löschen Sie die angegebene Session-Variablen (`$_SESSION["vorname"]`). Alle anderen Session-Variablen bleiben unberührt. Die Session läuft weiter.
- ④ Indem Sie der Variablen `$_SESSION` ein leeres Array zuweisen, löschen Sie alle Session-Variablen.



Ausgabe der Beispieldatei „session_destroy.php“

- ⑤ Danach löschen Sie die Session-Datei und damit die Session. Die Funktion `session_destroy()` liefert einen Wert zurück, und zwar TRUE, wenn der Löschkvorgang erfolgreich war, ansonsten FALSE. Über eine `if`-Abfrage generieren Sie eine entsprechende Ausgabe.

11.6 Fallbeispiel „Shop“

Sie haben in den vorigen Abschnitten die grundlegenden Funktionen rund um Sessions kennengelernt. Die bisherigen Beispiele waren allerdings linear aufgebaut. In typischen Anwendungsbeispielen von Sessions wie z. B. Shops werden Sie jedoch mit weiteren Problemen konfrontiert:

- ✓ Sie arbeiten in der Regel innerhalb einer Session mit mehreren Formularen und müssen achtgeben, dass sich in der Session gespeicherte Daten der einzelnen Formulare nicht gegenseitig überschreiben.
- ✓ Sie rufen ein Formular eventuell mehrfach auf. Im Idealfall sind bestellte Artikelmengen eingetragen und Sie können jederzeit weitere Artikel bestellen oder Bestellmengen verändern.
- ✓ Sie können die Seiten beliebig innerhalb der Session wechseln. Daten dürfen dabei nicht verloren gehen.

Beschreibung des Fallbeispiels

Um die genannten Sachverhalte abzubilden, zeigt das Fallbeispiel einen kleinen Shop mit zwei Formularen (= Artikelgruppen), über welche Sie Artikel bestellen können. Sie können jederzeit zum Warenkorb sowie auf die einzelnen Formularseiten wechseln. Um den Bestellvorgang zu beenden, geben Sie auf einer abschließenden Seite Ihre persönlichen Daten ein. Die Daten zur Bestellung werden ausgegeben und in einer *.csv-Datei gespeichert. Danach wird die Session beendet.

Das Fallbeispiel ist möglichst einfach gehalten. Aus Gründen der Übersichtlichkeit und Nachvollziehbarkeit wurden weder ein ansprechendes Layout noch zusätzliche Funktionen programmiert.

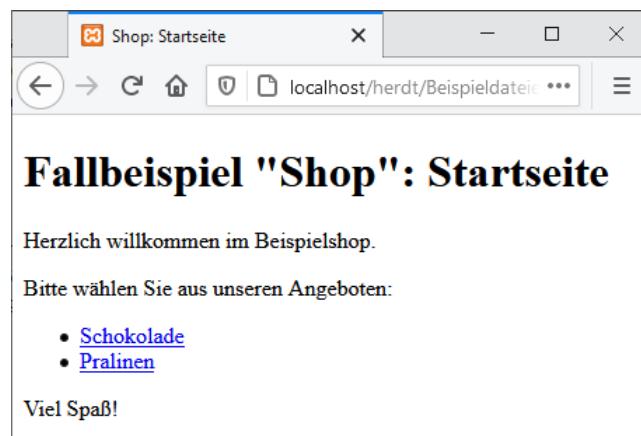
Das Shopbeispiel besteht aus folgenden Dateien:

Dateiname	Kurzbeschreibung
<code>fallbeispiel_start.php</code>	Startseite des Shops, mit Links auf die Formulare zur Bestellung von Schokolade und Pralinen
<code>fallbeispiel_artikel.inc.php</code>	Include-Datei, in der die verwendeten Artikel in Array-Variablen zur Verfügung gestellt werden
<code>fallbeispiel_form-schoko.php</code> <code>fallbeispiel_form-praline.php</code>	Formulardateien zur Bestellung von Artikeln. Pro Artikelgruppe wird eine separate Formulardatei verwendet.
<code>fallbeispiel_warenkorb.php</code>	Zentrale Auswertungsdatei für die übermittelten Daten aus den Formularen und dem Warenkorb
<code>fallbeispiel_kasse.php</code>	Datei zum Abschließen des Bestellvorgangs, Eingabe persönlicher Daten, Speichern der Bestellung in einer externen Datei und Beenden der Session

Beispiel: fallbeispiel_start.php

Die Startseite des Fallbeispiels verlinkt auf die Formulare im Shop. Die Datei trägt die Dateinamenerweiterung `*.php`, die Session wird über die Funktion `session_start()` initialisiert.

Beachten Sie, dass eine Session gestartet wird, obwohl die Datei abgesehen von der Initialisierung der Session ausschließlich aus HTML-Elementen besteht.



Ausgabe der Startseite des Fallbeispiels „fallbeispiel_start.php“

Beispiel: fallbeispiel_artikel.inc.php

In der Include-Datei werden in Array-Variablen die Artikelnummern und Artikelbezeichnungen definiert. In der Praxis werden diese Angaben in der Regel aus Datenbanken oder vergleichbaren Quellen ausgelesen und über entsprechende Variablen zur Verfügung gestellt.

```
<?php
① $array_schoko = array ("s-1" => "Weiße Schokolade",
                         "s-2" => "Vollmilch-Schokolade",
                         "s-3" => "Bio-Vollmilch-Schokolade",
                         "s-4" => "Zartbitter-Schokolade");
$array_praline = array ("p-1" => "Marzipan-Pralinen",
                        "p-2" => "Mokka-Pralinen",
                        "p-3" => "Nougat-Pralinen",
                        "p-4" => "Walnuss-Pralinen");
```

- ① Die Variable `$array_schoko` wird als Array-Variable definiert. Gespeichert wird ein assoziatives Array mit Angaben zu Artikelnummern (Schlüssel) und Artikelbezeichnungen (Werte). Analog dazu wird für jede weitere Artikelgruppe eine weitere Array-Variable nach gleichem Muster angelegt. Gerade bei inkludierten Dateien können Leerzeichen am Ende der Datei zu Problemen führen. Auf den schließenden PHP-Tag `?>` wird deswegen in dieser Datei auch verzichtet.

Beispiel: fallbeispiel_form-schoko.php (fallbeispiel_form-praline.php)

Die beiden Formulare sind nach dem gleichen Muster aufgebaut. Die Artikel werden aus der Datei `fallbeispiel_artikel.inc.php` eingelesen. Der Benutzer kann für die Artikel die gewünschten Bestellmengen eingeben. Bei nochmaligem Aufruf der Formulardatei wird eine eventuell früher eingetragene Bestellmenge aus der Session-Variablen ausgelesen und angezeigt. In diesem Beispiel können Sie durch Eingabe des Wertes `0` bei der Bestellmenge den Artikel aus der Session löschen.

```

<?php
①    session_start();
②    include("fallbeispiel_artikel.inc.php");
?><!DOCTYPE html>
<html>
    <head><meta charset="UTF-8">
        <title>Schokolade-Bestellformular</title></head>
    <body>
        <h1>Fallbeispiel "Shop": Formular 1 - Schokolade</h1>
        <p>Bestellung: Schokolade - tragen Sie die gewünschte Menge ein.</p>
③    <form action="fallbeispiel_warenkorb.php" method="POST">
        <table border="1" bgcolor="#D5F0F5">
            <tr><th>Art.-  

                Nr.</th><th>Artikel</th><th>Menge</th><th>Einheit</th></tr>
            <?php
④        foreach ($array_schoko as $key => $value) {
            echo "<tr><td align='center'>$key</td><td>$value</td>";
⑤            echo "<td><input type='text' name='$key' value=''";
            if (isset($_SESSION[$key])) ? $_SESSION[$key] : '0')
                . "' size='5' style='text-align:right'>";
            echo "</td><td>Tafel (100g)</td></tr>";
        }
        ?>
        <tr>
            <td colspan="4">
                <input type="submit" name="schoko" value="In den Warenkorb">
                <input type="submit" name="abbruch" value="Abbrechen">
            </td>
        </tr>
    </table></form></body></html>

```

- ① Durch Angabe von `session_start()` wird eine Session gestartet bzw. eine bestehende Session fortgesetzt.
- ② Per `include()` wird die Datei `fallbeispiel_artikel.inc.php` eingebunden, die Informationen zum Warenbestand enthält.
- ③ Die Datei `fallbeispiel_warenkorb.php` wird als Ziel-Skript des Formulars und damit zur Auswertung der Eingaben definiert.
- ④ In einer `foreach`-Schleife wird das Array der Artikel der gewünschten Warengruppe ausgelesen. Schlüssel und Wert werden flexibel in einer Tabelle dargestellt. In der Schleife wird pro Artikel ein Eingabefeld generiert, in das der Benutzer die Bestellmenge eingeben kann.

Art.-Nr.	Artikel	Menge	Einheit
s-1	Weiße Schokolade	5	Tafel (100g)
s-2	Vollmilch-Schokolade	0	Tafel (100g)
s-3	Bio-Vollmilch-Schokolade	0	Tafel (100g)
s-4	Zartbitter-Schokolade	3	Tafel (100g)

Anzeige der Beispieldatei „fallbeispiel_form-schoko.php“ mit einigen Beispieldaten

- ⑤ Über das Attribut `value` des `input`-Elements wird die Vorbelegung gesetzt. Dabei wird über `$key` die Session-Variable `$_SESSION` geprüft, ob für den Artikel bereits ein Wert gespeichert ist. Falls ja, wird dieser für die Vorbelegung des `input`-Elements verwendet, ansonsten wird das Feld mit 0 vorbelegt. Zu diesem Zweck wird im Beispiel die Kurzschreibweise der `if-else`-Anweisung – der ternäre Operator – verwendet.
- ⑥ Die `input`-Elemente ⑥ und ⑦ werden durch den Wert `submit` für das `type`-Attribut zur Absenden-Schaltfläche. Das `name`-Attribut wird mit `schoko` belegt, worauf das Ziel-Skript später reagiert.
- ⑦ Für die zweite Submit-Schaltfläche wird im `name`-Attribut der Wert `abbruch` vergeben. Das signalisiert später dem Ziel-Skript, dass, wenn dieser Button geklickt wird, keine Daten gespeichert werden sollen.

Beispiel: *fallbeispiel_warenkorb.php*

Die Auswertung der Formulareingaben der beiden Bestellformulare erfolgt in einer einzigen Datei, der Datei *fallbeispiel_warenkorb.php*. Durch eine `if`-Anweisung wird geprüft, aus welchem Formular Daten übergeben wurden. Die Daten werden in die Session-Datei übernommen, ohne dass bereits vorhandene Daten anderer Formulare beeinflusst werden. Das Skript dient einem weiteren Zweck, nämlich der Anzeige des Warenkorbs. In der Datei sind Verlinkungen auf alle anderen Dateien innerhalb des Shops enthalten.

```

<?php
    session_start();
    include("fallbeispiel_artikel.inc.php");
?><!DOCTYPE html>
<html>
    <head><meta charset="UTF-8">
        <title>Ihr Warenkorb</title></head>
    <body>
        <h1>Ihr Warenkorb</h1>
        <?php
            if (isset($_POST["schoko"]) or isset($_POST["praline"])) {
                foreach ($_POST as $key => $value) {
                    if (!is_numeric($value)) {
                        continue;
                    }
                    if ($value >= 1) {
                        $_SESSION[$key] = intval($value);
                    } else {
                        if (isset($_SESSION[$key])) {
                            unset($_SESSION[$key]);
                        }
                    }
                }
            }
            echo "<table border='1'>
                <tr><th>Art.-Nr.</th><th>Artikel</th>
                    <th>Menge</th></tr>";
            foreach ($_SESSION as $key => $value) {

```

```

⑩    if (substr($key, 0, 1) == "s") {
        echo "<tr><td>$key</td><td>$array_schoko[$key]</td>
              <td>$value</td></tr>";
    }
⑩    if (substr($key, 0, 1) == "p") {
        echo "<tr><td>$key</td><td>$array_praline[$key]</td>
              <td>$value</td></tr>";
    }
}
echo "</table>";
?>
<p>Was möchten Sie tun?</p>
<ul>
    <li><a href="fallbeispiel_form-schoko.php">Schokolade
        bestellen</a></li>
    <li><a href="fallbeispiel_form-praline.php">Pralinen
        bestellen</a></li>
    <li><a href="fallbeispiel_kasse.php">Bestellung
        abschließen</a></li>
</ul>
</body></html>

```

- ① Durch Angabe von `session_start()` wird eine Session gestartet bzw. eine bestehende Session fortgesetzt.
- ② Per `include()` wird die Datei `fallbeispiel_artikel.inc.php` eingebunden, welche Informationen zum Warenbestand enthält.
- ③ Es wird geprüft, ob die Variable `$_POST["schoko"]` oder `$_POST["praline"]` gesetzt ist. Das ist nur dann der Fall, wenn Daten aus einer der Formulardateien zur Auswertung versendet wurden und dort die entsprechende Schaltfläche geklickt wurde. Je nachdem, ob Sie vom Formular für Pralinen- oder von dem für Schokoladenbestellung zu diesem Skript gelangt sind, werden nur die dort vorgenommen Eintragungen in der Session gespeichert. Durch dieses Vorgehen können sich Daten aus verschiedenen Formularen nicht gegenseitig überschreiben. Falls ein Nutzer die Abbrechen-Schaltfläche geklickt hat, ist lediglich `$_POST["abbruch"]` gesetzt, in dem Fall wird nichts an der Session verändert.
- ④ Über eine `foreach`-Schleife werden alle Formulardaten (`$_POST`) durchlaufen.
- ⑤ Über die Funktion `is_numeric()` wird geprüft, ob der eingegebene Wert numerisch ist. Damit werden fehlerhafte Eingaben abgefangen, aber auch die Schalterflächen, die ebenfalls im `$_POST`-Array vorhanden sind und keine Zahl-Werte haben. Falls keine Zahl festgestellt werden kann, wird per `continue` zum nächste Schleifendurchlauf gesprungen.

Art.-Nr.	Artikel	Menge
s-1	Weiße Schokolade	5
s-4	Zartbitter-Schokolade	3
p-1	Marzipan-Pralinen	2
p-2	Mokka-Pralinen	1
p-4	Walnuss-Pralinen	12

Was möchten Sie tun?

- [Schokolade bestellen](#)
- [Pralinen bestellen](#)
- [Bestellung abschließen](#)

Anzeige des Warenkorbs im Fallbeispiel
(Datei „fallbeispiel_warenkorb.php“)

- ⑥+⑦ Die Artikelnummer bildet den Schlüssel (`$key`), die vom Benutzer eingegebene Bestellmenge den Wert (`$value`). Es wird abgefragt, ob mindestens der Wert 1 bei der Bestellmenge eingegeben wurde. Auf den Wert `$value` wird die Funktion `intval()` angewendet. Diese Funktion wandelt alle übergebenen Werte in einen Integer um. Damit fangen Sie falsche Eingaben ab und stellen sicher, dass Sie ausschließlich Ganzzahlen als sinnvolle Bestellwerte haben. Der Wert wird in der Session gespeichert.
- ⑧ Im `else`-Zweig wird die Variable für diesen Artikel – sofern in der Session vorhanden – aus der Session-Datei gelöscht. Ein Eintrag von 0 im Bestellformular löscht den betreffenden Eintrag wieder aus der Session, da die Prüfung in der `if`-Anweisung durch `$value >= 1` die Eingabe 0 nicht besteht und damit der `else`-Zweig ausgeführt wird.
- ⑨ Analog zur `foreach`-Schleife zum Einlesen der Formulardaten ④ erfolgt hier das Einlesen der Session-Daten, um den aktuellen Warenkorb darstellen zu können.
- ⑩ Über die Funktion `substr($key, 0, 1)` wird der erste Buchstabe der Artikelnummer ausgelesen. In diesem Beispiel sind nur die Werte `s` (Schokolade) oder `p` (Praline) möglich. Damit werden die Artikelbezeichnungen aus dem richtigen Artikel-Array entnommen.

Beispiel: fallbeispiel_kasse.php

Die Bestellung soll abgeschlossen werden. Der Benutzer gibt zu diesem Zweck seine persönlichen Daten ein (in diesem Beispiel nur Vorname, Name und Ort). Danach werden die kompletten Bestelldaten am Bildschirm angezeigt und in einer *.csv-Datei zur Weiterverarbeitung mit Excel gespeichert. Danach wird der Inhalt der Session-Variablen `$_SESSION` und abschließend die komplette Session gelöscht.

```
<?php
① session_start();
② include("fallbeispiel_artikel.inc.php");
?><!DOCTYPE html><html>
<head><meta charset="UTF-8"><title>Kasse</title></head>
<body>
    <h1>Fallbeispiel "Shop": Bestellung abschließen</h1>
    <?php
        ③ if (isset($_POST["absenden"])) {
            ④ $vorname = $_POST["vorname"];
            $nachname = $_POST["nachname"];
            $ort = $_POST["ort"];
            echo "<p>Sie haben folgende Bestellung
                übermittelt:</p>";
            echo "<p><strong>$vorname $nachname aus
                $ort</strong></p>";
            echo "<table border='1'><tr><th>Art.-Nr.</th>
                <th>Artikel</th><th>Menge</th></tr>";
            ⑤ $bestellung = "Art.-Nr.;Artikel;Menge\n";
        }
    <?php
</body>
</html>
```

```

⑥    foreach ($_SESSION as $key => $value) {
      if (substr($key, 0, 1) == "s") {
          echo "<tr><td>$key</td><td>$array_schoko[$key]</td>
                <td>$value</td></tr>";
          $bestellung .= "$key;$array_schoko[$key];$value\n";
      }
      if (substr($key, 0, 1) == "p") {
          echo "<tr><td>$key</td><td>$array_praline[$key]</td>
                <td>$value</td></tr>";
          $bestellung .= "$key;$array_praline[$key];$value\n";
      }
  }
$bestellung .=
    "\nbestellt von\n$vorname;$nachname;$ort\n\n";
echo "</table><p>Vielen Dank! Die Session wird
beendet.</p>";
if (file_put_contents("bestellung.csv", $bestellung,
    FILE_APPEND)) {
    echo "<p><em>Die Bestelldaten wurden in der Datei
bestellung.csv gespeichert</em></p>";
}
⑧ $_SESSION = array();
session_destroy();
⑨ } else {
?
<p>Bitte füllen Sie die nachfolgenden Eingabefelder
aus: </p>
<form action="<?php echo $_SERVER["PHP_SELF"]; ?>" 
      method="POST">
    <p>Vorname: <input type="text" name="vorname"></p>
    <p>Nachname: <input type="text" name="nachname"></p>
    <p>Wohnort: <input type="text" name="ort"></p>
    <p><input type="submit" name="absenden"
           value="Absenden und Bestellung abschließen"></p>
</form>
<?php
}
?
</body></html>

```

- ① Durch Angabe von **session_start()** wird eine Session gestartet bzw. eine bestehende Session fortgesetzt.
- ② Per **include()** wird die Datei *fallbeispiel_artikel.inc.php* eingebunden, welche Informationen zum Warenbestand enthält.
- ③ Zum Abschluss der Bestellung steht in dieser Datei das Formular zur Eingabe persönlicher Daten zur Verfügung ⑩. Mit der Funktion **isset(\$_POST["absenden"])** wird geprüft, ob das Formular bereits abgesendet wurde. Das Formular wird nur angezeigt, wenn es noch nicht abgesendet wurde ⑨.

- ④ Die Formulardaten werden zur einfachen Verwendung in den Variablen \$vorname, \$nachname und \$ort gespeichert.
- ⑤ Die Variable \$bestellung wird definiert und nach und nach mit Inhalt gefüllt. Die Variable beinhaltet später alle Bestelldaten.
- ⑥ Auch in diesem Skript erfolgten das Auslesen der Session-Daten und die Darstellung des Warenkorb-Inhalts auf dem Bildschirm.
- ⑦ Die Variable \$bestellung wird mit der Funktion `file_put_contents()` in die Datei `bestellung.csv` im selben Ordner gespeichert. Über den Schalter `FILE_APPEND` werden die Daten an eventuell bestehende Daten angehängt. Im Erfolgsfall wird eine entsprechende Meldung auf dem Bildschirm ausgegeben.
- ⑧ Schließlich wird die Variable `$_SESSION` geleert und die komplette Session gelöscht.

Fallbeispiel "Shop": Bestellung abschließen

Bitte füllen Sie die nachfolgenden Eingabefelder aus:

Vorname: Nachname: Wohnort:

Absenden und Bestellung abschließen

Art.-Nr.	Artikel	Menge
s-1	Weiße Schokolade	5
s-4	Zartbitter-Schokolade	3
p-1	Marzipan-Pralinen	2
p-2	Mokka-Pralinen	1
p-4	Walnuss-Pralinen	12

Vielen Dank! Die Session wird beendet.

Die Bestelldaten wurden in der Datei `bestellung.csv` gespeichert

Anzeige der Beispieldatei „fallbeispiel_kasse.php“ bei Eintrag persönlicher Daten (links) und bei Ende des Bestellvorgangs (rechts)

Am Ende der Session wurden die Bestelldaten per `file_put_contents()` in die *.csv-Datei geschrieben. Der Inhalt der Datei sieht wie folgt aus:

```
Art.-Nr.;Artikel;Menge
s-1;Weiße Schokolade;5
s-4;Zartbitter-Schokolade;3
p-1;Marzipan-Pralinen;2
p-2;Mokka-Pralinen;1
p-4;Walnuss-Pralinen;12

bestellt von
Maxi;Müller;Posemuckel
```

Inhalt der Datei „bestellung.csv“ nach Ausführung einer Beispielbestellung

11.7 Übung

Einen einfachen Shop erstellen

Level		Zeit	ca. 30 min
Übungsinhalte	<ul style="list-style-type: none"> ✓ Arbeiten mit HTML-Formularen ✓ Daten in einer Session speichern und wiederverwenden ✓ Verwenden von Session-Funktionen 		
Übungsdatei	--		
Ergebnisdateien	<i>u_formular.php, u_bestellung.php, u_abschluss.php</i>		

1. Erstellen Sie ein Bestellformular zur Bestellung von Honig (*u_formular.php*). Starten Sie mit dieser Datei eine Session. Die in das Bestellformular eingegebenen Daten sollen an die Datei *u_bestellung.php* übergeben werden.

Honig	Menge
Akazienhonig	<input type="text"/>
Heidehonig	<input type="text"/>
Kleehonig	<input type="text"/>
Tannenhonig	<input type="text"/>
<input type="button" value="Abschicken"/>	

Übungsdatei „u_formular.php“

2. Speichern Sie in der Datei *u_bestellung.php* die Daten aus dem Formular in der Session und lassen Sie die Session-Daten inklusive Session-ID anzeigen.

Sie haben folgende Mengen bestellt:

Akazienhonig: 6 Gläser
Heidehonig: 7 Gläser
Tannenhonig: 12 Gläser

Die Session-ID lautet: *6elnabd5t8fla74b9cv363poe*

[Weiter zur Eingabe persönlicher Daten und dem Abschluss der Bestellung.](#)

Anzeige der Übungsdatei „u_bestellung.php“

3. Verlinken Sie auf eine weitere Datei innerhalb der Session (*u_abschluss.php*), in der in einem Formular Angaben zu Name, Wohnort und Mailadresse gemacht werden können.

Honigbestellung - Abschluss

Bitte geben Sie noch Ihre Kontaktdaten ein:

Vorname : Maxi

Nachname : Müller

Wohnort : Posemuckel

Mailadresse : maxi@puppetmail.de

Abschicken

Anzeige der Übungsdatei „*u_abschluss.php*“ zur Eingabe persönlicher Daten

4. Lesen Sie anschließend nach Absenden des Formulars in derselben Datei die kompletten Session-Daten über eine Schleife aus und geben Sie sie am Bildschirm aus. Abschließend sollen die Session-Variable geleert und die Session beendet werden.

Dies sind die in der Session gesammelten Daten:

Akazienhonig: 6
Heidehonig: 7
Kleehonig:
Tannenhonig: 12
Vorname: Maxi
Nachname: Müller
Wohnort: Posemuckel
Mailadresse: maxi@puppetmail.de

Damit ist die Session beendet. [Klicken Sie hier](#), um eine neue Session zu beginnen.

Anzeige der Zusammenfassung der Session-Daten

5. Zusatzaufgabe: Schauen Sie sich das Fallbeispiel auf den vorherigen Seiten genau an und überlegen Sie, welche Anpassungen Sie vornehmen können, z. B. Layout, andere oder weitere Artikel, Erweitern des Beispiels um Preisangaben und -berechnungen. Setzen Sie eine Ihrer Ideen auf der Basis der vorliegenden Dateien um.

12

Grundlagen Datenbank MySQL



Beispieldateien: Dateien aus Ordner *Kapitel_12*

12.1 Die Datenbanken MySQL und MariaDB

PHP unterstützt die Arbeit mit verschiedenen Datenbankmanagementsystemen. Dabei wird PHP häufig in einem Atemzug mit MySQL genannt. In der Vergangenheit galt die Kombination der Programmiersprache PHP mit der MySQL-Datenbank als am weitesten verbreitet auf den Webservern weltweit.

Durch die Übernahme von MySQL durch Sun Oracle im Jahr 2008 war jedoch nicht mehr sicher, ob MySQL auch langfristig kostenfrei nutzbar sein würde und ob in Zukunft die Datenbank weiterentwickelt und Bugs beseitigt würden. Durch den Wechsel zur Datenbank **MariaDB** wurde eine sichere Alternative geschaffen. MariaDB ist durch eine Abspaltung von MySQL entstanden, was eine hohe Kompatibilität der beiden Datenbanken erklärt. Vor allem steht MariaDB unter der **GNU General Public License**. Das bedeutet, diese Datenbank ist auch in der Zukunft frei von Markenrechten. Die Gefahr, dass die Datenbank kostenpflichtig wird, ist nicht gegeben.



Der XAMPP-Webserver verwendet seit Ende 2015 statt MySQL das Datenbanksystem MariaDB. Im Sprachgebrauch wird zumeist von MySQL gesprochen, auch wenn PHP-Entwickler mit einer MariaDB-Datenbank arbeiten. Auch die Nutzungsoberfläche *phpMyAdmin* und das *XAMPP Control Panel* verwenden die Bezeichnung MySQL. Der Zugriff auf eine MariaDB-Datenbank geschieht ebenfalls über die herkömmlichen MySQL-Funktionen. Von daher wird in diesem Buch ebenfalls von MySQL gesprochen, wobei stets auch MariaDB gemeint ist.

12.2 MySQL-Datenbanken mit phpMyAdmin verwalten

In dem für dieses Buch eingesetzten Webserver-Paket XAMPP (Version 8.0.1) sind das Datenbankmanagementsystem MariaDB, Version 10.4.17, und phpMyAdmin, Version 5.0.4, enthalten.

Der XAMPP-Webserver wird regelmäßig auf neue PHP-, MySQL- bzw. MariaDB- und phpMyAdmin-Versionen aktualisiert (nicht jede neue Version einer Software wird integriert, einzelne Versionen von PHP bzw. MySQL werden mitunter übersprungen). So werden Sie in Zukunft XAMPP-Installationspakete finden, welche neuere Versionen der einzelnen Komponenten berücksichtigen. Diese können im Detail (Layout, Funktionen) von der in diesem Buch verwendeten Version abweichen.

MySQL



XAMPP Control Panel: Schaltzentrale zum Starten und Beenden der Komponenten. Die Modulnamen Apache und MySQL sind hellgrün hinterlegt, wenn die Server laufen (Windows) bzw. durch grüne Symbole in der Bedienoberfläche der XAMPP-Applikation (macOS, siehe Anhang).

MySQL und MariaDB sind Datenbankmanagementsysteme, die weitgehend den SQL-Standard unterstützen. Mithilfe von SQL-Befehlen werden Datenbanken, Tabellen und die darin enthaltenen Datensätze sowie Benutzer und ihre Berechtigungen verwaltet.

Zur Verwaltung von MySQL- und MariaDB-Datenbanken wird häufig die Bedienoberfläche phpMyAdmin verwendet. Beide Komponenten (MariaDB und phpMyAdmin) werden bei der Installation von XAMPP mit eingerichtet. Achten Sie im XAMPP Control Panel darauf, dass der **Apache-Webserver** und der **MySQL-Server** gestartet sind. Nur dann können Sie mit PHP und der Datenbank arbeiten. Falls das nicht der Fall sein sollte, starten Sie beide Komponenten über einen Klick auf die jeweils nebenstehende Schaltfläche *Start*.



Über die Checkboxen am linken Rand können Sie die Server als Dienste aktivieren, sodass diese beim Start des Rechners automatisch gestartet werden und das händische Starten überflüssig wird. Unter Windows kann es abhängig von der Benutzerkontensteuerung sein, dass Sie die grünen Häkchen im Control Panel nicht sehen und nicht bedienen können. In diesem Fall deaktivieren Sie die Benutzerkontensteuerung oder öffnen das Control Panel als Administrator.

PHP ist nicht auf MySQL beschränkt, sondern kann auch mit anderen Datenbanken arbeiten. Für verschiedene Datenbanken bietet PHP eigene Datenbankfunktionen an (z. B. für PostgreSQL oder SQLite). Alternativ kann über eine Abstraktionsschicht auf verschiedene Datenbanken zugegriffen werden (z. B. über PHP Data Objects, kurz PDO). Die Interaktionen zwischen PHP und den unterschiedlichen Datenbanken sind aber prinzipiell vergleichbar. Das Arbeiten mit Datenbanken wird in diesem Buch mit MySQL sowie der Nutzeroberfläche von phpMyAdmin vorgestellt.

phpMyAdmin

Bei phpMyAdmin handelt es sich um eine in PHP geschriebene Open-Source-Web-Applikation, also eine Sammlung von PHP-Skripten, mit der Sie MySQL-Datenbanken über eine grafische Weboberfläche verwalten können. phpMyAdmin wird hauptsächlich zur Verwaltung von MySQL-Datenbanken auf Webservern verwendet, auf denen die einzelnen Kunden keine Rechte haben, die von MySQL zur Verfügung gestellten Kommandozeilen-Verwaltungstools direkt auszuführen. Die meisten Internetprovider bieten phpMyAdmin als Standardtool für die Administration von MySQL-Datenbanken an.

phpMyAdmin ermöglicht Ihnen folgende Aktionen:

- ✓ Erstellen und Löschen von Datenbanken
- ✓ Erstellen, Kopieren, Löschen und Ändern von Tabellen
- ✓ Hinzufügen, Löschen und Ändern von Datensätzen
- ✓ Ausführen von SQL-Anweisungen
- ✓ Anzeigen der Datensätze in den Tabellen
- ✓ Laden von Textdaten in Tabellen
- ✓ Exportieren von Daten in unterschiedliche Formate, beispielsweise in CSV- oder SQL-Dateien
- ✓ Importieren von Daten
- ✓ Administrieren verschiedener Server und einzelner Datenbanken
- ✓ Verwaltung von MySQL-Benutzern

Die Startseite von phpMyAdmin rufen Sie über die Navigation der XAMPP-Oberfläche auf, die Sie über die Eingabe von <http://localhost> im Browser erreichen. Alternativ können Sie die direkte URL <http://localhost/phpmyadmin/> bzw. unter <http://localhost:8080/phpmyadmin/> auf macOS aufrufen.

The screenshot shows the phpMyAdmin interface with several panels:

- Left sidebar:** Shows a tree view of databases: Neu, information_schema, mysql, performance_schema, phpmyadmin, and test. A circled '1' is next to this sidebar.
- Header:** Shows the URL localhost / 127.0.0.1 | phpMyAdmin and the path localhost/phpmyadmin/.
- Top navigation bar:** Contains links for Datenbanken, SQL, Status, Benutzerkonten, Exportieren, Importieren, Einstellungen, and Mehr.
- Allgemeine Einstellungen (General Settings) panel:** Shows password change, connection encoding (utf8mb4_unicode_ci), and further settings. A circled '2' is next to this panel.
- Datenbank-Server (Database Server) panel:** Lists server details: 127.0.0.1 via TCP/IP, MariaDB server type, SSL connection status, version 10.4.17-MariaDB, root@localhost user, and UTF-8 Unicode encoding.
- Anzeige-Einstellungen (Display Settings) panel:** Shows language (Deutsch - German) and design (pmahomme). A circled '3' is next to this panel.
- Webserver panel:** Lists Apache 2.4.46, PHP 8.0.1, and MySQL 8.0.1 versions.
- phpMyAdmin panel:** Lists version information (5.0.4), documentation, and support links.
- Bottom footer:** Shows Konsole and a copyright notice: © HERDT-Verlag 2012.

Startseite von phpMyAdmin

Über diese Oberfläche können die verschiedenen MySQL-Befehle per Hyperlink, Schaltflächen oder über Eingabefelder ausgeführt werden. Auf der linken Seite finden Sie die Auflistung aller vorhandenen Datenbanken des MySQL-Servers ①. Mit der Auswahl einer Datenbank öffnen Sie die entsprechenden Tabellen. Im rechten Hauptbildschirm ② werden die einzelnen Funktionen als Eingabefelder und Schaltflächen angeboten. Auf der Startseite können Sie die Sprache auswählen, die von phpMyAdmin verwendet werden soll. Hier können Sie auch zum Layout vorheriger Versionen wechseln ③.

12.3 MySQL-Datenbanken mit phpMyAdmin erstellen

Neue Datenbanken in MySQL anlegen

- ▶ Um eine neue Datenbank zu erzeugen, wählen Sie in der oberen Navigation den Bereich *Datenbanken* aus. Geben Sie in das Eingabefeld *Neue Datenbank anlegen* den Namen für die neue Datenbank ein, in unserem Beispiel „*obstladen*“. Mit Klick auf die Schaltfläche *Anlegen* erzeugen Sie die neue Datenbank.

Datenbanken

Neue Datenbank anlegen

Datenbankname: utf8_general_ci Anlegen

Der Zeichensatz (Kollation) legt den Default-Schriftzeichensatz für die Datenbank fest. Dieser beeinflusst, wie später Datensätze sortiert werden. Auch die einzelnen Tabellen können jeweils mit einer bestimmten Kollation angelegt werden. Da seit der Version PHP 5.4 UTF-8 als Standardzeichensatz festgelegt ist, empfiehlt es sich auch hier, den Zeichensatz *utf8_general_ci* auszuwählen.

Tabellen erstellen

Haben Sie eine neue Datenbank erstellt, erkennt phpMyAdmin, dass die neue Datenbank noch keine Tabellen enthält. Es wird Ihnen ein Formular angezeigt, über welches Sie neue Tabellen erstellen können.

- ▶ Tragen Sie in das Eingabefeld *Name* ① den Tabellennamen *bestellung* und in das Eingabefeld *Anzahl der Spalten* ② die Anzahl – nämlich „6“ – ein, die in der Tabelle erstellt werden sollen.
- ▶ Betätigen Sie die Schaltfläche *OK* ③.

Erzeuge Tabelle

⚠ Es wurden keine Tabellen in der Datenbank gefunden.

Name: bestellung (1) Anzahl der Spalten: 6 (2)

(3) OK

Neue Tabelle mit phpMyAdmin erstellen

Felder erstellen

Es öffnet sich ein neues Fenster im Webbrowser, in dem die von Ihnen angegebene Anzahl an Feldern angezeigt wird.

- ▶ Tragen Sie in die vorhandenen Eingabefelder gemäß folgender Abbildung die entsprechenden Werte ein.
- ▶ Als Feldnamen verwenden Sie *id*, *vorname*, *nachname*, *ort*, *sorte* und *menge*. Für die Felder *id* und *menge* belassen Sie unter *Typ* die Vorauswahl auf *INT*. Für die anderen Felder wählen Sie den *Typ VARCHAR*.
- ▶ Die Längen für die *VARCHAR*-Felder legen Sie jeweils mit 50 Zeichen fest bzw. 20 Zeichen für *sorte*.
- ▶ Wählen Sie in der Zeile des Felds *id* unter *Index* die Option *PRIMARY* aus.
- ▶ Aktivieren Sie in gleicher Zeile die Checkbox unter *A_1* (kurz für Auto-Inkrement, kann mit „Sich selbst hochzählende Nummerierung“ umschrieben werden).

- Aktivieren Sie mit Ausnahme in der Zeile von *id* die Checkboxen in der Spalte *Null*.

Name	Typ	Länge/Werte	Standard	Kollation	Attribute	Null	Index	A_I
id	INT		Kein(e)			<input type="checkbox"/>	PRIMARY	<input checked="" type="checkbox"/>
vorname	VARCHAR	50	Kein(e)			<input checked="" type="checkbox"/>	--	<input type="checkbox"/>
nachname	VARCHAR	50	Kein(e)			<input checked="" type="checkbox"/>	--	<input type="checkbox"/>
ort	VARCHAR	50	Kein(e)			<input checked="" type="checkbox"/>	--	<input type="checkbox"/>
sorte	VARCHAR	20	Kein(e)			<input checked="" type="checkbox"/>	--	<input type="checkbox"/>
menge	INT		Kein(e)			<input checked="" type="checkbox"/>	--	<input type="checkbox"/>



Der Aufbau der Beispieldatentabelle ist nur exemplarisch. Angaben wie Straße, PLZ, E-Mail-Adresse etc. fehlen. Außerdem würden Kundendaten und Bestelldaten in einer relationalen Datenbank in unterschiedlichen Tabellen verwaltet. Auch das Vorhandensein der möglichen Nullwerte soll die Arbeit mit diesem Beispiel vereinfachen.

Datentypen zuweisen

In den Feldern können Sie folgende Eintragungen bzw. Einstellungen vornehmen. Für das Beispiel sind folgende Felder zu beachten:

Feld	Beschreibung	Parameter in SQL-Syntax
Name	Tragen Sie in dieses Eingabefeld den Feldnamen ein.	<Feldname>
Typ	Aus dieser Selectbox wählen Sie den Datentyp des Feldes aus. Im Beispiel werden aus einer Vielzahl möglicher Datentypen nur die Typen <i>INT</i> (Ganzzahl) und <i>VARCHAR</i> (<i>various characters</i> , beliebiger Text) verwendet.	<Datentyp>
Länge/Werte	In dieses Eingabefeld tragen Sie die maximale Länge des Datenfeldinhaltens ein.	(AnzahlZeichen)
Standard	Wählen Sie hier den Standardwert aus, der dem Feld zugewiesen werden soll, wenn das Feld bei der Dateneingabe keinen Wert erhält.	DEFAULT '<Wert>'
Null	Wenn das Feld beim Eintragen eines Datensatzes leer bleiben darf, wird das Häkchen gesetzt, anderenfalls wird eine Eingabe für das Feld erwartet.	NULL oder NOT NULL
Index	Die Selectbox bietet u. a. die Einträge <i>Primary</i> und <i>Unique</i> zur Festlegung eines Feldes als Primärschlüssel bzw. als Feld, das eindeutige Werte enthalten muss.	PRIMARY, UNIQUE

Feld	Beschreibung	Parameter in SQL-Syntax
A_I	Über diese Checkbox können Sie den Wert auto_increment einem ganzzahligen Datenbankfeld zuweisen. Beim Einfügen eines neuen Datensatzes in die Tabelle wird der Wert des mit auto_increment gekennzeichneten Feldes automatisch durch MySQL um eins erhöht (inkrementiert). Innerhalb der Tabelle darf nur ein Feld mit diesem Attribut versehen werden, dieses Feld ist automatisch der Primärschlüssel der Tabelle.	AUTO_INCREMENT

Primärschlüssel anlegen

Um ein Feld als Primärschlüssel zu definieren, wählen Sie im Feld *Index* die entsprechende Option aus. Beachten Sie bei der Zuweisung des Primärschlüssels, dass innerhalb einer Tabelle ...

- ✓ nur **ein** Primärschlüssel zugewiesen werden darf;
- ✓ das Feld, dem der Primärschlüssel zugewiesen wird, **nicht** als Null-Feld definiert werden darf;
- ✓ das Feld, dem der Primärschlüssel zugewiesen wird, **eindeutige** Werte enthalten muss (kein Wert darf mehrfach vorhanden sein. Beim Versuch, denselben Wert mehrfach einzufügen, reagiert MySQL mit einer Fehlermeldung).

Pro Tabelle gibt es maximal einen Primärschlüssel. Die Einstellung *Unique* können Sie mehrfach definieren. Felder, die als *Unique* definiert werden, können ebenfalls nur einmal mit demselben Wert gefüllt werden. Auch hier reagiert MySQL mit einer Fehlermeldung, falls ein Wert für ein *Unique*-Feld bereits vorhanden ist und Sie diesen noch einmal eintragen möchten.

Sollten Sie in einer Tabelle kein Feld mit eindeutigen Werten haben, wird empfohlen, wie im Beispiel ein zusätzliches Feld mit einer automatisch generierten laufenden Nummerierung (Auto-Inkrement-Wert) hinzuzufügen, das als Primärschlüssel dient.

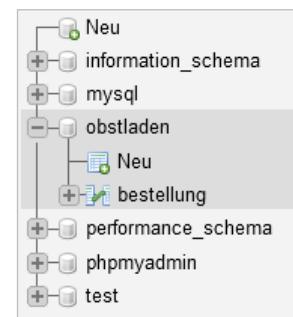
Tabelle speichern

- Betätigen Sie die Schaltfläche *Speichern*, um die Tabelle zu speichern.

Neue Tabelle in vorhandener Datenbank erstellen

Wollen Sie in einer vorhandenen Datenbank eine neue Tabelle erstellen, wählen Sie in der linken phpMyAdmin-Navigation die Datenbank aus. Unterhalb des Namens der Datenbank werden die darin enthaltenen Tabellen angezeigt sowie ein Link *Neu*, über den Sie ein neues Formular zum Anlegen einer Tabelle aufrufen können.

- Stellen Sie sicher, dass Sie sich nach dem Erstellen einer neuen Tabelle im Register *Struktur* befinden.
oder
- Klicken Sie im Startbildschirm von phpMyAdmin auf den Eintrag *Datenbanken* und in der Übersicht der vorhandenen Datenbanken auf den Namen der Datenbank, in der Sie eine Tabelle erstellen wollen.



Anzeige der Datenbanken

Beispiel: Datenbank *obstladen*

The screenshot shows the 'Tabellenstruktur' (Table Structure) page in phpMyAdmin. The table 'bestellung' has the following structure:

#	Name	Typ	Kollation	Attribute	Null	Standard	Kommentare	Extra	Aktion
1	id	int(11)	utf8_general_ci	Nein	kein(e)			AUTO_INCREMENT	Bearbeiten Löschen Mehr
2	vorname	varchar(50)	utf8_general_ci	Ja	NULL				Bearbeiten Löschen Mehr
3	nachname	varchar(50)	utf8_general_ci	Ja	NULL				Bearbeiten Löschen Mehr
4	ort	varchar(50)	utf8_general_ci	Ja	NULL				Bearbeiten Löschen Mehr
5	sorte	varchar(20)	utf8_general_ci	Ja	NULL				Bearbeiten Löschen Mehr
6	menge	int(11)		Ja	NULL				Bearbeiten Löschen Mehr

Below the table, there are several buttons: 'Alle auswählen', 'markierte:', 'Anzeigen', 'Bearbeiten', 'Löschen', 'Primärschlüssel', 'Unique', 'Index', 'Volltext', 'Zu zentralen Spalten hinzufügen', 'Zentrale Spalten entfernen', 'Drucken', 'Tabellenstruktur analysieren', 'Tabelle verfolgen', 'Spalten verschieben', 'Normalisieren'. A modal dialog at the bottom shows the current index configuration:

Aktion	Schlüsselname	Typ	Unique	Gepackt	Spalte	Kardinalität	Kollation	Null	Kommentar
Bearbeiten Löschen	PRIMARY	BTREE	Ja	Nein	id	0	A	Nein	

Buttons in the modal: 'Index über 1', 'Spalten anlegen', 'OK'.

Tabelle „bestellung“ in der Datenbank „obstladen“ in phpMyAdmin

Tabellenstruktur verändern

Die Tabellenstruktur können Sie im Nachhinein ändern, beispielsweise mit den Hyperlinks *Bearbeiten* oder *Löschen*. Diese Hyperlinks befinden sich in der Spalte *Aktion* neben den Feldnamen in der Tabellenstrukturansicht und sind in phpMyAdmin mit sprechenden Links versehen, die auf die möglichen Optionen hinweisen:

The screenshot shows the 'Tabellenstruktur' (Table Structure) page in phpMyAdmin. The table 'bestellung' has the following structure:

#	Name	Typ	Kollation	Attribute	Null	Standard	Kommentare	Extra	Aktion
1	id	int(11)	utf8_general_ci	Nein	kein(e)			AUTO_INCREMENT	Bearbeiten Löschen Mehr
2	vorname	varchar(50)	utf8_general_ci	Ja	NULL				Bearbeiten Löschen Mehr
3	nachname	varchar(50)	utf8_general_ci	Ja	NULL				Bearbeiten Löschen Mehr
4	ort	varchar(50)	utf8_general_ci	Ja	NULL				Bearbeiten Löschen Mehr
5	sorte	varchar(20)	utf8_general_ci	Ja	NULL				Bearbeiten Löschen Mehr
6	menge	int(11)		Ja	NULL				Bearbeiten Löschen Mehr

Neben dem Verändern der Tabellenstruktur haben Sie auch die Möglichkeit, neue Felder hinzuzufügen. Unterhalb der Strukturübersicht der Tabelle finden Sie folgendes Formular:

The screenshot shows the 'Spalte(n) einfügen' (Add Column) form in phpMyAdmin. The form contains the following fields:

- An input field for the number of columns: '1'.
- A dropdown menu for the position: 'nach menge'.
- A 'OK' button.

Um neue Datenfelder zur bestehenden Tabelle hinzuzufügen, geben Sie die Anzahl der neuen Felder ein und wählen Sie aus, an welcher Position innerhalb der Tabelle die neuen Felder erstellt werden sollen.

12.4 Mit einer MySQL-Tabelle arbeiten

Datensätze in eine Tabelle eintragen

Sie können Datensätze in eine MySQL-Tabelle sowohl über die phpMyAdmin-Weboberfläche als auch, wie später beschrieben, über PHP-Befehle eintragen.

- ▶ Wechseln Sie per Klick auf den Hyperlink [Tabellenname] ① zur Ansicht der Tabelle.
- ▶ Klicken Sie am oberen Bildschirmrand auf den Hyperlink *Einfügen*, um Daten für einen Datensatz einzugeben.

Im Beispiel handelt es sich bei dem Feld *id* um ein Feld mit einer automatischen Nummerierung ②. Aus diesem Grund lassen Sie das Feld leer. MySQL vergibt automatisch den nächsthöheren freien Wert bei der Speicherung des Datensatzes.

- ▶ Geben Sie für alle anderen Felder Daten ein.
- ▶ Klicken Sie auf die Schaltfläche *OK* ③, um zur Tabelle zurückzukehren, oder wählen Sie aus dem Kombinationsfeld ④ den Eintrag *anschließend einen weiteren Datensatz einfügen*, um einen weiteren Datensatz einzufügen.
- ▶ Über den Hyperlink *Anzeigen* am oberen Browserrand können Sie sich die Datensätze der Tabelle ansehen.

Spalte	Typ	Funktion	Null	Wert
<i>id</i>	int(11)			②
<i>vorname</i>	varchar(50)		<input type="checkbox"/>	Petra
<i>nachname</i>	varchar(50)		<input type="checkbox"/>	Meyer
<i>ort</i>	varchar(50)		<input type="checkbox"/>	Wien
<i>sorte</i>	varchar(20)		<input type="checkbox"/>	Gala
<i>menge</i>	int(11)		<input type="checkbox"/>	5

(3) **OK**

Als neuen Datensatz speichern **und dann** anschließend einen weiteren Datensatz einfügen **④**

SQL Vorschau **Zurücksetzen** **OK**

Einen neuen Datensatz in eine Tabelle einfügen

- Tragen Sie die Datensätze aus der folgenden Abbildung in die Tabelle ein.

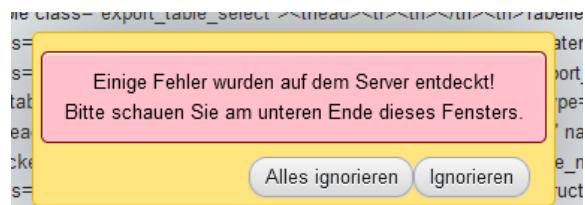
			id	vorname	nachname	ort	sorte	menge
<input type="checkbox"/>		Bearbeiten	1	Arndt	Hoffmann	Stuttgart	Elstar	15
<input type="checkbox"/>		Kopieren	2	Corinna	Delphi	Hamburg	Janagold	3
<input type="checkbox"/>		Kopieren	3	Petra	Meyer	Wien	Gala	5
<input type="checkbox"/>		Kopieren	4	Peter	Schmidt	Berlin	Elstar	25

Beispieldateneingabe „bestellung“ in Datenbank „obstladen“

12.5 SQL-Dumps exportieren und importieren

Für die Arbeit mit einem PHP-Projekt werden Sie in der Regel mit bereits vorhandenen Datenbanken arbeiten. phpMyAdmin stellt Funktionen für den Import und Export von Datenbank- und Tabellenstruktur sowie der Daten bereit.

In der aktuellen Version des phpMyAdmin tauchen beim Importieren und Exportieren Fehlermeldungen auf. Diese sind durch die Änderungen der Fehler-Level seit PHP 8.0 verursacht bzw. phpMyAdmin ist noch nicht vollständig auf PHP 8.0 umgestellt. Diese können getrost ignoriert werden, sie haben keinen Einfluss auf die korrekte Verarbeitung der Daten.



Fehlermeldung beim Import und Export

Daten exportieren

Datenbank- und Tabellenstruktur sowie die Daten selbst können per Klick aus phpMyAdmin exportiert werden. Das Ergebnis eines Exports ist ein sogenannter **SQL-Dump** (SQL-Exportdatei). Dieser kann als Datei gespeichert werden. Es handelt sich um eine Text-Datei mit SQL-Statements, die zum Import der Daten dient. SQL-Dumps werden mit der Dateinamenserweiterung **.sql** abgespeichert.

- Je nachdem, ob Sie eine ganze Datenbank oder nur eine Tabelle exportieren möchten, klicken Sie in der phpMyAdmin-Navigation links eine Datenbank oder eine Tabelle an.
- Wählen Sie in der oberen Menüleiste im Hauptfenster den Menüpunkt **Exportieren**. Falls nur die Datenbank ① angezeigt wird, können Sie alle Tabellen der Datenbank exportieren. Wird dort zusätzlich ein Tabellenname ② angezeigt, führt der Klick auf **Exportieren** zum Export der einzelnen Tabelle.



Anzeige von Datenbank und Tabelle am oberen Browserrand

phpMyAdmin bietet zwei Varianten des Datenexports, den schnellen sowie den angepassten.

- ▶ Standardmäßig ist *Schnell* ausgewählt. Hier müssen Sie keine weiteren Einstellungen vornehmen, der SQL-Dump wird mit Standardeinstellungen generiert, der für den Import der Tabellen in andere Datenbanken notwendig ist.
- ▶ Möchten Sie zusätzlich die Datenbank selber exportieren, müssen Sie unter dem Punkt *Objekterstellungsoptionen* die Auswahl *Angepasst* treffen und zusätzlich den Befehl `CREATE DATABASE` aktivieren.
- ▶ Bestätigen Sie mit *OK*. Wählen Sie im geöffneten Dialogfenster, ob Sie den SQL-Dump mit einem Editor öffnen oder auf Ihrer Festplatte speichern möchten.

Ein SQL-Dump enthält alle SQL-Befehle, die zum Anlegen von Datenbanken und Tabellen bzw. zum Einfügen von Daten notwendig sind. Die SQL-Befehle liegen genau in der Reihenfolge vor, wie sie für den Import von Datenbank- und Tabellenstruktur und Daten benötigt werden. SQL-Dumps werden mit der Dateinamenerweiterung `*.sql` gespeichert.

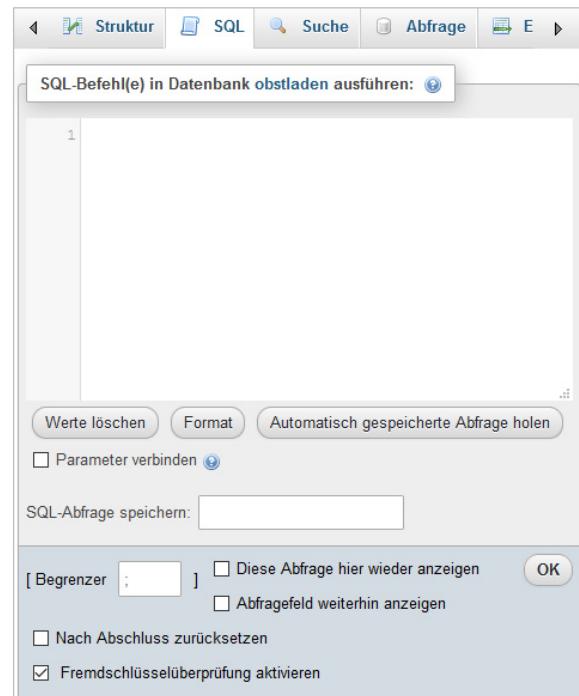
- ▶ Alternativ zum Export als Downloaddatei können Sie bei Auswahl des angepassten Datenexports die Option *Ausgabe als Text anzeigen* auswählen. In dem Fall werden die SQL-Befehle in einem Textfeld angezeigt und können von dort direkt zum Import herauskopiert werden.

Daten importieren

Der Import eines SQL-Dumps ist einfach. Um die Daten einer `*.sql`-Datei zu importieren, haben Sie zwei Möglichkeiten:

- ▶ Öffnen Sie die SQL-Datei mit einem Texteditor, z. B. Notepad++.
- ▶ Wählen Sie im phpMyAdmin in der oberen Menüleiste den Eintrag *SQL* aus. Falls Sie nur eine Tabelle in eine Datenbank importieren möchten, wählen Sie zuerst eine Datenbank aus. Sie erkennen an der obersten Leiste im Hauptfenster von phpMyAdmin, ob Sie sich auf der Datenbankserverübersicht oder auf der Übersicht einer ausgewählten Datenbank befinden.
- ▶ Nach Klick auf den Menüpunkt *SQL* öffnet sich ein Textfeld zur Eingabe. Kopieren Sie die Befehle aus der SQL-Datei in das Textfeld hinein, bestätigen Sie das Speichern mit dem Button *OK*. Die Datenbank bzw. Tabelle wird importiert, indem alle SQL-Befehle ausgeführt werden.

Im Fehlerfall zeigt phpMyAdmin eine Fehlermeldung an. Übliche Fehler sind z. B.: Eine Datenbank, die Sie importieren möchten, existiert bereits oder eine fehlerhafte SQL-Datei bzw. Datensätze mit eindeutigen IDs, die in Tabellen bereits vergeben sind.



- ▶ Alternativ können Sie auch den Menüpunkt *Importieren* wählen.
In dem Fall steht Ihnen ein Upload-Formular zur Verfügung.
- ▶ Über die Schaltfläche *Durchsuchen* wählen Sie die SQL-Datei auf Ihrer Festplatte aus.
- ▶ Bestätigen Sie den Button *OK*.



Der Import startet. Je nach Größe der Datei dauert dies einen Moment. Auch hier kann es zur Fehleranzeige kommen, welche die gleichen Ursachen haben, wie oben bereits beschrieben.

Beispieldaten importieren: *obstladen-bestellung.sql*

Die in den vorherigen Kapiteln erstellte Datenbank, die Tabelle sowie die Daten stehen als SQL-Dump unter den Beispieldaten zur Verfügung. Beachten Sie bei Import-Dateien Folgendes:

- ▶ Das CREATE-Statement für die Datenbank wird mit `CREATE DATABASE IF NOT EXISTS` eingeleitet. Die Datenbank wird nur dann erzeugt, wenn keine Datenbank mit dem Namen `obstladen` vorhanden ist.
- ▶ Das Statement zum Erstellen der Tabelle wird ebenfalls mit `CREATE TABLE IF NOT EXISTS` eingeleitet. Die Tabelle wird nur erzeugt, wenn sie noch nicht vorhanden ist.
- ▶ Sowohl Datenbank als auch Tabelle werden mit dem UTF8-Zeichensatz angelegt.
- ▶ Die Import-Datei enthält den Befehl `TRUNCATE TABLE 'bestellung' ;`. Dieser Befehl löscht eventuell vorhandene Daten aus der Tabelle `bestellung`.
- ▶ Die Daten der Tabelle werden importiert, der Primärschlüssel wird gesetzt, der Wert für das Autoinkrement wird angepasst.

12.6 PHP und MySQL

Nachdem Sie eine MySQL-Datenbank sowie eine Tabelle darin erstellt haben, können Sie über PHP Datensätze aus einer Tabelle bearbeiten, beispielsweise anzeigen, ändern, löschen, aber auch erzeugen. Die notwendigen Schritte für den Zugriff auf eine Datenbank werden in diesem Kapitel gezeigt. Zur Bearbeitung der Datenbank benötigen Sie entsprechende Berechtigungen.

! Vor PHP 7.0 war die Erweiterung *MySQL* eine Standardmethode, um auf Datenbanken zuzugreifen. Mit PHP 7.0 wurde diese Erweiterung entfernt und kann nicht mehr genutzt werden.

Stattdessen empfiehlt www.php.net, die *mysqli*-Erweiterung (MySQL Improved Extension) oder die *PDO_MYSQL*-Erweiterung (**P**HP **D**ata **O**bjects (**P**DO) **I**nterface) zu nutzen. Beide Erweiterungen nutzen den objektorientierten Ansatz der PHP-Programmierung.



Obwohl auch *mysqli* eine objektorientiert Klasse ist, bietet diese Erweiterung sogenannte Alias-Funktionen. Das sind Funktionsaufrufe, die den Funktionen der alten *MySQL*-Erweiterung in der Schreibweise ähneln, tatsächlich verbergen sich dahinter aber Methodenaufrufe der *mysqli*-Klasse. Aufgrund der einfacheren Syntax der *mysqli*-Funktionen wird in diesem Buch diese PHP-Erweiterung für das Arbeiten mit Datenbanken vorgestellt.

Benutzer für Datenbankzugriffe über phpMyAdmin anlegen

Über die obere Navigationsleiste des phpMyAdmin finden Sie den Link *Benutzerkonten*. Über diesen Link gelangen Sie zur Nutzerverwaltung der Datenbank. In der ersten Übersicht sehen Sie die bereits mit der Installation von XAMPP eingerichteten Nutzer.

	Benutzername	Hostname	Passwort	Globale Rechte	Benutzergruppe	GRANT
<input type="checkbox"/>	Jeder	%	Nein	USAGE		Nein
<input type="checkbox"/>	pma	localhost	Nein	USAGE		Nein
<input type="checkbox"/>	root	127.0.0.1	Nein	ALL PRIVILEGES		Ja
<input checked="" type="checkbox"/>	root	::1	Nein	ALL PRIVILEGES		Ja
<input type="checkbox"/>	root	localhost	Ja	ALL PRIVILEGES		Ja

Nutzer der Standardinstallation des XAMPP-Webservers

XAMPP wird standardmäßig mit einem Administrationsnutzerkonto für den Zugriff auf den Datenbankserver eingerichtet. Der Name des Nutzers ist `root`, dieser wird ohne Passwort angelegt. Dieses Nutzerkonto verfügt über die Administrationsrechte für den kompletten Datenbankserver.



Der `root`-Nutzer sollte in PHP-Skripten auf keinen Fall genutzt werden. Sowohl die vollständigen Administrationsrechte als auch das fehlende Passwort stellen ein hohes Sicherheitsrisiko dar.

PHP-Nutzer für den Zugriff aus PHP-Skripten anlegen

- ▶ Legen Sie zuerst einen Nutzer für die Verwendung in PHP-Skripten an.
- ▶ Wechseln Sie über den Link *Benutzerkonten* (phpMyAdmin, oberes Menü) in die Nutzerverwaltung.
- ▶ Unterhalb der Benutzerkontenübersicht finden Sie den Link *Benutzerkonto hinzufügen*.

Mit Klick auf den Link *Benutzerkonto hinzufügen* öffnet sich das entsprechende Formular. Der Benutzername ist frei wählbar. Das Feld *Hostname* ist beim Öffnen mit einem %-Zeichen vorausgefüllt. Dieses können Sie belassen, damit kann der Benutzer den Datenbankserver über einen beliebigen Host ansprechen, z. B. auch über die IP-Adresse des Rechners.

The screenshot shows the 'Anmeldeinformation' (Login Information) form in phpMyAdmin. The form fields are as follows:

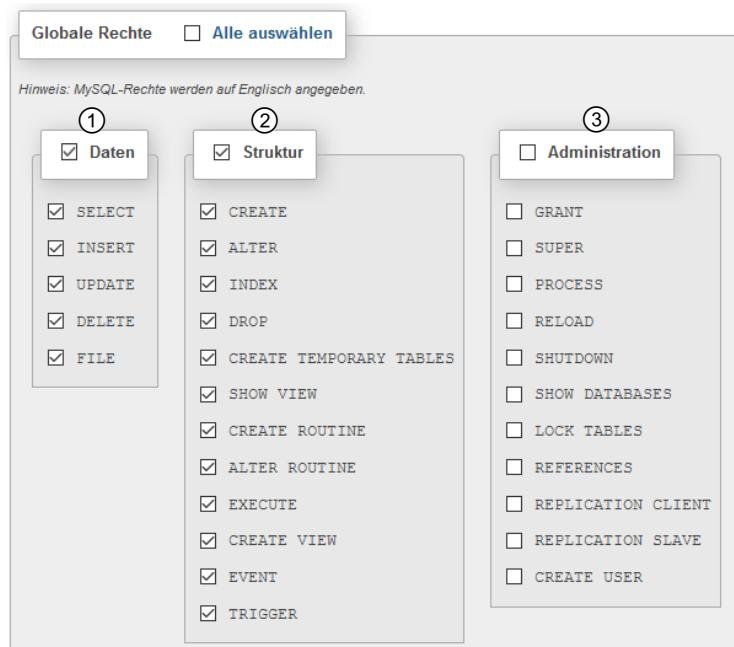
- Benutzername: Textfeld verwenden: php-user
- Hostname: Jeder Host %
- Passwort: Textfeld verwenden: Strength:
- Wiederholen:
- Authentifizierungs Plugin: Native MySQL-Authentifizierung
- Passwort generieren:

Falls Sie im Auswahlfeld *Hostname* den Eintrag *Lokal* auswählen, wird das Feld automatisch mit dem Wert *localhost* gefüllt. Auf macOS ist die Einstellung auf *localhost* empfohlen.

Damit schränken Sie den Zugriff auf den Datenbankservernamen *localhost* ein, was eine höhere Sicherheit darstellt. Vergeben Sie zusätzlich ein sicheres Passwort oder verwenden Sie die Schaltfläche *Generieren*, um ein Passwort zu erzeugen.

Zusätzlich müssen die Rechte für den Benutzer gesetzt werden.

Für einfache Datenbankabfragen reichen die Rechte aus der Gruppe *Daten* ①. Sollen über PHP auch Änderungen an der Tabellenstruktur möglich sein, vergeben Sie zusätzlich entsprechende Rechte aus der Gruppe *Struktur* ②. Administrationsrechte ③ hingegen werden einem Benutzer für PHP-Skripte nicht vergeben, dies würde ein Sicherheitsrisiko darstellen. Bei Internetprovidern sind in der Regel Rechte wie *CREATE DATABASE*, *CREATE USER* und *SET PASSWORD* geblockt.



Selbst wenn der Provider phpMyAdmin zur Verwaltung der Datenbank anbietet, können Sie in der Regel darüber keine Datenbanken löschen oder Nutzer anlegen. Für diese Aktionen bieten Provider andere Administrationsoberflächen an.

- ▶ Nach Eingabe der Daten und Vergabe der Rechte betätigen Sie die Schaltfläche *OK*. Der Benutzer wird gespeichert und steht sofort für den Einsatz in PHP-Skripten zur Verfügung.

PHP mit dem MySQL-Server verbinden

Eine Verbindung zum MySQL-Server wird aus dem PHP-Programm heraus hergestellt. Folgende Funktionen werden zum Aufbau und Beenden einer Verbindung zu einem MySQL-Server verwendet:

<code>mysqli_connect();</code>	Verbindung zum MySQL-Server aufnehmen
<code>mysqli_close();</code>	Verbindung zum verbundenen MySQL-Server beenden

Syntax der Funktion `mysqli_connect()`

```
mysqli_connect ([Server [, Benutzername [, Passwort [, Datenbank  
[, Port [, Socket]]]]]);
```

- ✓ Erster Parameter ist der Name oder die IP des Datenbankservers. Falls `NULL` oder `localhost` übergeben wird, versucht die Funktion die Verbindung zum lokalen Datenbankserver aufzunehmen.
- ✓ Zweiter Parameter ist der Datenbank-Benutzername, dritter das dazugehörige Passwort. Wird kein Benutzername angegeben, wird der Name des Benutzers verwendet, dem der Server-Prozess gehört. Wird kein Passwort angegeben, wird ein leerer Passwort benutzt.
- ✓ Als vierter Parameter kann der Name der Datenbank übergeben werden. Falls Sie diesen Parameter weglassen, stellt `mysqli_connect()` die Verbindung zum Datenbankserver her, ohne aber eine Datenbank auszuwählen. Mit der Angabe einer Datenbank wird diese mit dem Funktionsaufruf sofort ausgewählt.
- ✓ Als fünften Parameter können Sie einen Port angeben. Für die Verbindung zum Datenbankserver ist standardmäßig der Port 3306 konfiguriert, was bei den meisten Webservern auch der Fall ist. Insofern ist die Angabe eines Ports zumeist überflüssig. Ist für den Datenbankserver ein anderer Port konfiguriert, müssen Sie den entsprechenden Port angeben.
- ✓ Der letzte optionale Parameter ist der Pfad zu einem Socket, z. B. `/tmp/mysql.sock`.
- ✓ Alle Parameter von `mysqli_connect()` sind optional.
- ✓ Als Rückgabewert erhalten Sie ein Objekt, welches Sie in einer Variablen speichern. Diese Variable hat automatisch den Datentyp `object`. Diese Variable repräsentiert die Verbindung zur Datenbank, Sie benötigen diese für alle weiteren Aktionen mit der Datenbank.
- ✓ Scheitert der Verbindungsaubau zur Datenbank, gibt `mysqli_connect()` `FALSE` zurück.
- ✓ Sie können in einem PHP-Skript mehr als eine Datenbankverbindung aufbauen (z. B. weil Sie auf mehrere Datenbanken zugreifen wollen). In dem Fall vergeben Sie für jedes Datenbankobjekt eine individuelle Variable (z. B. `$verbindung1`, `$verbindung2`).

Schlägt `mysqli_connect()` fehl, wird nur eine Warnung ausgegeben, das PHP-Skript jedoch weiter ausgeführt. Dies ist häufig nicht gewünscht, da jeder weitere Befehl zur Steuerung von Datenbanken damit fehlschlägt. Es empfiehlt sich der Einsatz der Funktion `mysqli_connect()` in Kombination mit dem Befehl `or die("Meldung")`.

Dieser `or`-Zweig wird dann ausgeführt, wenn die Datenbankverbindung nicht aufgebaut werden konnte.

```
mysqli_connect() or die("Meldung");
```

Die Ausführung des Skripts wird mit der Funktion `die()` nach der Ausgabe des Parameters `Meldung` abgebrochen.

Beispiel:

```
mysqli_connect("localhost", "php-user", "DS5JiWHLqrYsPsJS")
or die("Verbindung konnte nicht hergestellt werden.");
```

Alternativ können Sie den Rückgabewert auf `FALSE` überprüfen, um die Seite nicht abrupt abzubrechen und eine nutzerfreundliche Meldung anzuzeigen, die Sie entsprechend in das Webseitenlayout implementieren. In dem Fall dürfen nachfolgende SQL-Zugriffe nicht ausgeführt werden, sondern sollten über `if`-Abfragen übersprungen werden.

Syntax der Funktion `mysqli_close()`

- ✓ Diese Funktion beendet eine Verbindung mit dem Datenbankserver. **`mysqli_close(Verbindungsvariable);`**
Die Angabe der Verbindungsvariable ist zwingend notwendig. Ohne diesen Parameter gibt PHP einen *Fatal error* aus. Die Datenbankverbindung wird dann nicht geschlossen, vor allem bricht aber das PHP-Skript ab.
- ✓ PHP schließt, nachdem ein PHP-Skript vollständig ausgeführt wurde, automatisch die Verbindung zum Datenbankserver. Sie sollten dennoch eine Verbindung immer selbst schließen, um genutzte Ressourcen sicher – und möglichst frühzeitig – freizugeben. Dies gehört unter anderem zum guten Stil in der PHP-Programmierung.

Beispiel: `db_connect.php`

Es wird eine Verbindung zum lokalen MySQL-Server hergestellt und wieder geschlossen. Zur Demonstration werden entsprechende Meldungen am Bildschirm ausgegeben.

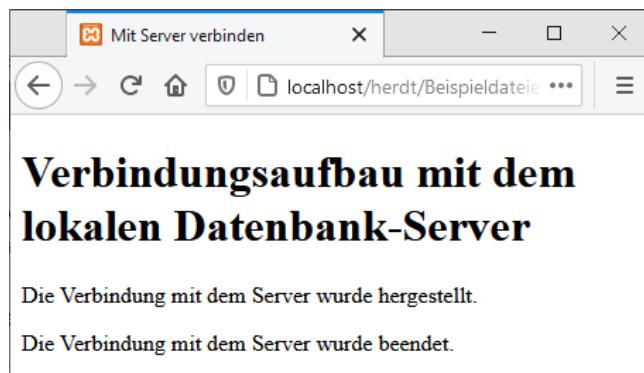
```
<?php
①    $server = "localhost";
      $user = "php-user";
      $pass = " DS5JiWHLqrySpJS";

      // Verbindungsaufnahme mit dem MySQL-Server
      $verbindung = mysqli_connect($server, $user, $pass)
      ②          or die("Verbindung konnte nicht hergestellt werden.");

      echo "<p>Die Verbindung mit dem Server wurde hergestellt.</p>";
      ③      $return = mysqli_close($verbindung);
      if ($return) {
          echo "<p>Die Verbindung mit dem Server wurde beendet. </p>";
      } else {
          echo "<p>Die Verbindung mit dem Server konnte nicht geschlossen
                  werden. </p>";
      }
?>
```

- ① Zu Beginn des Skripts werden die Anmeldevariablen für einen Zugriff auf den MySQL-Server hinterlegt. Der Server ist auf dem lokalen Rechner installiert (`localhost`). Den Variablen `$user` für den Benutzer und `$pass` für das Passwort werden die Werte des PHP-Benutzers zugewiesen.
- ② Über die Funktion `mysqli_connect()` und mit der Angabe der Anmeldevariablen wird eine Verbindung zum MySQL-Server hergestellt. Das zurückgegebene Objekt wird in der Variable `$verbindung` gespeichert.
- ③ Sind die Anmeldevariablen nicht korrekt bzw. ist der MySQL-Server nicht gestartet, kann keine Verbindung hergestellt werden. In diesem Fall erhält die Variable `$verbindung` den Rückgabewert `FALSE`, der den mit `or` eingeleiteten Zweig auslöst. Dort wird über die Funktion `die()` eine Fehlermeldung ausgegeben und das aktuelle Skript beendet.

- ④ War die Verbindung erfolgreich, wird am Ende des Skripts die aktuelle Verbindung zum Server mit der Funktion `mysqli_close()` getrennt. Als Parameter wird die Variable `$verbindung` übergeben, welche das Verbindungsobjekt enthält.



Alle Beispieldateien arbeiten mit dem Nutzer `$user = "php-user"` und dem Passwort `$pass = "DS5JiWHLqrYsPsJS"`. Zum schnellen Einstieg und falls Sie keinen Nutzer einrichten möchten, können Sie auch den Nutzer `root` ohne Passwort verwenden. Eine Empfehlung ist dieser Hinweis jedoch nicht!

Die gewünschte Datenbank auswählen

Nach einer erfolgreichen Verbindung zum MySQL-Server bestimmen Sie mit der Funktion `mysqli_select_db()`, welche Datenbank genutzt werden soll.

<code>mysqli_select_db();</code>	Auswahl der Datenbank
----------------------------------	-----------------------

Syntax der Funktion `mysqli_select_db()`

<code>mysqli_select_db(Verbindungskennung, Datenbankname);</code>

- ✓ Das von `mysqli_connect()` zurückgelieferte Objekt wird als erster Parameter angegeben.
- ✓ Als zweiten Parameter übergeben Sie den Namen der zu nutzenden Datenbank `Datenbankname`.
- ✓ Der Rückgabewert ist bei Erfolg `TRUE`, ansonsten `FALSE`.

Beispiel: `db_select.php`

Nachdem eine Verbindung zum lokalen MySQL-Server hergestellt wurde, muss nun die gewünschte Datenbank ausgewählt werden.

<pre><?php \$server = "localhost"; \$user = "php-user"; \$pass = "DS5JiWHLqrYsPsJS"; \$database = "obstladen"; // Verbindungsaufnahme mit dem MySQL-Server \$verbindung = mysqli_connect(\$server, \$user, \$pass) or die ("Verbindung konnte nicht hergestellt werden."); // Auswahl der gewünschten Datenbank</pre>
--

```

① mysqli_select_db($verbindung, $database) or die("Fehler
beim Zugriff auf die gewünschte Datenbank");

② echo "<p><em>Die Datenbank <strong>$database</strong> wurde
ausgewählt.</em></p>";
?>

```

- ① Die gezeigten Skriptzeilen werden in die Beispieldatei *db_select.php* integriert. Nach der Verbindung mit dem Datenbankserver wird über die Funktion `mysqli_select_db()` die als zweiter Parameter angegebene Datenbank *obstladen* gespeichert in der Variablen `$database` ausgewählt. Wichtig ist, dass als erster Parameter die Objektvariable für die Verbindungs-kennung angegeben ist. Schlägt die Auswahl der Datenbank fehl, wird das Skript nach der Ausgabe einer Meldung beendet.
- ② Es erfolgt eine Ausgabe, dass die Datenbank ausgewählt wurde. Diese Zeile des Skripts wird nur erreicht, wenn die Auswahl der Datenbank erfolgreich war.



Verbindung mit einem Datenbankserver herstellen und eine Datenbank auswählen

Beispiel: *db_connect_select.php*

Die beiden Schritte, Verbindungsaufbau zum Datenbankserver und Auswahl der Datenbank können in einem Schritt umgesetzt werden:

```

<?php
$server = "localhost";
$user = "php-user";
$pass = "DS5JiWHLqrYsPsJS";
$database = "obstladen";

// Verbindungsaufnahme mit dem MySQL-Server und Datenbankauswahl
$verbindung = mysqli_connect($server, $user, $pass, $database)
    or die("Verbindung konnte nicht hergestellt werden.");

② echo "<p>Die Verbindung mit dem Server wurde hergestellt, die
Datenbank <strong>$database</strong> wurde ausgewählt.</p>";
?>

```

- ① Beim Verbindungsaufbau über `mysqli_connect()` wird der vierte Parameter für die Auswahl der Datenbank angegeben. Nach dem Herstellen der Verbindung wird die angegebene Datenbank ausgewählt. Eine separate Auswahl der Datenbank erübrigts sich damit. Falls die Verbindung nicht hergestellt oder die Datenbank nicht ausgewählt werden kann, liefert `mysqli_connect()` `FALSE` zurück, das PHP-Skript wird in dem Fall mit dem `die()`-Befehl und der Ausgabe einer Fehlermeldung beendet.

- ② Nur wenn der Verbindungsaufbau und die Auswahl der Datenbank erfolgreich waren, wird diese Zeile im PHP-Code erreicht – es wird eine Erfolgsmeldung angezeigt.



12.7 MySQL-Abfragen

Abfrage senden

Zur Abfrage der Tabellendaten einer bestimmten Datenbank sind die nachfolgenden Befehle notwendig.

<code>mysqli_query();</code>	Senden einer MySQL-Anfrage zur aktiven Datenbankverbindung
------------------------------	--

Syntax der Funktion `mysqli_query()`

<code>mysqli_query(Verbindungskennung, SQL-Abfrage);</code>

- ✓ Eine Anfrage an MySQL realisieren Sie über die Funktion `mysqli_query()`.
- ✓ Der erste Parameter stellt die Verbindungskennung dar, als zweiten geben Sie ein SQL-Statement, also eine beliebige Datenbankabfrage an.
- ✓ Als Rückgabewert erhalten Sie entweder ein Objekt (bei SELECT-Statements), ein TRUE (z. B. bei INSERT-Statements) oder ein FALSE im Fehlerfall. Das zurückgelieferte Objekt bei einem SELECT-Statement liefert die Anzahl der Felder sowie die Anzahl der Zeilen, die ermittelt worden sind, die eigentlichen Daten sind in diesem Objekt **nicht** enthalten. Die Daten selbst werden danach über weitere `mysqli`-Funktionen ermittelt (weiter unten).

Beispiel: *mysqli_query.php*

In diesem Beispiel wird eine Verbindung zur Datenbank aufgenommen und eine SQL-Anweisung ausgeführt. Zur Veranschaulichung wird jede Aktion als Meldung im Browser ausgegeben.

```
<?php
    $server = "localhost";
    $user = "php-user";
    $pass = "DS5JiWHLqrYsPsJS";
    $database = "obstladen";

①    $verbindung = mysqli_connect($server, $user, $pass, $database)
        or die("Verbindung konnte nicht hergestellt werden.");

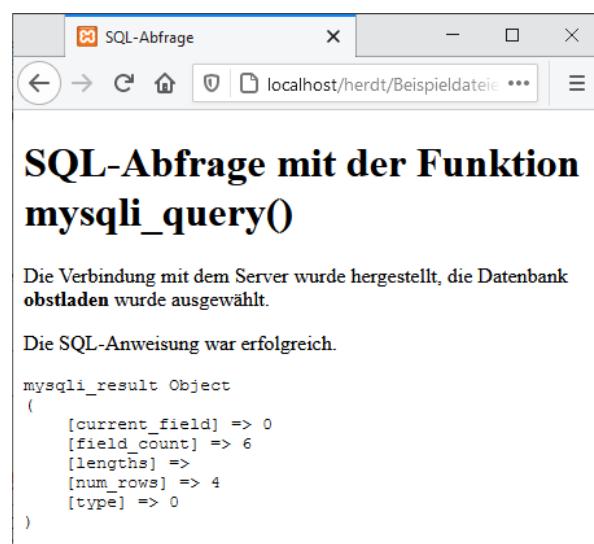
    echo "<p>Die Verbindung mit dem Server wurde hergestellt, die
        Datenbank <strong>$database</strong> wurde ausgewählt.</p>";

②    $sql = "SELECT * FROM bestellung";
③    if ($result = mysqli_query($verbindung, $sql)) {
        echo "<p>Die SQL-Anweisung war erfolgreich.</p>";
        echo "<pre>";
        print_r($result);
        echo "</pre>";
    } else {
        echo "<p>Die SQL-Anweisung ist fehlgeschlagen.</p>";
    }
④    $return = mysqli_close($verbindung);
    if (!$return) {
        echo "<p>Die Verbindung mit dem Server konnte
            nicht geschlossen werden.</p>";
    }
}
?>
```

- ① Es wird versucht, eine Verbindung zum SQL-Server herzustellen und mit dem Aufruf gleichzeitig die Datenbank `obstladen` (über die Variable `$database`) ausgewählt. Kommt keine Verbindung zustande, wird das Skript mit einer Fehlermeldung abgebrochen.

- ② Es folgt die Formulierung einer SQL-Abfrage. Der SQL-Befehl kann in Anführungszeichen direkt in der Funktion `mysqli_query()` notiert werden. Eleganter und übersichtlicher ist es hingegen, den SQL-Befehl in einer Variablen zu speichern. So haben Sie die Möglichkeit, den Befehl über PHP dynamisch zusammenzusetzen.

Im Beispiel werden mit dem `*` alle Felder der Tabelle `bestellung` ausgewählt werden. Beachten Sie, dass die Abfrage in Zeile ② noch nicht ausgeführt, sondern lediglich in der Variablen `$sql` gespeichert wird.



The screenshot shows a browser window titled "SQL-Abfrage" with the URL "localhost/herdt/Beispieldatei". The page content is:

SQL-Abfrage mit der Funktion mysqli_query()

Die Verbindung mit dem Server wurde hergestellt, die Datenbank **obstladen** wurde ausgewählt.

Die SQL-Anweisung war erfolgreich.

```
mysqli_result Object
(
    [current_field] => 0
    [field_count] => 6
    [lengths] =>
    [num_rows] => 4
    [type] => 0
)
```

- ③ Danach wird die Funktion `mysqli_query()` mit den Parametern `$verbindung` und `$sql` aufgerufen und in einem Schritt das zurückgelieferte Objekt in der Variablen `$result` gespeichert. Die Funktion gibt im Erfolgsfall ein Objekt zurück, ansonsten FALSE. Je nachdem, ob die Anweisung erfolgreich ausgeführt werden konnte, wird eine entsprechende Meldung angezeigt. Zusätzlich wird das Objekt `$return` auf dem Bildschirm ausgegeben. Es enthält u. a. Informationen über die Anzahl der Felder und Zeilen.
- ④ Zum Schluss wird die Verbindung zum MySQL-Server wieder geschlossen, eine Meldung wird nur noch im Fehlerfall angezeigt.

Fehlertext von SQL-Operationen anzeigen

<code>mysqli_error();</code>	Ausgabe eines SQL-Fehlers
------------------------------	---------------------------

Syntax der Funktion `mysqli_error()`

<code>mysqli_error(Verbindungskennung);</code>
--

Um Informationen zum aufgetretenen Fehler – sei es bei der Verbindungsaufnahme, bei Abfragen oder anderen Operationen – zu erhalten, verwenden Sie die Funktion `mysqli_error()`. Diese leitet den Fehlercode der zuletzt ausgeführten MySQL-Funktion vom MySQL-Server an das PHP-Skript weiter. Diese Informationen können dabei helfen, Fehler schnell zu finden und zu beheben.

Beispiel: Ausschnitt aus `mysqli_error.php`

In diesem Ausschnitt soll eine Fehlerrückgabe ausgelöst werden. Hierfür wurde bewusst ein syntaktischer Fehler ① in eine SQL-Abfrage eingefügt. Es werden keine Feldnamen bzw. kein * für die Rückgabe aller Feldnamen, die für eine SELECT-Anweisung zwingend erforderlich sind, angegeben, die aus der Tabelle ausgelesen werden sollen.

```
①
$sql = "SELECT FROM bestellung";
// FEHLER: hier wurde nicht angegeben,
// welche Felder der Tabelle ausgewählt werden sollen

if (mysqli_query($verbindung, $sql)) {
    echo "<p>Die SQL-Anweisung war erfolgreich.</p>";
} else {
    echo "<p>SQL-Fehler! SQL meldet:<br>" .
        mysqli_error($verbindung) . "</p>";
}
```



*Ansicht der Beispieldatei „mysqli_error.php“. „You have an error... near...“
Sie erhalten einen Hinweis, an welcher Stelle des SQL-Befehls ein Fehler aufgetreten ist.*

12.8 Rückgabe aus MySQL-Abfrage auswerten

Datensätze einer MySQL-Tabelle mithilfe von PHP anzeigen

Nachdem die Verbindung zur MySQL-Datenbank `obstladen` hergestellt wurde, können Sie z. B. die Anzahl der Datensätze oder alle Datensätze der Tabelle `bestellung` anzeigen. Wenn Sie wissen möchten, wie viele Datensätze die SQL-Abfrage zurückliefert, verwenden Sie die Funktion `mysqli_num_rows()`.

Syntax der Funktion `mysqli_num_rows()`

- ✓ Diese Anweisung liefert Ihnen die Anzahl der Datensätze, die mit dem SQL-Befehl `SELECT` an das Skript zurückgeliefert werden.
- ✓ Der Parameter `Abfrageergebnis` ist das zurückgelieferte Objekt der aktuellen SQL-Abfrage, also dem Rückgabewert der Funktion `mysqli_query()`.

Beispiel

```
$ergebnis = mysqli_query("SELECT * FROM bestellung");
echo mysqli_num_rows($ergebnis);
```

Ausgabe der Datensätze mit der Funktion `mysqli_fetch_array()`

Über die Funktion `mysqli_fetch_array()` in Kombination mit einer `while`-Schleife können Sie die einzelnen Zeilen der Datenbankabfrage auslesen. Dazu notieren Sie im Kopf der `while`-Schleife die Funktion `mysqli_fetch_array()` mit dem Objekt, das Sie von der Funktion `mysqli_query()` als Rückgabewert erhalten haben, als Parameter. Bei jedem Schleifendurchlauf wird eine Ergebniszelle als Array gespeichert. Die `while`-Schleife wird so lange ausgeführt, bis alle Einträge der Datenbankabfrage durchlaufen sind.

Syntax der Funktion `mysqli_fetch_array()`

```
mysqli_fetch_array(Abfrageergebnis [, Ergebnistyp]);
```

- ✓ Der anzugebende Parameter `Abfrageergebnis` stellt die Ergebniskennung der aktuellen SQL-Abfrage dar.
- ✓ Mit der optionalen Angabe `Ergebnistyp` legen Sie fest, mit welchem Array-Typ die Daten des Ergebnisses zurückgegeben werden. Die möglichen Parameter sind `MYSQLI_ASSOC`, `MYSQLI_NUM` oder `MYSQLI_BOTH`. Hiermit bestimmen Sie, wie Sie die Elemente des Arrays ansprechen möchten.
 - ✓ Bei `MYSQLI_ASSOC` verwenden Sie den Feldnamen der Tabelle als Schlüssel im Array (z. B. `zeile["sorte"]`, assoziatives Array).
 - ✓ Bei `MYSQLI_NUM` arbeiten Sie über die Angabe des numerischen Indexes (z. B. `zeile[3]`, numerisch indiziertes Array).
 - ✓ Wenn Sie die Option `MYSQLI_BOTH` verwenden, sind beide Varianten zum Ansprechen eines Elements möglich. (Das Array enthält dann sowohl numerische als auch assoziative Schlüssel, d. h., jeder Wert ist im Array doppelt vorhanden und über zwei verschiedene Schlüssel ansprechbar.)
- ✓ Wenn Sie keinen `Ergebnistyp` angeben, wird standardmäßig die Option `MYSQLI_BOTH` verwendet.

Beispiel: `mysqli_fetch_array.php` (Auszug)

In diesem Beispiel werden Datensätze aus einer Tabelle ausgelesen und angezeigt.

```
$sql = "SELECT * FROM bestellung";
$abfrage = mysqli_query($verbindung, $sql);
if (!$abfrage) {
    echo "<p>Die SQL-Anweisung ist fehlgeschlagen.</p>";
}
$anzahl = mysqli_num_rows($abfrage);
echo "<p>In der Tabelle befinden sich $anzahl Datensätze:</p>";
echo "<ul>";
② while ($zeile = mysqli_fetch_array($abfrage)) {
    echo "<li> " . $zeile["id"] . ":" .
        . $zeile["vorname"] . " "
        . $zeile["nachname"] . ", "
        . $zeile["ort"] . ", "
        . $zeile["menge"] . " kg "
        . $zeile["sorte"] . ".";
}
echo "</ul>";
```

Auszug aus der Datei „`mysqli_fetch_array.php`“. Aufbau der Datenbankverbindung und das Schließen der Verbindung werden hier nicht dargestellt.

- ① Die Anzahl der Datensätze ermitteln Sie mit der Funktion `mysqli_num_rows()`.

- ② Mithilfe der Funktion `mysqli_fetch_array()` ermitteln Sie ein Array, das dem aktuellen Datensatz (also einer Zeile des Ergebnisses aus der Tabelle) entspricht. Gleichzeitig wird durch diese Funktion der sogenannte Datensatzzeiger auf den nächsten Datensatz des Ergebnisses gesetzt. Durch die `while`-Schleife werden somit nacheinander alle Datensätze der Tabelle an ein Array (`$zeile`) übergeben. Die Schlüssel dieses Arrays sind die Feldnamen der MySQL-Tabelle. Wenn keine weiteren Datensätze vorliegen, wird `FALSE` zurückgegeben und damit die Schleife beendet.

Daten aus einer Tabelle auslesen und darstellen

In der Tabelle befinden sich 4 Datensätze:

- 1: Arndt Hoffmann, Stuttgart, 15 kg Elstar.
- 2: Corinna Delphi, Hamburg, 3 kg Janagold.
- 3: Petra Meyer, Wien, 5 kg Gala.
- 4: Peter Schmidt, Berlin, 25 kg Elstar.

Anzeige der Beispieldatei „`mysqli_fetch_array.php`“

12.9 Formulardaten in einer MySQL-Datenbank speichern

Die von Ihnen erstellten Tabellen der Datenbank können Sie von den Besuchern Ihrer Webseite automatisch füllen lassen. Dazu füllt der Benutzer ein Formular aus. Über PHP werden dann die Formulardaten in die MySQL-Datenbanktabelle eingetragen.

Beispiel: `bestellformular_db.html`

```

<h1>Apfelkauf im Obstladen</h1>
<p>Bitte geben Sie folgende Daten für Ihre Bestellung ein:</p>
<form action="bestellung_db.php" method="post">
  ① <p>Vorname: <input type="text" name="vorname"></p>
  ② <p>Nachname: <input type="text" name="nachname"></p>
  <p>Wohnort: <input type="text" name="ort"></p>
  <p>Menge (in kg): <input type="text" size="5"
                           name="menge"></p>
  <p>Apfelsorte:
    <input type="radio" name="sorte" value="Jonagold">Jonagold
    <input type="radio" name="sorte" value="Gala">Gala
    <input type="radio" name="sorte" value="Elstar">Elstar
  </p>
  <p><input type="submit" value="Abschicken"></p>
</form>

```

- ① Die Daten des Formulars werden an das PHP-Skript *bestellung_db.php* gesendet.
- ② Die vom Besucher einzugebenden Daten sind: Vorname, Nachname, Wohnort, Menge sowie die Auswahl einer Sorte über Radiobuttons.

Beispiel: *bestellung_db.php* (Auszug)

Im Folgenden wird ein PHP-Skript erstellt, das die Daten des obigen HTML-Formulars auswertet. Die eingetragenen Daten sollen in die Tabelle *bestellung* der Datenbank *obstladen* gespeichert werden.

The screenshot shows a browser window with the title "Daten in MySQL-Datenbank". The URL bar shows "localhost/herdt/Beispieldateie...". The main content is a form titled "Apfelkauf im Obstladen" asking for the following information:

- Vorname: [Input field]
- Nachname: [Input field]
- Wohnort: [Input field]
- Menge (in kg): [Input field]
- Apfelsorte: Jonagold Gala Elstar
-

```

① $vorname = $_POST["vorname"];
$Nachname = $_POST["Nachname"];
$ort = $_POST["ort"];
$sorte = $_POST["sorte"];
$menge = $_POST["menge"];

② $verbindung = mysqli_connect($server, $user, $pass, $database)
    or die("Verbindung konnte nicht hergestellt werden.");

③ $sql = "INSERT INTO bestellung(vorname, nachname, ort, sorte,
    menge)";
$sql .= " VALUES ('$vorname', '$Nachname', '$ort', '$sorte',
    $menge)";

④ $abfrage = mysqli_query($verbindung, $sql);
if ($abfrage) {
    echo "<p>Vielen Dank, Ihre Bestellung wurde gespeichert.</p>";
} else {
    echo "<p>Die SQL-Anweisung ist fehlgeschlagen.</p>";
}

```

- ① Die Variablen `$vorname`, `$Nachname`, `$ort`, `$sorte` und `$menge` werden mit den Daten aus dem Formular gefüllt.
- ② Die Verbindung zum Datenbankserver wird hergestellt und die entsprechende Datenbank ausgewählt.
- ③ Die SQL-Anweisung zum Eintrag der übermittelten Werte in die Tabelle *bestellung* wird definiert. Die Felder müssen nach dem Tabellennamen angegeben werden, da in der Tabelle sechs Felder existieren, aber nur fünf Werte übergeben werden. Das Feld `id` ist vom Typ `auto_increment` und wird automatisch hochgezählt.
- ④ Die Abfrage wird ausgeführt. Anschließend wird eine Fehler- oder Erfolgsmeldung ausgegeben.



Wissenstest: Arrays bis Sessions

12.10 Übung

Einen Newsletter abonnieren

Level		Zeit	ca.30 min
Übungsinhalte	<ul style="list-style-type: none"> ✓ HTML-Formulare ✓ Datenbankverbindungen mit der PHP-Erweiterung <i>mysqli</i> ✓ Daten in Datenbanken schreiben 		
Übungsdatei	--		
Ergebnisdateien	<i>newsletter.html</i> , <i>newsletter.php</i> ,		
MySQL-Dump	<i>homepage-newsletter.sql</i>		

1. Auf vielen Webseiten können Benutzer ihren Namen und E-Mail-Adresse hinterlassen, um einen Newsletter zu erhalten. Erstellen Sie eine Webseite, auf der die Besucher der Webseite Ihren Newsletter bestellen können.
2. Erstellen Sie über phpMyAdmin eine neue MySQL-Datenbank mit dem Namen *homepage*.
3. Erzeugen Sie in dieser Datenbank die Tabelle *newsletter*. Die Tabelle soll folgende zwei Felder enthalten:

Feldname	Feldtyp	Null	Unique
<i>UserName</i>	<i>varchar(50)</i>	<i>null</i>	
<i>UserMail</i>	<i>varchar(50)</i>	<i>not null</i>	x

Alternativ können Sie die Import-Datei *homepage-newsletter.sql* verwenden, um Datenbank und Tabelle zu erstellen.

4. Die Anmeldung für den Newsletter geschieht über ein HTML-Formular, das Sie – wie nebenstehend angezeigt – aufbauen können.
5. Die Daten des Formulars sollen an das PHP-Skript *newsletter.php* gesendet werden.
6. Erstellen Sie das PHP-Skript *newsletter.php*, das die Daten des HTML-Formulars auswertet. Aufgabe des PHP-Skripts ist es, den Namen und die E-Mail-Adresse des Besuchers in die Tabelle *newsletter* der Datenbank *homepage* einzutragen.
7. Überprüfen Sie, ob der Benutzer eine E-Mail-Adresse angegeben hat. Wenn das Feld keinen Eintrag hat, soll der Benutzer eine Meldung und einen Link erhalten, mit dem er zum Formular zurück navigieren kann.

The screenshot shows a browser window with the title "Übung Kapitel 12". The URL bar shows "localhost/herdt/Ergebnisdatei". The main content is a form titled "Newsletter abonnieren". It contains two input fields: "Ihr Name:" and "Ihre E-Mail-Adresse:", both with placeholder text. Below the fields is a "Absenden" button. The entire form is enclosed in a light gray box.

Anzeige der Formulardatei „*newsletter.html*“

8. Der Versuch, eine E-Mail-Adresse mehrfach einzutragen, schlägt fehl, da das E-Mail-Feld als `unique` deklariert wurde. Lassen Sie für diesen Fall mit Hilfe der Funktion `mysqli_error()` eine entsprechende Fehlermeldung ausgeben.
9. Nachdem die Daten an die Tabelle `newsletter` übergeben wurden, soll eine Meldung erscheinen, z. B. *Die E-Mail-Adresse maxi@puppetmail.de wurde gespeichert.*



Bestätigungsseite nach erfolgreichem Eintrag

A

Installation und Konfiguration der Software

A.1 Installation und Konfiguration von XAMPP

Was ist XAMPP?

XAMPP ist eine kostenlose Apache-Distribution, welche als Software-Paket mit einer einfachen Installationsroutine auf einem „x-beliebigen“ internetfähigen Betriebssystem einen Apache-Webserver einschließlich MariaDB, PHP und Perl auf dem eigenen Rechner installiert werden kann. Vor allem für unerfahrene Benutzer ist es sehr zu empfehlen, sich nicht durch die teilweise komplexe und schwierige Installation und Konfiguration der Einzelkomponenten zu kämpfen. Aber auch fortgeschrittene Programmierer schätzen durchaus die fast automatische Installation innerhalb weniger Minuten. Herausgeber der XAMPP-Pakets sind die Apache Friends.

Kostenloser Download des bereits konfigurierten Webservers:

<https://www.apachefriends.org/de/index.html>

Die für das Buch notwendige XAMPP-Version 8.0.1 mit der PHP-8.0.1-Version steht aktuell für drei Betriebssysteme zur Verfügung:

- ✓ für Windows
- ✓ für Linux
- ✓ für macOS

Alternative für Mac-Nutzer

Download MAMP: <https://www.mamp.info/de/>

Alternativ zu XAMPP können Sie auch MAMP (**M**y **A**pache – **M**y**SQ**L – **P**HP) 6.3 verwenden, welches von der MAMP GmbH angeboten wird. Dieser vorkonfigurierte und einfach zu installierende Webserver stand früher nur Mac-Benutzern zur Verfügung. In der Mac-Version 6.3 kann zwischen PHP 7.4.12 oder PHP 8.0.0 gewählt werden, aber auch ältere Versionen stehen in der PRO-Version zur Verfügung. Seit Anfang 2014 ist MAMP auch für Windows verfügbar. Die Windows-Version 4.2 wird mit PHP 7.4.1 ausgeliefert. MAMP steht zusätzlich in einer kostenpflichtigen Version (MAMP PRO) zur Verfügung.

Alle PHP-Skripte und Beispiele in diesem Buch sind auf dem XAMPP-Webserver umgesetzt und getestet worden. Es werden alle Komponenten (Apache, PHP und MariaDB) bis auf die Sprache Perl verwendet. Die Beispiele sind in einer Windows-10-Umgebung entwickelt worden, daher wird auch für die Installation der Einsatz von XAMPP und Konfiguration Windows beispielhaft herangezogen.

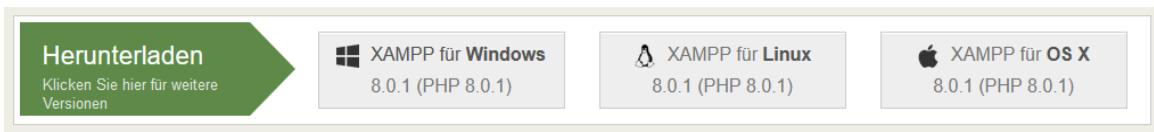
Bei der Erstellung dieses Buchs wurde mit der XAMPP-Version 8.0.1 vom 5. Januar 2021 gearbeitet. Das Software-Paket umfasst die für das Buch relevanten Komponenten in folgenden Versionen:

- ✓ PHP, Version 8.0.1
- ✓ Apache Webserver, Version 2.4.46
- ✓ MariaDB, Version 10.4.17
- ✓ phpMyAdmin, Version 5.0.4

Installiert wurde das in der Abbildung gezeigte XAMPP 8.0.1 unter dem Betriebssystem Windows 10 Home und den Standardeinstellungen für alle Programme.

Auch wenn XAMPP seit Oktober 2015 statt MySQL die Datenbank MariaDB verwendet, hat sich die Begrifflichkeit in die Entwicklersprache nicht etabliert. MariaDB ist eine Abspaltung von MySQL, diese beiden Datenbanken sind weitestgehend kompatibel miteinander. Außerdem tragen alle PHP-Funktionen, die auf diese Datenbanken zugreifen, „mysql“ im Funktionsnamen. Von daher sprechen so gut wie alle Entwickler von MySQL, auch wenn sie tatsächlich mit einer MariaDB arbeiten. Ebenso halten wir es in diesem Buch.

Download von XAMPP



Auf der Startseite von <https://www.apachefriends.org> finden Sie Download-Links der Installer-Versionen für die unterschiedlichen Betriebssysteme.

Ältere XAMPP-Versionen stehen auf der Download-Seite <https://www.apachefriends.org/download.html> zur Verfügung. Dort finden Sie Versionen sowohl für Windows, Linux als auch für macOS.

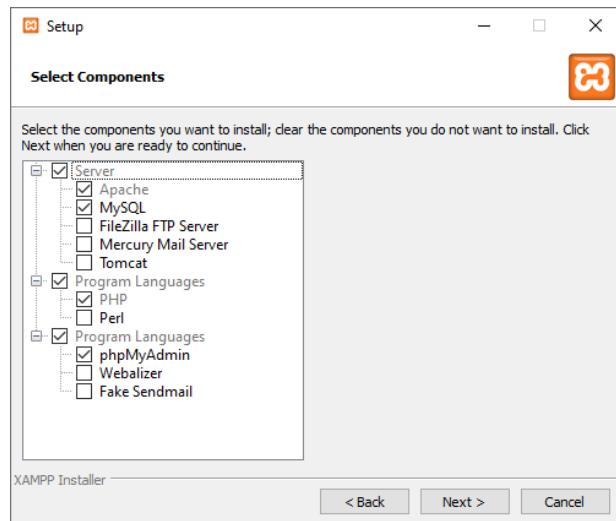
- ▶ Laden Sie von der angegebenen Internetadresse das aktuelle XAMPP-Paket herunter.

Installation von XAMPP (Windows)

Der XAMPP-Installationsassistent ist so voreingestellt, dass Sie für eine Standardinstallation keine Einstellungen verändern müssen.

- ▶ Um XAMPP 8.0.1 zu installieren, folgen Sie den Schritten des Installationsassistenten.
- ▶ Falls Sie eine Firewall oder einen Virenschanner installiert haben oder die Windows Benutzerkontensteuerung (User Account Control, abgekürzt UAC) aktiviert haben, erscheinen zu Beginn der Installation Warnhinweise. Bestätigen Sie diese mit *OK*. Falls Sie als Mac-Nutzer ein Passwort für den root-Nutzer verwenden, kann die Installation nur nach Eingabe des Passwortes gestartet werden.
- ▶ Klicken Sie in allen Dialogen die Schaltfläche *Next*, um die Installation durchzuführen.

- ▶ Wählen Sie die Module von XAMPP aus, die Sie installieren möchten (Windows). Wählen Sie mindestens *PHP*, *MySQL* und *phpMyAdmin* aus. Die anderen Module sind optional und für das Arbeiten mit diesem Buch nicht notwendig. Unter macOS ist die Auswahl reduziert, hier wählen Sie mindestens die *XAMPP Core Files* aus.
- ▶ Falls Sie XAMPP nicht in dem voreingestellten Verzeichnis *C:\xampp* installieren möchten, wählen Sie ein anderes Verzeichnis aus (gilt für Windows). Eine Auswahl des Installationspfads unter macOS ist nicht möglich.
- ▶ Nach Abschluss der Installation ist die Checkbox *Do you want to start the Control Panel now?* ausgewählt. Lassen Sie diese ausgewählt, klicken Sie *Finish*, um die Installation abzuschließen und gleichzeitig das Control Panel zu öffnen (vgl. nächsten Abschnitt).
- ▶ Mac-Nutzer erhalten einen anderen Dialog mit der Option *Launch XAMPP*. Über die Schaltfläche *Finish* schließen Sie die Installation ab und starten den Apache Webserver.



Wenn Sie bei der Installation die Standardeinstellungen beibehalten, werden folgende Einstellungen vorgenommen:

- ✓ Die Installation erfolgt unter Windows standardmäßig in das Verzeichnis *C:\xampp*.
- ✓ Unter macOS erfolgt die Installation im Verzeichnis */Applications/XAMPP*.

Sollten Sie mit einer anderen XAMPP-Version arbeiten, können die Schritte während der Installation, die Einstellungsmöglichkeiten und die Bezeichnungen der Schaltflächen von dieser Beschreibung abweichen. Bei allen Installationsassistenten in der Vergangenheit waren die Standardeinstellungen jedoch stets sinnvoll voreingestellt, so dass auch bei anderen Versionen keine speziellen Einstellungen vorgenommen werden müssen.

Die Installation des MAMP-Webservers ist vergleichbar. Folgen Sie den Anweisungen während der Installation oder lesen Sie die Dokumentation, welche unter <https://documentation.mamp.info> zur Verfügung steht.

Probleme bei der Installation von XAMPP

Durch die einfache und automatische Installation von XAMPP sind Probleme bei der Installation recht selten. Die häufigsten Probleme bereiten folgende Bereiche:

- ✓ **Berechtigungen:** Der Benutzer, der XAMPP installieren möchte, verfügt nicht über ausreichende Berechtigung für diese Softwareinstallation. Installieren Sie gegebenenfalls XAMPP als Administrator.
- ✓ **Unter Windows kann die Benutzerkontensteuerung (User Account Control, UAC) die Installation verhindern.** Eine Lösung für dieses Problem ist, die Benutzerkontensteuerung zu deaktivieren. Eine andere ist, den Standardinstallationspfad *C:\xampp* zu verändern und die XAMPP-Installation in einem eigenen Benutzerverzeichnis zu installieren, beispielweise in den eigenen Dokumenten.

- ✓ **Belegter Port 80.** Standardmäßig wird der XAMPP-Webserver unter dem Port 80 installiert. Unter Windows 10 belegt mitunter der *WWW-Publishingdienst* den Port 80. Das Problem kann gelöst werden, indem Sie den Start-Typ dieses Dienstes von *Auto* auf *Manuell* umstellen (*Systemsteuerung - System und Sicherheit - Verwaltung - Dienste*). Ebenfalls belegt unter Umständen *Skype* den Port 80. Auch in diesem Fall wird nach der Installation von XAMPP der Start des Webservers verhindert. Hierzu deaktivieren Sie die Option *Skype beim Windows-Start ausführen*. Diese finden Sie in den Skype-Einstellungen unter *Aktionen - Optionen - Allgemeine Einstellungen*. Nachdem Sie diese Einstellungen vorgenommen und den Rechner neu gestartet haben, steht der Port 80 für XAMPP zur Verfügung.
- ✓ **MySQL ist bereits installiert:** Andere Software installiert mitunter einen eigenen MySQL-Server. Das kann bei der Installation von XAMPP zu Problemen zwischen den beiden vorhandenen Instanzen des Servers führen. In der Regel hilft die Deinstallation des ersten MySQL-Servers, damit XAMPP keinen bereits installierten MySQL-Server vorfindet.
- ✓ **Installation einer alten XAMPP-Version:** Das Passwort für den MySQL-SuperUser kann in einem Cookie gespeichert werden. Auch bei einer vollständigen Neuinstallation von XAMPP bleibt das Cookie im Browser erhalten, was eine Zugriffsverweigerung von phpMyAdmin auf die Datenbank zur Folge haben kann. Hier hilft das Löschen des Cookies oder der Wechsel der phpMyAdmin-Authentifikation von *cookie* auf *http*. Damit wird das Cookie nicht mehr abgefragt.

Weitere Hilfen zu Fragen und Problemen rund um die Installation und den Betrieb von XAMPP finden Sie im Internet unter den folgenden Adressen:

- ✓ Windows FAQ: https://www.apachefriends.org/faq_windows.html,
- ✓ Linux FAQ: https://www.apachefriends.org/faq_linux.html,
- ✓ macOS FAQ: https://www.apachefriends.org/faq_osx.html.

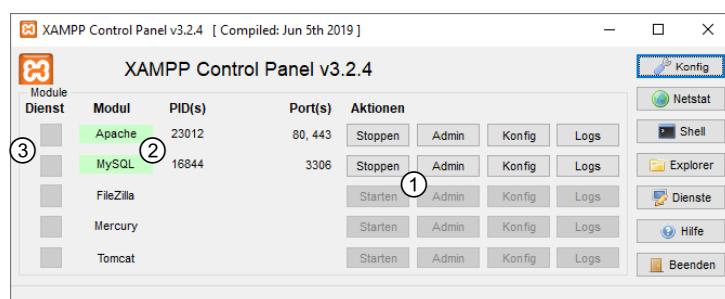
A.2 Mit XAMPP arbeiten

Das XAMPP Control Panel (Windows)

Das XAMPP Control Panel ist die Schaltzentrale zur Steuerung der einzelnen Komponenten. Standardmäßig wird beim Abschluss der Installation angeboten, das XAMPP Control Panel zu starten.

Verwenden Sie diese Applikation in erster Linie dazu, den Betrieb der einzelnen Komponenten zu steuern und zu überprüfen. Der Apache-Webserver (einschließlich dem PHP-Interpreter) als auch der MySQL-Server müssen gestartet sein, damit PHP-Skripte überhaupt geprägt werden und Datenbankzugriffe möglich sind.

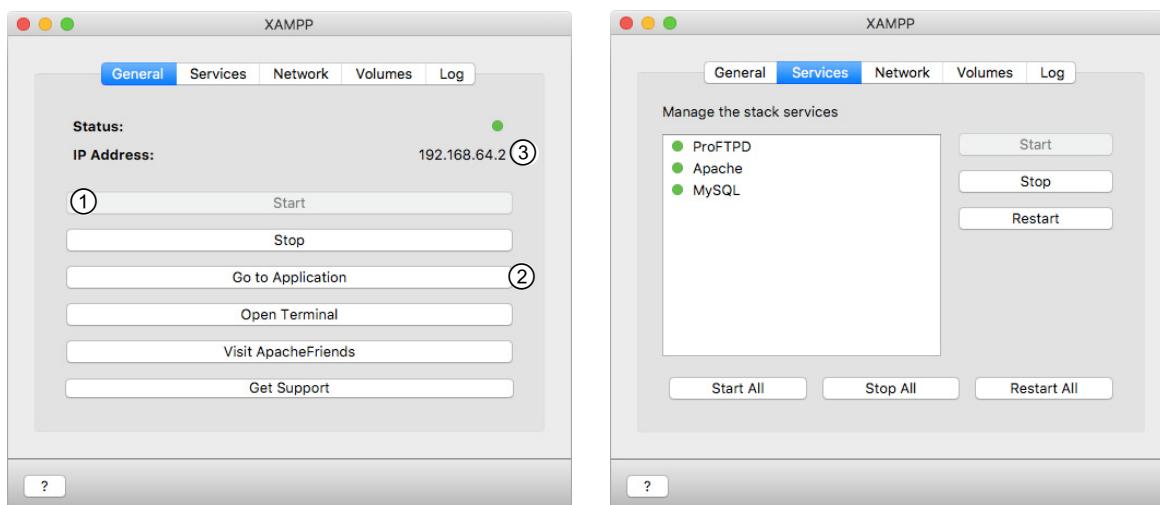
Welchen Status die Komponenten haben, erkennen Sie daran, ob die jeweilige Schaltfläche *Starten* oder *Stoppen* anzeigt ①. Über diese Schaltflächen starten und stoppen Sie die Komponenten. Zusätzlich erkennen Sie die aktivierte Komponente an der grünen Hinterlegung des Modulnamens ②.



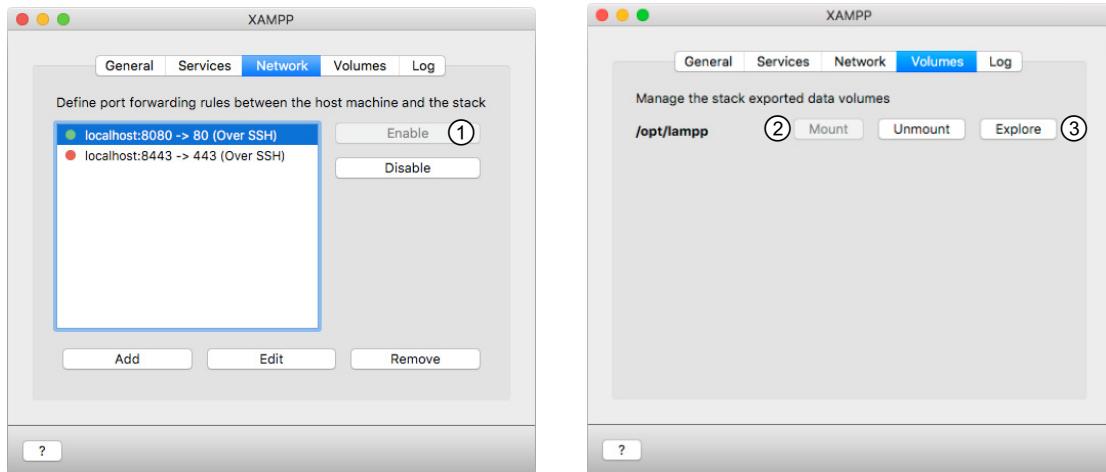
Sollen Apache-Webserver und MySQL-Server beim Hochfahren des Rechners automatisch gestartet werden, können Sie die Checkbox vor dem Modulnamen aktivieren ③. Dazu muss die jeweilige Komponente gestoppt werden, falls sie gerade aktiv ist. Mit der Auswahl der Checkbox wird die Komponente als Dienst installiert und startet dann automatisch beim Start des Computers. Sollten die Checkboxen nicht bedienbar sein, haben Sie keine Zugriffsrechte. In dem Fall melden Sie sich als Administrator an (Gehen Sie in das Verzeichnis C:\xampp, gehen Sie auf die Datei *xampp-control.exe* und wählen Sie im Kontextmenü die Option *Als Administrator ausführen*) und nehmen dann die gewünschte Einstellung vor. Dies muss nur einmal getan werden. Sind die Dienste einmal eingerichtet, starten die Server beim Windows-Start auch für normale Windows-Nutzer. Diese Einstellung ist sinnvoll, da Sie nicht jedes Mal die beiden Server manuell starten müssen.

XAMPP unter macOS

Laden Sie sich XAMPP für Mac herunter und installieren Sie das Paket unter *Programme* (Applications). Mit Doppelklick auf die XAMPP-Applikation starten Sie das Programm. Im ersten Schritt starten Sie den XAMPP-Server ① und damit den Webserver Apache sowie die Datenbank MySQL. Sobald das grüne Symbol erscheint, können Sie mit Klick auf *Go to Application* ② die Startseite von XAMPP öffnen. Diese wird unter der IP geöffnet, die Ihnen in dem Fenster angezeigt wird ③. Dort finden Sie unter der Seite *FAQs* eine Schritt-für-Schritt-Anleitung, wie Sie XAMPP auf dem Mac vollständig installieren.



Im zweiten Register *Services* können Sie bei Bedarf die einzelnen Dienste starten und stoppen. Auch hier wird durch ein grünes Symbol signalisiert, welche der Dienste laufen. Ein Neustart des Apache-Webservers ist immer dann notwendig, wenn Sie Veränderungen in der *php.ini* vorgenommen haben.



Damit Sie den PHP-Sever auch unter *localhost* nutzen können, müssen Sie den Port 80 „enablen“ ①. Sie können dann die PHP-Skripte über <http://localhost:8080> aufrufen. Gegebenenfalls müssen Sie im Register *Volumes* noch den Ordner */opt/lampp* „mounten“ ②, damit der Webserver die PHP-Skripte findet.

Das Verzeichnis */opt/lampp* verweist auf eine virtuelle Maschine, die auf dem Mac unter */Users/[Ihr Username]/.bitnami/stackman/machines/* liegt. Über die Schaltfläche *Explore* ③ gelangen Sie zu allen Konfigurationsdateien sowie zum Ordner *htdocs*, wo Sie Ihre PHP-Skripte ablegen.

! Bei einer Neuinstallation der XAMPP-Software müssen Sie sicherstellen, dass die virtuelle Maschine unter */Users/[Ihr Username]/.bitnami/stackman/machines* gelöscht wird. Ansonsten bleiben alte Daten und Konfigurationseinstellungen trotz Neuinstallation erhalten.

Webserver testen

Jeder Webserver im Internet ist über einen eindeutigen Domain-Namen zu erreichen, beispielsweise <https://herdt.com>. Genauso verhält es sich mit dem lokalen Webserver.

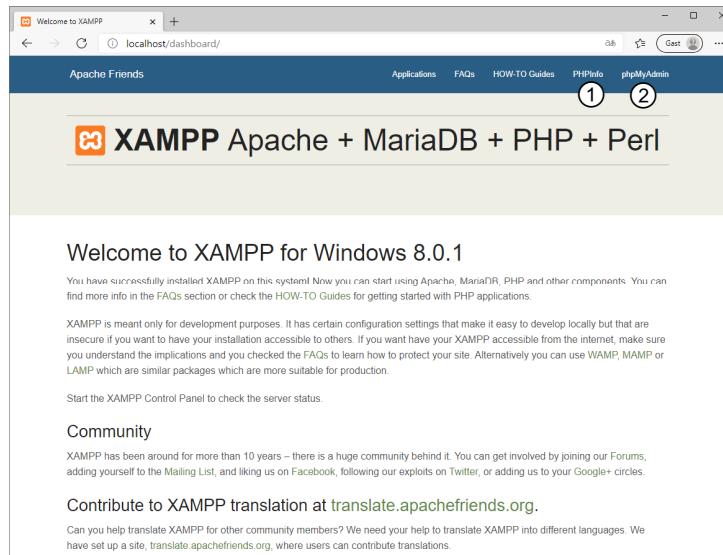
Sie rufen den lokalen Webserver über die Adresse **<http://localhost>** oder alternativ über die IP-Adresse **<http://127.0.0.1>** auf. Auf einem Mac müssen Sie zusätzlich den Port 8080 mit angeben.

- ▶ Starten Sie Ihren Browser.
- ▶ Geben Sie in der Adresszeile <http://127.0.0.1> oder <http://localhost> ein.
- ▶ Auf dem Mac geben Sie zusätzlich den Port an: <http://localhost:8080> bzw. <http://127.0.0.1:8080>.

Bei erfolgreicher Installation und gestartetem Webserver erscheint die Startseite des installierten XAMPP.

In der Navigation auf der Startseite von XAMPP sind die häufig genutzten Funktionen als Links angegeben. Sie können unter anderem ...

- ✓ die Konfiguration von PHP per *PHPInfo* abrufen ① oder
- ✓ *phpMyAdmin* zur Steuerung des lokalen MySQL-Servers aufrufen ②.



! Aus Gründen der Vereinfachung wird in diesem Buch nicht immer auf die Angabe des Ports 8080 hingewiesen.

Speicherort für PHP-Dateien

Der Ordner für alle Webdokumente, das sogenannte *root directory*, lautet bei einer Standardinstallation unter Windows *C:\xampp\htdocs*, unter macOS finden Sie den entsprechenden Ordner unter dem gemounteten Ordner *lampp/htdocs*. Alle Dokumente, die sich in diesem Ordner befinden, können Sie im Browser bei gestartetem Apache-Webserver unter der Adresse <http://localhost/<Dateiname>> bzw. <http://localhost:8080/<Dateiname>> auf dem Mac aufrufen.

Noch einfacher ist es, im Ordner *htdocs* einen Unterordner zu erstellen, z. B. *myphp*, der all Ihre PHP-Dateien wie die Beispielskripte dieses Buches enthält. Dann können Sie mit der Adresse <http://localhost/myphp> den Ordnerinhalt auflisten und müssen die gewünschte Datei zur Anzeige nur noch anklicken.

A.3 Installation und Konfiguration von Notepad++

Arbeiten mit einem Texteditor

Um ein PHP-Skript zu schreiben, wie Sie es beispielsweise von HTML kennen, benötigen Sie einen Texteditor. Ein Editor ist eine Software zur Bearbeitung von ASCII-Texten und kann daher zur Eingabe von PHP-Code verwendet werden. Texteditoren sind auf allen Computer-Plattformen verfügbar und besitzen unterschiedliche Funktionsausstattungen. Ein Editor unterscheidet sich von einer Textverarbeitung oder einem Layout-Programm dadurch, dass er keine – normalerweise unsichtbaren – Formatierungsanweisungen in den Text einfügt, um ihn in Absätze, Listen, Tabellen usw. zu gliedern.

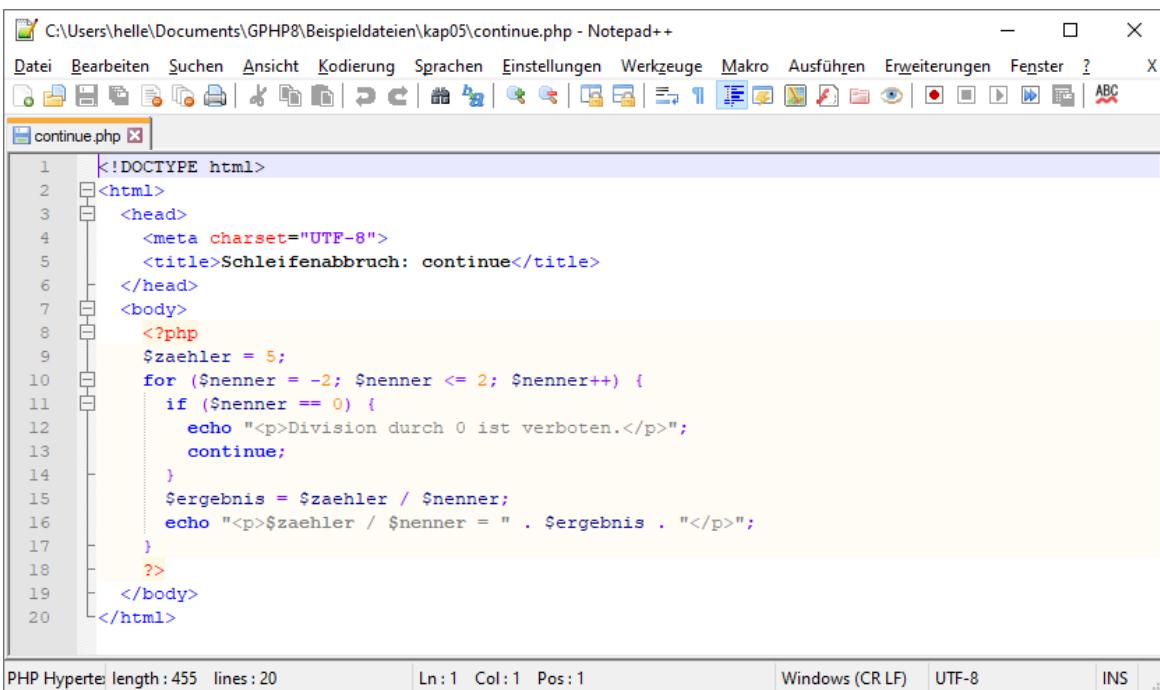
PHP-Editor einsetzen

Im Internet finden Sie auch spezielle PHP-Editoren, die eine Hervorhebung der Befehle und Hilfestellungen für verschiedene Sprachelemente bieten und Sie somit beim Erstellen Ihrer Webseiten visuell unterstützen.

Bei der Erstellung des vorliegenden Buchs wurde mit dem Texteditor **Notepad++**, Version 7.9.2, (<https://notepad-plus-plus.org/downloads/>) gearbeitet. Notepad++ ist ein einfach zu bedienender Editor, der viele Programmiersprachen unter Windows unterstützt.

Download und Installation von Notepad++

- ▶ Laden Sie das Programm Notepad++ von der Webseite <https://notepad-plus-plus.org/downloads/> herunter.
- ▶ Um Notepad++ zu installieren, folgen Sie den Schritten des Installationsassistenten.
- ▶ Starten Sie nach erfolgreicher Installation das Programm.



The screenshot shows the Notepad++ interface with the file 'continue.php' open. The code editor displays the following PHP script:

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title>Schleifenabbruch: continue</title>
6   </head>
7   <body>
8     <?php
9       $zaehler = 5;
10      for ($nenner = -2; $nenner <= 2; $nenner++) {
11        if ($nenner == 0) {
12          echo "<p>Division durch 0 ist verboten.</p>";
13          continue;
14        }
15        $ergebnis = $zaehler / $nenner;
16        echo "<p>$zaehler / $nenner = " . $ergebnis . "</p>";
17      }
18    ?>
19   </body>
20 </html>

```

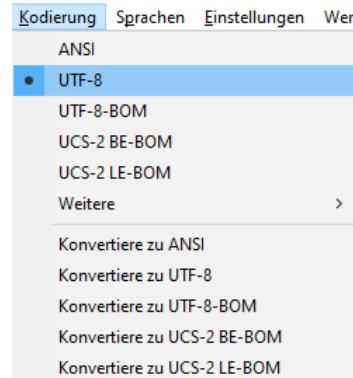
The status bar at the bottom shows: PHP Hypertext length:455 lines: 20 | Ln:1 Col:1 Pos:1 | Windows (CR LF) | UTF-8 | INS | ...

Notepad++: Ansicht mit geöffnetem PHP-Skript

Notepad++ wird bei einer 32-Bit-Windows-Version standardmäßig in den Ordner *C:\Program Files\Notepad++* installiert, bei einer 64-Bit-Version in den Ordner *C:\ Program Files (x86)\Notepad++*. Der Editor ist ohne weitere Konfiguration sofort für die Arbeit mit PHP-Skripten einsatzbereit.

Über das Menü *Einstellungen - Optionen* haben Sie die Möglichkeit, den Editor bei Bedarf Ihren persönlichen Vorstellungen anzupassen, z. B. können Sie PHP als Standardsprache einstellen oder das Farbschema zur Anzeige von PHP-Code ändern.

Seit PHP 5.4 ist der Standard-Zeichensatz auf UTF-8 gesetzt. Dies erleichtert den Umgang mit deutschen Sonderzeichen. UTF-8 unterstützt Umlaute und das ß, sodass diese im Quelltext nicht mehr als HTML-Entities angegeben werden müssen. Stellen Sie sicher, dass die Datei mit dem UTF-8-Zeichensatz gespeichert wird. Diese Einstellung finden Sie in Notepad++ unter dem Menüpunkt *Kodierung*.



Alternative für Mac-Nutzer

Notepad++ steht nur für Windows zur Verfügung. Für Mac-Nutzer ist der Editor **Sublime Text** verfügbar (<https://www.sublimetext.com/>). Beachten Sie die Installationshinweise auf der Webseite.

A.4 Mit den XAMPP-Konfigurationsdateien arbeiten

Auf die Konfigurationsdateien von XAMPP zugreifen

Im XAMPP werden alle Einstellungen klassisch über Konfigurationsdateien verändert. Standardmäßig werden die Dateien bei der Installation von XAMPP wie folgt abgelegt (auf dem Mac per NFS im Ordner `/opt/lampp`):

Konfigurationsdatei (Auswahl der wichtigsten Komponenten)	Verzeichnis (Windows)	Verzeichnis (Mac)
<i>PHP: php.ini</i>	<code>C:\xampp\php</code>	<code>lampp/etc/</code>
<i>Apache: httpd.conf</i>	<code>C:\xampp\apache\conf\</code>	<code>lampp/etc/</code>
<i>MariaDB: my.ini (my.cnf unter Mac)</i>	<code>C:\xampp\mysql\bin\</code>	<code>lampp/etc/</code>
<i>phpMyAdmin: config.inc.php</i>	<code>C:\xampp\phpMyAdmin\</code>	<code>lampp/phpmyadmin/</code>

PHP-Konfigurationsdatei `php.ini`

Änderungen an der Konfigurationsdatei `php.ini` sind Ihnen als Kunde eines Internet-Providers in der Regel nicht möglich. Bei einer lokalen Installation haben Sie hingegen Zugriff auf diese Datei. Zum tieferen Verständnis der Konfigurationsmöglichkeiten ist die Kenntnis der Datei `php.ini` von Vorteil.

Den Pfad zur Konfigurationsdatei erfahren Sie mithilfe der PHP-Funktion `phpinfo()`. Dort finden Sie einen entsprechenden Eintrag unter *Loaded Configuration File*.

Der folgende Code zeigt einen Ausschnitt aus der `php.ini`. Zeilen, die mit einem Semikolon beginnen, sind Kommentarzeilen und werden beim Starten des Webservers ignoriert. In der `php.ini` selbst finden Sie Beschreibungen bzw. Erläuterungen der jeweiligen Einstellung auf Englisch. Durch das Löschen eines Semikolons können Sie die jeweilige Einstellung aktivieren.

```

; Resource Limits ;
;

; Maximum execution time of each script, in seconds
; http://php.net/max-execution-time
; Note: This directive is hardcoded to 0 for the CLI SAPI
max_execution_time=120

; Maximum amount of time each script may spend parsing request data.
It's a good
; idea to limit this time on production servers in order to
eliminate unexpectedly
; long running scripts.
; Note: This directive is hardcoded to -1 for the CLI SAPI
; Default Value: -1 (Unlimited)
; Development Value: 60 (60 seconds)
; Production Value: 60 (60 seconds)
; http://php.net/max-input-time
max_input_time=60

; Maximum input variable nesting level
; http://php.net/max-input-nesting-level
;max_input_nesting_level = 64

; How many GET/POST/COOKIE input variables may be accepted
;max_input_vars = 1000

; Maximum amount of memory a script may consume
; http://php.net/memory-limit
memory_limit=512M

```

Über die *php.ini* wird das komplette Verhalten des PHP-Interpreters gesteuert, hier u. a. der Eintrag `max_execution_time`, welcher die maximale Laufzeit eines PHP-Skripts definiert.



Falls Sie Änderungen in der *php.ini* vornehmen, müssen Sie anschließend den Webserver neu starten, damit die Änderungen wirksam werden. Bei jedem Start des Webservers liest der PHP-Interpreter die Einstellungen erneut ein, erst dann werden veränderte Werte eingelesen und verwendet.

Wichtige Einträge in der *php.ini*

Eintrag	Wirkung
<code>display_errors = On/Off</code>	Schaltet die Fehlermeldungen an/aus
<code>allow_url_fopen = On/Off</code>	Erlaubt/verbietet, dass eine URL wie eine Datei aufgerufen werden kann
<code>session.save_path = "c:/xampp/tmp"</code>	Legt den Speicherort für die Session-Datei fest
<code>session.name = PHPSESSID</code>	Gibt den Standardnamen der Session an

Eintrag	Wirkung
post_max_size	Bestimmt, welche Dateimenge über ein POST-Formular vom Server angenommen wird
max_execution_time	Maximale Ausführungszeit eines PHP-Skripts. Per Standard wird ein PHP-Skript seit PHP 8.0 nach 120 Sekunden (früher: 30 Sekunden) vom Webserver beendet (es bricht ab). Kann erhöht werden, wenn ein Skript länger zur Ausführung benötigt.
memory_limit	Bestimmt, wieviel Arbeitsspeicher ein PHP-Skript verwenden darf. Werden mehr Daten verarbeitet, bricht ein Skript ebenfalls ab. Kann für speicherintensive Skripte verändert werden. Zum Beispiel bei Zugriff auf viele Datenbankdaten oder bei der Bildgenerierung von PHP.

Einen Überblick verschiedener Einstellungen der *php.ini* finden Sie unter <https://www.php.net/manual/de/ini.core.php>.

A.5 Mit MySQL und phpMyAdmin arbeiten

Die grafische Oberfläche des phpMyAdmin erreichen Sie unter <http://localhost/phpmyadmin/> bzw. <http://localhost:8080/phpmyadmin/> auf dem Mac.

Das Berechtigungssystem von MySQL

Das MySQL-Berechtigungssystem steuert den Zugriff von Benutzern auf den MySQL-Server, MySQL-Datenbanken und die darin enthaltenen Tabellen und Felder.

Folgende Benutzerrechte können festgelegt werden:

- ✓ Globale Zugriffsrechte, um den Zugriff auf den MySQL-Server und die Datenbanken auf diesem zu regeln;
- ✓ Zugriffsrechte auf Datenbankebene, um den Zugriff auf eine bestimmte Datenbank und deren Tabellen festzulegen;
- ✓ Tabellenzugriffsrechte zur Steuerung des Zugriffs auf eine Tabelle und deren Spalten;
- ✓ Spaltenzugriffsrechte, um den Zugriff auf eine bestimmte Spalte festzulegen.

Die verschiedenen Zugriffsrechte für Benutzer werden in der Datenbank *mysql* gespeichert.

Die Datenbank *mysql*

Standardmäßig werden mit der Datenbank *mysql* die Benutzerkonten und die Zugriffsrechte verwaltet.

! Der automatisch bei der Installation des MySQL-Servers eingerichtete Benutzer `root` hat uneingeschränkten Zugriff auf den MySQL-Server und somit auch auf die Datenbank *mysql*.

Das Anlegen und Verwalten von Benutzern ist in Abschnitt 12.6 beschrieben.

Benutzer `root` ist ohne Passwort hinterlegt

! Apache-Friends weisen in Ihren FAQ ausdrücklich darauf hin, dass XAMPP nur zu Entwicklungszwecken vorgesehen ist und nicht für den produktiven Einsatz. Der Hinweis ist vor allem auf fehlende Sicherheitsfeatures begründet, u. a. da der `root`-Benutzer kein Passwort verwendet. Um das Lernen der PHP-Grundlagen nicht unnötig zu verkomplizieren, wird das so belassen.

Datenbank-Zugriffsrechte

Die Rechte eines MySQL-Nutzers für PHP-Skript sollten sich daran orientieren, welche Aufgaben zu erfüllen sind. So ist es denkbar, dass einem Nutzer nur Leserechte auf eine Datenbank eingerichtet werden, wenn nur Daten wiedergegeben werden sollen.

In der folgenden Tabelle finden Sie eine Beschreibung der Rechte. Für einen Standard-PHP-Nutzer ist in der Regel die Berechtigung der Befehle SELECT, INSERT, UPDATE und DELETE im Bereich *Daten* vergeben, eventuell die Befehle CREATE (Table), ALTER und DROP im Bereich *Struktur*, wenn auch Anpassungen der Datenbankstruktur notwendig sind. Die anderen Rechte bleiben dem Administrator vorbehalten.

Zugriffsrecht	Beschreibung
Bereich Daten	
SELECT	Das Recht, die SELECT-Anweisung auszuführen, um ausgewählte Datenfelder aus Tabellen abzufragen
INSERT	Die SQL-Anweisung INSERT darf ausgeführt werden, um neue Datensätze in eine Tabelle einzufügen.
UPDATE	Die SQL-Anweisung UPDATE darf ausgeführt werden, um Daten zu bearbeiten und zu ändern.
DELETE	Diese Angabe gewährt das Recht zum Löschen von Daten.
FILE	Der Zugriff auf das lokale Dateisystem des Rechners mit dem MySQL-Server wird gestattet.

Zugriffsrecht	Beschreibung
Bereich Struktur	
CREATE	Das Recht, Datenbanken und Tabellen zu erstellen
ALTER	Das Recht zum Ändern der Tabellenstruktur mit der SQL-Anweisung ALTER
INDEX	Das Recht, Indizes in Tabellen zu erstellen und zu löschen
DROP	Das Recht, Datenbanken und Tabellen zu löschen
Bereich Administration	
GRANT	Das Recht, die gewährten Zugriffsrechte an andere Benutzer weiterzugeben
PROCESS	Das Recht, die Prozessliste einzusehen und Prozesse zu löschen
RELOAD	Das Recht zum Ausführen der FLUSH-Anweisung bzw. der RELOAD-Anweisung des Programms mysqladmin, um damit die Berechtigungstabelle neu einzulesen
SHUTDOWN	Das Recht, den MySQL-Server mithilfe des Programms mysqladmin herunterzufahren

Hochschulversion

\$			
<u>\$_POST</u>	195	Call-by-value	119
<u>\$_REQUEST</u>	98	CamelCase-Schreibweise	31
<u>\$_SERVER</u>	107	case	58
<u>\$_SESSION</u>	194	Case-sensitiv	76
<u>\$GLOBALS</u>	98	checkdate()	186
<u>.</u>		Codieren, Tipps	16
<u>.</u>	35	config.inc.php	244
<u>.=</u>	35	continue	68, 69
<u><</u>		Cookie-Banner	191
<u><?php</u>	13, 15	count()	171
A		Counter	148
Addition	33	D	
AND	54	date()	176, 182
Anführungszeichen	20	Dateien einbinden	130
Anweisung	49	Dateien sperren	147
Anweisungen, PHP	16	Dateien überschreiben	144
Anweisungsblock	50	Dateinamenerweiterung <i>php</i>	13
Apache	236	Dateizeiger	137, 138
Apache-Webserver	4, 237	Datenausgabe	18
Array	30, 73	Datenbank <i>mysql</i>	247
array()	76	Datenbanken, Oracle	10
Array, assoziatives	74, 76	Datentypen,	
Array, Datentyp	30	automatische Zuweisung	39
Array, Eigenschaften	74	Datentypen, numerische	32
Array, eindimensionales	74	Datentypen, PHP	30
Array, Index	74	Datentypen, Zeichen	35
Array, Kurzschreibweise	78	Datenübertragung bei	
Array, mehrdimensionales	74, 84	Formularen	94
Array, numerisch indiziertes	74	Datum	174
Array, Schlüssel	74	Datum formatieren	176, 181
Array-Typ, passender	90	Datum, Anpassung an Sprache	179
Aufteilen, Zeichenkette	170	Datum, deutsch	179
Ausführungsdauer	184	Datum, Formatanweisungen	177, 181
Auswahl, verschachtelte	55	Datumsdifferenz	183
Auto-Inkrement	215	Datumsprüfung	186
B		default	58
Bedingungen	46	define()	41
Bedingungen verknüpfen	54	Deklaration	32
Benannte Parameter	126	Dezimalpunkt	33
Benutzerkontensteuerung	237	die()	223, 224
BOM, Byte Order Mark	23	Division	33
break	58, 68	double	30
C		do-while-Schleife	64, 66
Callable	30	DSGVO	191
Callbacks	30	E	
Call-by-reference	119	echo	18, 20
F		Einbinden, Datei	130
		else	52
		elseif	57
		empty()	105
		error_reporting()	26
		Escape-Sequenzen	20, 21, 166
		Exponent	33
		Exponent, Basis	33
		Externe Dateien nutzen	135
		Externe Dateien öffnen/lesen	136
		Externe Dateien schließen	136, 139
D		Fallauswahl	58
		FALSE	46
		fatal error	25
		fclose()	139
		Fehlerarten in PHP	25
		Fehlerarten, Benachrichtigung	25
		Fehlerarten, Fehler	25
		Fehlerarten, Warnung	25
		Fehler-Level	11, 39, 218
		Fehlersuche	27
		Feldvariablen	73
		fgets()	138
		file()	141
		FILE_APPEND	146, 147, 206, 207
		file_get_contents()	141
		file_put_contents()	146
		Fließkomazahl	32
		Fließkomazahl, Datentyp	30
		float	30
		flock()	147
		floor()	184
		fopen()	136
		for	67
		foreach()	82, 87, 197
		foreach-Schleife	64, 81, 87
		Formatieren, Zeichenketten	153
		Formularauswertung	94
		Formulardaten	98
		Formulare auswerten	97, 98, 103, 106, 108
		Formulare, Auswertung in derselben PHP-Datei	107
		Formulare, Checkboxen und Radiobuttons	103
		Formulare, Daten eingeben	98
		Formulare, Eingabefelder	97
		Formulare, mehrere Absende- Schaltflächen	105
		Formulare, PHP	94
		Formularelemente	100
		for-Schleife	64, 66
		for-Schleife, Operatoren	68
		fseek()	148, 149
		func_get_args()	124
		function	112, 113, 114, 117
		function()	112
		Funktion	111
		Funktion aufrufen	114, 115
		Funktion erstellen	112
		Funktion, benutzerdefinierte	111

Funktion, call-by-reference	119	J	Notepad installieren	242
Funktion, call-by-value	119		Notepad++	4, 242
Funktion, optionale Parameter	117		<i>notice</i>	25
Funktion, Parameter	113, 116		<i>null</i>	30
Funktion, <i>return</i>	113		Null coalescing-Operator	49
Funktion, Rückgabewert	112, 120		<i>number_format()</i>	155
Funktion, vordefinierte	111	K	Numerische Datentypen	32
Funktionen, variadische	124			
<i>fwrite()</i>	144			
<hr/>				
G		O		
Ganze Zahlen	30		<i>Object</i>	30
Ganzzahl, Datentyp	30		Open-Source-Datenbanksystem	211
GET	94		Operator, arithmetischer	33
<i>getdate()</i>	174		Operatoren, relationale	46
GET-Methode	95		OR	54
<i>gettext()</i>	41		Oracle	10
Gleichheitsoperator	47	L		
Gleitkommazahl	32		<i>parse_error</i>	25
Gleitkommazahl, Datentyp	30		PHP, Anweisungen	16
<i>global</i>	127		PHP, Befehle einfügen	13
Globale Variablen	127		PHP, Definition	5
Greenwich-Zeit	182		PHP, Entwicklung	8
Gültigkeitsbereich, Variablen	127		PHP, Interpreter	5
<hr/>				
H		M		
HTML, PHP-Code in	6		PHP, Kommentare	16
HTML5	100		PHP, Merkmale	8
<i>htmlentities()</i>	24		PHP, öffnendes Tag	7
HTML-Tags in PHP-Anweisungen	19		PHP, schließendes Tag	7
HTTP	94, 190		<i>php.ini</i>	244
<i>httpd.conf</i>	244		PHP-Anweisungen trennen	16
Hypertext Transfer Protocol	94		PHP-Code, in HTML einfügen	7
Hypertext-Übertragungsprotokoll	94		PHP-Editor	243
<hr/>			<i>phpinfo()</i>	14, 15
I			PHP-Informationsdatei	14
Identisch-Operator	47, 163		phpMyAdmin	210, 237
<i>if</i>	49, 55, 57		phpMyAdmin, Tabelle erstellen	213
<i>if, else</i>	52		PHPSESSID	192
<i>if, elseif</i>	57		PHP-Skript, abwärtskompatibles	117
<i>if, verschachteltes</i>	55		PHP-Version anzeigen	15
<i>implode()</i>	170		POST	94
<i>include()</i>	130		Postdekrement	34
<i>include_once()</i>	131		Postinkrement	34
Index	74		POST-Methode	95
Initialisierung	32		Potenz	33
Installation	236		Prädekrement	33, 34
<i>int</i>	30, 63, 73, 75		Präinkrement	33
<i>integer</i>	30		Primärschlüssel	215
<i>is_iterable()</i>	83		<i>print_r()</i>	99, 175, 196
<i>is_numeric()</i>	204		<i>printf()</i>	151
<i>isset()</i>	83, 105	N		
<i>Iterables</i>	30			
<hr/>				
R				
			<i>readfile()</i>	141
			Rechenregeln, mathematische	33
			Relationale Operatoren	46
			<i>require()</i>	130

require_once()	131	strrstr()	157	Webserver	236
Reservierte Wörter, PHP	31, 113	strtolower()	167	Webserver testen	241
<i>Resource</i>	30	strtotime()	185	Webserver, PHP-Unterstützung	8
return	113	strtoupper()	167	Wertzuweisung	32
rtrim()	166	strtr()	168	while	62, 64
		substr()	162, 205	while-Schleife	64
S					
Schleife, for	66	substr_count()	164		
Schleife, foreach	81	Subtraktion	33	X	
Schleife, while	64	Suchen, Zeichen	161	XAMPP	4, 6, 210, 211, 236
Schleifen verwenden	64	Superglobale Variablen	127	XAMPP Control Panel	239
Schlüssel	76, 77	switch	58	XAMPP konfigurieren	244
Schlüssel, Array	74	switch, break	58	XAMPP, Arbeit mit	239
Schlüsselwörter, PHP	31	switch, case	58	XAMPP,	
Schreibweise, CamelCase	31	switch, default	58	document root-Verzeichnis	15
Selektion, switch-Anweisung	58	switch-case, erweiterte Notation	61	XAMPP, Download	237
Session	190			XAMPP, Installationsprobleme	238
Session fortsetzen	192	Texteditor	242	XAMPP, root directory	242
Session konfigurieren	191	Throwable Interface	28	XAMPP-Installationsassistent	237
Session löschen	198	time()	182	XML-Schreibweise	13, 14
Session starten	192	Tipps zum Codieren	16	XOR	54
Session, Daten lesen	197	Trailing comma	12, 117	Z	
Session, Daten löschen	198	trim()	166	Zahlen formatieren	155
Session, Name	192	TRUE	46	Zeichenkette, Datentyp	30
session_destroy()	198, 200			Zeichenketten	30, 35
session_id()	192	ucfirst()	167	Zeichenketten ausgeben	151
session_name()	193	ucwords()	167	Zeichenketten formatieren	151
session_start()	191, 192	Union Types	12, 123	Zeichenketten modifizieren	166
Session-Datei	194	UNIX-Timestamp	182	Zeichenketten suchen	157, 158
Session-Datei anzeigen	196	unset()	198, 199	Zeichenketten umwandeln	
Session-ID	191	UTF-8	23	in Arrays	170
setlocale()	180	utf8_encode()	24	Zeichenketten vergleichen	157, 165
Sitzungen	190			Zeichenketten wiederholen	166
Skalare Ausdrücke	42			Zeichenketten, Konkatenation	35
Spaceship-Operator	48			Zeichenketten, Länge	163
Splat-Operator	124			Zeichenketten,	
SQL	211	Value	74	Position ermitteln	161
SQL-Abfrage senden	227	var_dump()	99	Zeichenketten,	
SQL-Dump	218	Variablen	30	Teilstring bestimmen	162
sqrt()	116	Variablen, Ausgabe von	36	Zeichenketten, Vergrößern von	35
str_contains()	159	Variablen, globale/lokale/		Zeichenketten, Verketten von	35
str_ends_with()	160	superglobale	127	Zeichenketten,	
str_repeat()	166	Variablen, Gültigkeitsbereich	127	Zeichen austauschen	168
str_replace()	169	Variablen, Wertzuweisung	32	Zeichenketten, Zeichenfolge	
str_starts_with()	160	Variablenname	31	austauschen	168
strcasecmp()	165	Variadische Funktionen	124	Zeichenkettenoperator	35
strchr()	158	Vergleichsoperatoren	46, 48	Zeichensatz, UTF-8	23
strcmp()	165	Verkettungsoperator .	35	Zeit	174, 182
strftime()	181, 182	Verschachtelte Auswahl	55	Zeitspannen berechnen	184
<i>string</i>	30	Vorgabewert	118	Zeitstempel	182, 184
stristr	157			Zend Engine	9
strlen()	163			Zend Technologies Ltd.	9
strpos()	161	Wahrheitswert, boolscher Wert	30	Zugriffszähler	148
strrchr()	158	warning	25	Zuweisungsoperator =	32
strrpos()	161			Zuweisungsoperatoren	34

Impressum

Matchcode: GPHP8

Autor: Stephan Heller

Produziert im HERDT-Digitaldruck

1. Ausgabe, Mai 2021

HERDT-Verlag für Bildungsmedien GmbH
Am Kümmerling 21–25
55294 Bodenheim
Internet: www.herdt.com
E-Mail: info@herdt.com

© HERDT-Verlag für Bildungsmedien GmbH, Bodenheim

Alle Rechte vorbehalten. Kein Teil des Werkes darf in irgendeiner Form (Druck, Fotokopie, Mikrofilm oder einem anderen Verfahren) ohne schriftliche Genehmigung des Verlags reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Dieses Buch wurde mit großer Sorgfalt erstellt und geprüft. Trotzdem können Fehler nicht vollkommen ausgeschlossen werden. Verlag, Herausgeber und Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.

Wenn nicht explizit an anderer Stelle des Werkes aufgeführt, liegen die Copyrights an allen Screenshots beim HERDT-Verlag. Sollte es trotz intensiver Recherche nicht gelungen sein, alle weiteren Rechteinhaber der verwendeten Quellen und Abbildungen zu finden, bitten wir um kurze Nachricht an die Redaktion.

Die in diesem Buch und in den abgebildeten bzw. zum Download angebotenen Dateien genannten Personen und Organisationen, Adress- und Telekommunikationsangaben, Bankverbindungen etc. sind frei erfunden. Eventuelle Übereinstimmungen oder Ähnlichkeiten sind unbeabsichtigt und rein zufällig.

Die Bildungsmedien des HERDT-Verlags enthalten Verweise auf Webseiten Dritter. Diese Webseiten unterliegen der Haftung der jeweiligen Betreiber, wir haben keinerlei Einfluss auf die Gestaltung und die Inhalte dieser Webseiten. Bei der Bucherstellung haben wir die fremden Inhalte daraufhin überprüft, ob etwaige Rechtsverstöße bestehen. Zu diesem Zeitpunkt waren keine Rechtsverstöße ersichtlich. Wir werden bei Kenntnis von Rechtsverstößen jedoch umgehend die entsprechenden Internetadressen aus dem Buch entfernen.

Die in den Bildungsmedien des HERDT-Verlags vorhandenen Internetadressen, Screenshots, Bezeichnungen bzw. Beschreibungen und Funktionen waren zum Zeitpunkt der Erstellung der jeweiligen Produkte aktuell und gültig. Sollten Sie die Webseiten nicht mehr unter den angegebenen Adressen finden, sind diese eventuell inzwischen komplett aus dem Internet genommen worden oder unter einer neuen Adresse zu finden. Sollten im vorliegenden Produkt vorhandene Screenshots, Bezeichnungen bzw. Beschreibungen und Funktionen nicht mehr der beschriebenen Software entsprechen, hat der Hersteller der jeweiligen Software nach Drucklegung Änderungen vorgenommen oder vorhandene Funktionen geändert oder entfernt.