

Philipp Rieber

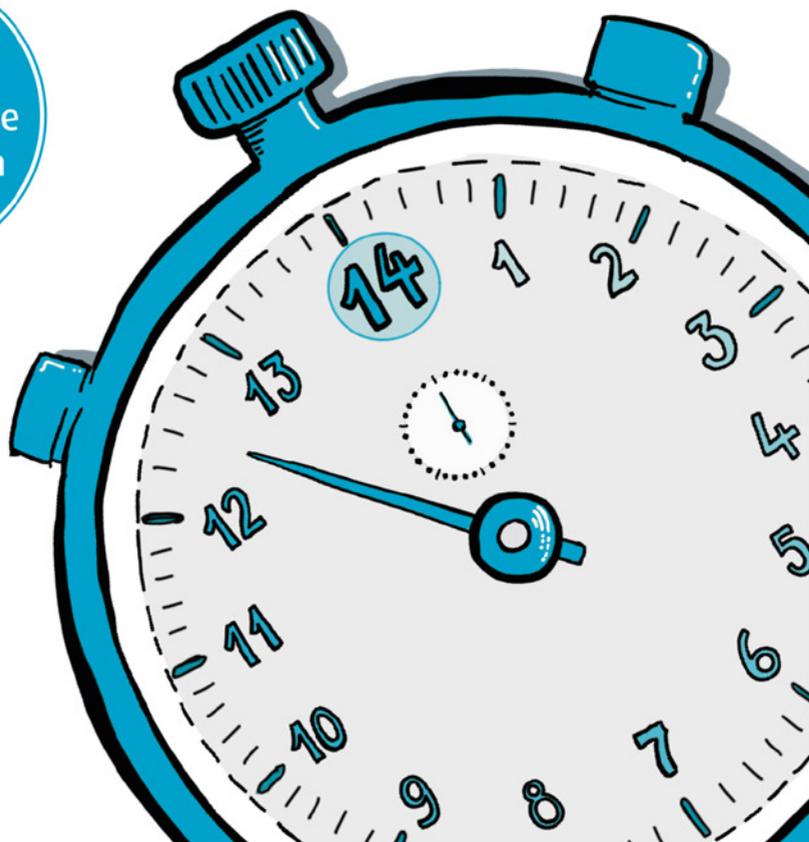
12,99 €
inkl. E-Book

PHP & MySQL Schnelleinstieg

Programmieren lernen in 14 Tagen

> Einfach und ohne Vorkenntnisse <

Zahlreiche
Praxisbeispiele
und Übungen



Inhalt

Einleitung

E.1	Programmieren lernen in 14 Tagen	11
E.2	Der Aufbau des Buchs	11
E.3	Programmtexte und Lösungen zum Download	12
E.5	Fragen und Feedback	12



Erste Schritte mit PHP

1.1	Wofür wird PHP eingesetzt?	13
1.2	Stärken von PHP	22
1.3	Schwächen von PHP	23
1.4	Was ist für die Entwicklung mit PHP notwendig?	24
1.5	Quelltext und PHP-Interpreter	25
1.6	Kommandozeile nutzen	26
1.7	PHP installieren	31
1.8	Quelltext-Editor verwenden: Visual Studio Code	36
1.9	Das erste PHP-Skript ausführen	38
1.11	Wie entstand PHP?	41
1.12	Übungen	42



Variablen, Datentypen und Konstanten

2.1	Daten in einer Variablen erfassen	43
2.2	Einfache Datentypen	50
2.3	Der spezielle Datentyp: NULL	58
2.4	Der vielseitige Datentyp: Array	59
2.5	Datentypen umwandeln	63
2.6	Programmstrukturen	66

2.7	Konstanten	67
2.8	Übungen	70



Programmablauf mit Kontrollstrukturen steuern

3.1	Programmablauf verzweigen	73
3.2	Bedingungen formulieren	81
3.3	Programmabschnitte mit Schleifen wiederholen	85
3.4	Alternative Syntax für Kontrollstrukturen	94
3.5	Übungen	95



Programmierfehler und PHP-Konfiguration

4.1	Programmierfehler	97
4.2	PHP konfigurieren: die php.ini-Datei	99
4.3	Fehlerstufen	102
4.4	Fehlersichtbarkeit einstellen	103
4.5	Übungen	104



Funktionen

5.1	Native Funktionen aus der PHP-Bibliothek verwenden	105
5.2	Eigene Funktionen definieren	122
5.3	Übungen	127



Webseiten entwickeln und veröffentlichen

6.1	Was geschieht beim Abruf einer Webseite?	129
6.2	Webserver auf dem eigenen Computer betreiben	132
6.3	HTML-Grundgerüst	138
6.4	Anfragen und Antworten mit dem HTTP-Protokoll	140
6.5	Webspace mieten und Webseite veröffentlichen	146
6.6	Übungen	148



Dynamische Webseiten und Formulare

7.1	\$_GET: Daten aus dem Query-String der URL	149
7.2	Formulardaten im Query-String der URL übermitteln	158
7.3	Einsatzgebiete der GET-Methode	165
7.4	\$_POST: Formulardaten unsichtbar übermitteln	166
7.5	Vergleich zwischen GET und POST	169
7.6	Übungen	170



Mit Dateien arbeiten

8.1	Quelltext in mehreren Dateien strukturieren	173
8.2	Dateien schreiben und lesen	180
8.3	Dateien über das Internet laden	182
8.4	Datei-Uploads: \$_FILES	184
8.5	Datei-Zugriffsrechte	190
8.6	Übungen	192



Cookies und Sessions

9.1	Cookies	195
9.2	Sessions: Benutzersitzungen	202
9.3	Übungen	210



Einstieg in die objektorientierte Programmierung (OOP)

10.1	Grenzen der prozeduralen Programmierung	211
10.2	Grundbegriffe der »OOP«	212
10.3	Vererbung	221
10.4	Ausnahmen (Exceptions)	224
10.5	Native Objektorientierung in PHP	227
10.6	Übungen	229



Datenverwaltung mit MySQL

11.1	Was ist eine relationale Datenbank?	231
11.2	Was ist ein Datenbankmanagementsystem?	232
11.3	Was ist SQL?	233
11.4	Erste Schritte mit MySQL	234
11.5	Kommentare	241
11.6	Datenbanken	241
11.7	Tabellen	243
11.8	Bezeichner	247
11.9	Daten schreiben, lesen, ändern und löschen	248
11.10	Übungen	253



Fortgeschrittene Datenbankabfragen mit MySQL

12.1	Grafische Datenbankverwaltung mit phpMyAdmin	255
12.2	MySQL beim Webhoster	259
12.3	Fortgeschrittene Datenabfragen	260
12.4	Alias-Namen für Tabellenspalten	264
12.5	SQL-Funktionen	265
12.6	Performance und Indizes	269
12.7	Übungen	272



PHP und MySQL kombinieren

13.1	PHP mit MySQL verbinden	273
13.2	Beispiel: MySQL-Version abfragen	278
13.3	Beispiel: Benutzer-Accounts	278
13.4	Passwörter sicher speichern	284
13.5	Sicherheitslücke: SQL-Injection	285
13.6	Übungen	288



Abschlussprojekt: Ein Blog programmieren

14.1	Installation	289
14.2	Übersicht: Das fertige Blog	290
14.3	Verwendete Techniken	291
14.4	So funktioniert das Blog	296
14.5	Anwendungsbeispiele	301
14.6	Ausblick: Was kommt als Nächstes?	308
14.7	Übungen	311

Stichwortverzeichnis

Einleitung

E.1 Programmieren lernen in 14 Tagen

Mit diesem Buch haben Sie sich für einen einfachen, praktischen und fundierten Einstieg in die Welt der Programmierung entschieden. Sie lernen ohne unnötigen Ballast alles, was Sie wissen müssen, um PHP und MySQL effektiv für Projekte in Ihrem Berufs- und Interessensgebiet einzusetzen.

Wenn Sie Zeit genug haben, können Sie jeden Tag ein neues Kapitel durcharbeiten und so innerhalb von zwei Wochen Programmieren lernen. Alle Erklärungen sind leicht verständlich und setzen keine Vorkenntnisse voraus. Am besten lesen Sie das Buch neben der Computer-Tastatur und probieren die Programmbeispiele und Übungen gleich aus. Sie werden schnell erste Erfolge erzielen und Freude an der Programmierung finden.

E.2 Der Aufbau des Buchs

Die Kapitel bauen Schritt für Schritt aufeinander auf. Das Buch beschreibt zunächst die Einsatzgebiete von PHP und die Einrichtung Ihres Computers zur Entwicklung des ersten PHP-Programms. Anschließend lernen Sie wichtige Sprachelemente und Einstellungen von PHP kennen. Sie werden vertraut mit der Verwendung von Funktionen, der Entwicklung und Veröffentlichung dynamischer Webseiten, dem Lesen und Schreiben von Dateien, der Anbindung eines externen Webservices sowie dem Einsatz von Cookies und Sessions. Nach einer Einführung in die objektorientierte Programmierung erlernen Sie den Umgang mit einem MySQL-Datenbanksystem und der Datenbanksprache SQL. Das letzte Kapitel vereint alles Gelernte in einem Projekt zur Programmierung eines Weblogs und gibt Anregungen zur Weiterentwicklung Ihrer Programmierkenntnisse nach dem Schnelleinstieg.

Gelegentlich stoßen Sie auf kleine Aufgaben oder Zwischenfragen, die als Lernaktivierung gedacht sind. Ihr Tagespensum schließt mit praktischen Programmierübungen, in denen Sie Ihr neu gewonnenes Wissen vertiefen können. Die Lösungen zu diesen Übungen und die Antworten zu den Zwischenfragen stehen in einem Online-Kapitel zum Download zur Verfügung. Mehr dazu im nächsten Abschnitt.

Am Ende des Buchs finden Sie ein Stichwortverzeichnis, das Ihnen hilft, bestimmte Themen im Buch schneller zu finden.

E.3 Programmtexte und Lösungen zum Download

Das Buch enthält viele kleine Beispielprogramme. Sie sind als »Starterprojekte« gedacht und sollen Sie ermuntern, den Code weiterzuentwickeln und selbst etwas Neues auszuprobieren.

Der Code aller Beispielprogramme sowie die Lösungen zu den Übungen und Zwischenfragen stehen Ihnen auf der Webseite des Verlags unter www.mitp.de/0395 zum Download zur Verfügung.

Dort finden Sie außerdem ein praktisches Glossar mit den wichtigsten Fachbegriffen.

E.5 Fragen und Feedback

Unsere Verlagsprodukte werden mit großer Sorgfalt erstellt. Sollten Sie trotzdem einen Fehler bemerken oder eine andere Anmerkung zum Buch haben, freuen wir uns über eine direkte Rückmeldung an lektorat@mitp.de.

Falls es zu diesem Buch bereits eine Errata-Liste gibt, finden Sie diese unter www.mitp.de/0395 im Reiter **DOWNLOADS**.

Wir wünschen Ihnen viel Erfolg und Spaß bei der Programmierung mit PHP und MySQL!

Philipp Rieber und das mitp-Lektorat



Erste Schritte mit PHP

Dieses Kapitel gibt eine praxisnahe Einführung in eine der populärsten Programmiersprachen des Internets: PHP. Um die 80 % aller Webseiten werden von PHP erzeugt. Das Spektrum reicht von Internetpräsenzen, Blogs, Portalen, Online-Shops und spezialisierten Web-Anwendungen bis zu Schnittstellen für die Datenverarbeitung von Mobile Apps und dem Internet of Things. PHP ist für Hobby-Anwender und den professionellen Einsatz in geschäftskritischen Softwaresystemen gleichermaßen geeignet. PHP-Kenntnisse eröffnen Ihnen die Welt hinter den graphischen Benutzeroberflächen des Internets und unzählige Möglichkeiten, um selbst privat oder beruflich in die Webentwicklung einzusteigen.

Nach einem Überblick zu den Einsatzgebieten von PHP führe ich Sie in diesem Kapitel zur erfolgreichen Ausführung Ihres ersten PHP-Programms auf dem eigenen Computer. Dabei erlernen Sie wichtige Grundkenntnisse und die Einrichtung einer Entwicklungsumgebung. Durch erste Programmbeispiele machen Sie sich »hands-on« an der Tastatur Ihres Computers mit den Grundeigenschaften von PHP vertraut. Zum Abschluss des Kapitels erhalten Sie einen Überblick zur Entstehungsgeschichte von PHP.



Wo vorhanden, verwendet dieses Buch deutsche Fachbegriffe. Da die englischen Entsprechungen für Recherchen, Fehlersuchen oder in der Kommunikation mit anderen Programmierern unerlässlich sind, mache ich Sie nebenbei auch mit den englischen Begriffen vertraut.

1.1 Wofür wird PHP eingesetzt?

PHP ist eine kostenlose, universell einsetzbare Programmiersprache. Sieht man von Nischen wie der Programmierung von Alexa Skills oder Desktop-Programmen ab, konzentriert sich der Einsatz auf drei Einsatzgebiete:

- Erzeugung dynamischer Webseiten
- Bereitstellung von Webservices
- Kommandozeilenprogramme

Die nächsten Abschnitte erklären Grundlagen zu den verschiedenen Gebieten. Stellen Sie sich zur Veranschaulichung eine fiktive Zeitungsredaktion vor, die eine Präsenz im Internet aufbaut. Schrittweise entwickelt sich die Internetpräsenz von einer reinen Text-Webseite über eine ansehnlichere HTML-Webseite zu einer fortschrittlichen dynamischen Webseite. Anschließend veröffentlicht die Zeitung ihre eigene Mobile App und automatisiert wiederkehrende Aufgaben.

1.1.1 PHP zur Erzeugung dynamischer Webseiten

Was ist der Unterschied zwischen einer *statischen* und einer *dynamischen* Webseite? Der Abruf einer statischen Webseite von einer Internetadresse im Web-Browser liefert das immer gleiche, »statische« Ergebnis. Eine dynamische Webseite hingegen wird erst im Zuge des Abrufs erzeugt. Dabei werden Inhalte aus verschiedenen Quellen wie Datenbanken oder externen Webdiensten zusammengetragen und zur Anzeige aufbereitet. Eingaben des Benutzers oder dessen Kontext (Identität, Standort, Tageszeit etc.) können den Inhalt beeinflussen.

Eine fiktive Zeitungsredaktion schreibt für ihre ersten Schritte zu einer Internetpräsenz alle Artikel in die einfache Textdatei `articles.txt` und veröffentlicht sie auf einem Computer im Internet, dem *Webserver*.



Abb. 1.1: Die Textdatei `articles.txt` im Text-Editor

Interessierte Leser rufen die Datei anhand der passenden Internetadresse (*URL*, Uniform Resource Locator) in einem Browser auf. Die Datei wird vom Webserver auf den eigenen Computer, den *Client*, übertragen und im Browserfenster angezeigt. Solange die Redaktion die Text-Datei nicht durch eine aktualisierte Version ersetzt, führt jeder weitere Aufruf zur immer gleichen, »statischen« Anzeige des Inhalts – auch für jeden anderen Besucher.



Abb. 1.2: Statische Text-Webseite articles.txt im Browser

Da die Gestaltungsmöglichkeiten mit reinem Text begrenzt sind, wechselt die Redaktion auf die Nutzung der Auszeichnungssprache HTML (*Hypertext Markup Language*).



Abb. 1.3: Die HTML-Datei articles.html im Text-Editor

HTML ermöglicht die Strukturierung der Inhalte mit Hilfe von maschinenlesbaren Hinweisen, den *HTML-Tags*. Die HTML-Tags markieren eingeschlossene Inhalte dabei mit einer gewünschten Bedeutung, zum Beispiel Überschrift, Link, Liste etc. Dies nennt man *semantische* Strukturierung. Die Redaktion verwendet im Beispiel Elemente für eine Überschrift ersten Grades (*heading 1* = *h1*), Hyperlinks (*anchor* = *a*) und eine ungeordnete Liste (*unordered list* = *ul*) mit Listenelementen (*list item* = *li*). Die Auszeichnungen beginnen mit einem öffnenden Tag `<eLement>` und enden mit einem schließenden Tag `</eLement>`:

```
<h1>Überschrift ersten Grades</h1>
<a href="https://www.google.de">Link zu Google</a>
<ul>
  <li>Erstes Listenelement</li>
  <li>Zweites Listenelement</li>
</ul>
```

HTML-Tags bleiben für den menschlichen Betrachter unsichtbar. Der Browser versteht jedoch die versteckten Auszeichnungen, stellt die Inhalte entsprechend dar und schafft einfache Interaktion durch klickbare Links.

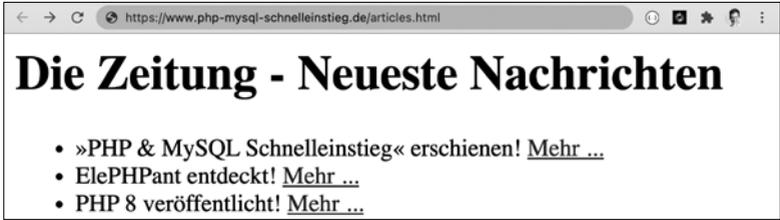


Abb. 1.4: Statische HTML-Webseite articles.html im Browser



Eine genaue Kontrolle über Formatierungen (Farben, Schriftgrößen, Positionierungen usw.) ermöglicht die ergänzende Formatierungssprache CSS (Cascading Style Sheets). Die CSS-Formatierungsangaben sind für den Betrachter ebenso unsichtbar wie HTML-Tags, der Browser nutzt sie jedoch zur Anpassung der Darstellung. Mehr zu HTML und CSS erfahren Sie z.B. unter <https://wiki.selfhtml.org>.

Abbildung 1.5 zeigt den Kreislauf aus Anfrage des Browsers an einen Webserver und dessen Antwort. Dieser Kreislauf wiederholt sich bei jeder angefragten Webseite.

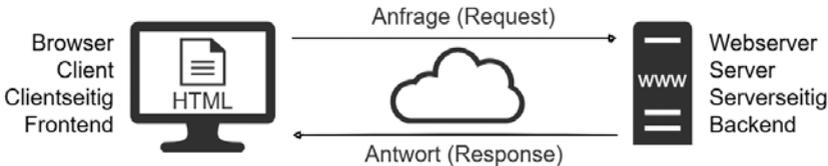


Abb. 1.5: Der Kreislauf aus HTTP-Anfrage und -Antwort

Bald kommt in der Zeitungsredaktion eine neue Idee auf: Sie möchte das aktuelle Tagesdatum ohne tägliche manuelle Bearbeitung einblenden. Doch HTML kann keine Inhalte erzeugen und hat keinen Zugriff auf eine Uhr mit dem aktuellen Datum. Für diesen Zweck ist Programmierlogik erforderlich. Die Zeitung engagiert eine Webagentur. Die Agentur aktiviert PHP auf dem Webserver, benennt `articles.html` in `articles.php` um und beginnt PHP-Programmlogik zur Anzeige des aktuellen Datums in das HTML einzubetten:

```
<h1>Nachrichten</h1>
<p>Heute ist der <?php echo date('d.m.Y'); ?>!</p>
<ul>...</ul>
```

Im Gegensatz zur HTML-Datei liefert der Webserver die PHP-Datei nicht direkt an den Browser zurück, sondern lässt zunächst den enthaltenen PHP-

Programmcode ausführen. Alle PHP-Bereiche werden durch die dabei generierten Ausgaben ersetzt.

<?php	Hier beginnt PHP-Programmlogik
echo	Anweisung zur Ausgabe
date('d.m.Y')	Aufruf der in PHP eingebauten Funktion date(), die das aktuelle Datum in einem gewünschten Format liefert. Im Beispiel wird das Format d.m.Y = day.month.Year = Tag.Monat.Jahr verwendet.
;	Ende der Anweisung
?>	Hier endet PHP-Programmlogik

Aus dem bestehenden HTML und den durch PHP dynamisch ergänzten Inhalten ergibt sich die gewünschte Webseite, die an den Browser zurückgeschickt wird. Bei einem Abruf der Webseite am 13. März 2022 lautet der generierte Inhalt:

```
<h1>Nachrichten</h1>
<p>Heute ist der 13.03.2022!</p>
<ul>...</ul>
```

Aufgabe 1

Können Sie die Dokumentation zur PHP-Funktion date() auf <https://www.php.net> finden?

Aus Sicht des Browsers auf dem eigenen Computer, des Clients, erscheint die empfangene Webseite genauso statisch wie zuvor. Die dynamische Erzeugung erfolgte bereits *serverseitig* auf dem entfernten Webserver. Eine Installation von PHP ist daher nur auf dem Webserver erforderlich, nicht auf den Computern der Webseiten-Besucher. Der Browser kümmert sich wie zuvor nur um die Darstellung, unabhängig von der Entstehung des Inhalts.

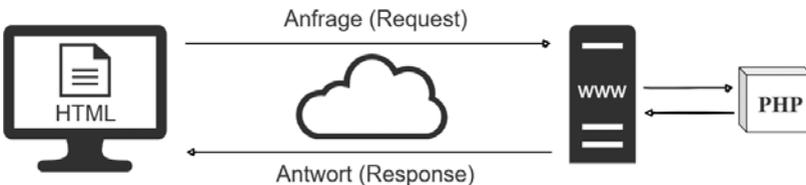


Abb. 1.6: Der Kreislauf aus Anfrage und Antwort mit PHP

Die Pflege der Zeitungsartikel in der Datei durch die Redakteure erfordert HTML-Kenntnisse, Absprachen zwischen den Redakteuren und ständige Übertragungen neuer Versionen auf den Webserver. Mit fortgeschrittenen Methoden der PHP-Entwicklung kann die Webagentur den nächsten Wunsch der Zeitungsredaktion realisieren: Eine Vereinfachung der Artikel-Verwaltung, die von den Redakteuren keine technischen Kenntnisse mehr erfordert. Die Artikelinhalte werden nicht länger in der Datei `articles.php` gepflegt, sondern in eine Datenbank ausgelagert. PHP kann die Artikel zum Zeitpunkt des Abrufs der Webseite aus der Datenbank lesen und ähnlich wie zuvor das Tagesdatum dynamisch in das HTML einbauen. Weitere Abrufe der Webseite wiederholen die Generierung der Inhalte und damit den Abruf der Artikel aus der Datenbank. Neu in die Datenbank eingepflegte Artikel erscheinen somit automatisch auf der Webseite.

Zur Erstellung neuer Artikel durch Redakteure ergänzt die Webagentur einen durch Login geschützten Bereich mit einem Eingabeformular für neue Artikel. PHP überträgt die Eingaben in die Datenbank. Außer der Browserbedienung benötigen die Redakteure keine weiteren technischen Kenntnisse.

Aufgabe 2

Was geschieht, wenn Sie eine PHP-Datei direkt im Browser öffnen? Ziehen Sie zum Ausprobieren eine PHP-Datei in das Browserfenster (Drag & Drop). Vergleichen Sie mit Dateien vom Typ TXT, HTML, ZIP und PDF.

1.1.2 PHP zur Bereitstellung von Webservices

Die fiktive Zeitungsredaktion aus dem vorigen Abschnitt stellt einen großen Anteil an Smartphone-Besuchern auf ihrer Webseite fest. Sie erweitert das Angebot daher um eine eigene Mobile App. Diese folgt bezüglich Gestaltung und Bedienung anderen Gesetzen als eine Webseite für den Browser. Die App ist nicht am Abruf einer HTML-Webseite interessiert, sondern nur an den reinen Artikeldaten, um damit die Gestaltungselemente einer Mobile App zu füllen. In der ersten Version baut die Webagentur einen einfachen »Nachrichtenticker« zur Anzeige der Artikel mit Datum und Überschrift, der sich durch Drücken einer Schaltfläche aktualisieren lässt.

Die Programmierung der Mobile App selbst erfolgt nicht in PHP, sondern in einer Programmiersprache für die jeweilige Plattform, z.B. Java für Android- oder Swift für iPhone-Apps. Wie die meisten Apps muss sie im Hintergrund in Kontakt mit einem zentralen Service stehen, um aktuelle Daten abzurufen

oder Benutzereingaben dorthin zu senden. Den unsichtbaren Kommunikationspartner im Hintergrund bezeichnet man als *_Backend_*. Eine dem Benutzer zugewandte Anwendung wie die Webseite oder die Mobile App heißt *_Frontend_*.

Zur Kommunikation greift die App (das Frontend) ähnlich dem Browser anhand einer URL auf einen Webserver (das Backend) zu, etwa zum Abruf der neuesten Artikel beim Start der App oder bei einer gewünschten Aktualisierung. Statt einer HTML-Webseite mit vielen irrelevanten Bereichen ist die App an einem maschinenlesbaren Format zum Datenaustausch interessiert. Ein verbreiteter Standard dafür ist *JSON* (ausgesprochen: Jason). Die übermittelten Daten für den Nachrichten-Ticker könnten im JSON-Format so aussehen:

```
[
  {
    "title": "ElePHPant entdeckt!",
    "link": "https://www.php.net/elephpant"
  },
  {
    "title": "PHP 8 veröffentlicht!",
    "link": "https://www.php.net/releases/8.0/de"
  },
]
```

Da sich mit PHP neben HTML auch jede andere Art von Ausgabe erzeugen lässt, kann PHP der Mobile App als Backend dienen. Statt HTML gibt das PHP-Skript JSON aus:

```
<?php
$articles = Abruf der Artikel aus der Datenbank;
echo json_encode($articles);
?>
```

<?php	Beginn der PHP-Programmlogik
<code>\$articles = ...</code>	Definiert die Variable <code>\$articles</code> – eine Art Daten-Container für die Zeitungsartikel. Die Variable ist eine Referenz zum späteren Zugriff auf die Daten.
<i>Abruf der Artikel aus Datenbank</i>	Steht stellvertretend für Programmcode zum Abruf der Daten aus einer Datenbank.

;	Ende der Anweisung
echo	Anweisung zur Ausgabe
json_encode(\$articles)	Eine in PHP eingebaute Funktion, die die übergebenen Zeitungsartikel-Daten im JSON-Format zurückliefert.
;	Ende der Anweisung
?>	Ende der PHP-Programmlogik

Mit Hilfe von PHP stellt der Webserver die Artikel als Service über das Internet zur Verfügung – als *Webservice*. Im Vergleich zu einer Webseite für menschliche Benutzer richtet sich das Angebot eines Webservice an andere Software – unabhängig vom endgültigen Verwendungszweck.

Später entscheidet sich die Zeitung zur Veröffentlichung ihres Webservice. Ein Newsletter-Versender kann die verfügbaren Daten zum Beispiel unabhängig von der Zeitungs-Webseite nutzen, um seinen wöchentlichen E-Mail-Newsletter automatisiert mit Inhalten der Zeitung anzureichern. In umgekehrter Weise erweitert die Zeitung ihr eigenes Angebot um einen Wetterbericht, dessen Daten sie vom Webservice eines Drittanbieters abrufen.



Ein Webservice wird oft auch technischer als *Web-API* (Web Application Programming Interface) bezeichnet – eine Programmierschnittstelle über das Internet.

Weitere Beispiele für Webservices beziehungsweise Web-APIs: Währungs- und Aktienkurse, Fahrpläne, SMS- und E-Mail-Versand, Daten zu Sportereignissen, Steuerung von Smart-Home-Geräten, Bonitätsauskünfte, Validierung von Postadressen und vieles mehr. Auch die großen Tech-Unternehmen wie Twitter, Facebook, Google, Amazon etc. stellen Web-APIs zur Integration in eigene Angebote bereit.

Mit Hilfe von PHP können Sie sowohl eigene Webservices bereitstellen (*produzieren*) als auch fremde Webservices nutzen (*konsumieren*). Mit entsprechenden Maßnahmen kann ein Webservice auch gegen Bezahlung angeboten werden.

1.1.3 Kommandozeilenprogramme mit PHP

Ein Kommandozeilenprogramm stellt die einfachste Form eines PHP-Programms dar. Ist PHP auf einem Computer installiert, kann ein solches Programm darauf ausgeführt werden. Ein Browser oder Webserver spielt dabei keine Rolle, die Ausführung des Programms erfolgt rein *clientseitig* bezie-

ungsweise *lokal*. PHP-Kommandozeilenprogramme können vielfältige Aufgaben übernehmen und dabei sichtbare Ausgaben erzeugen. Betrachten Sie das Programm in der Datei `dice.php` zur Simulation eines Würfels:

```
Sie haben eine <?php echo random_int(1, 6); ?> gewürfelt.
```

Listing 1.1: PHP-Kommandozeilenprogramm `dice.php`

<code><?php</code>	Beginn der PHP-Programmlogik
<code>echo</code>	Anweisung zur Ausgabe
<code>random_int(1, 6)</code>	Eine in PHP eingebaute Funktion, die eine zufällige Ganzzahl (<i>random integer</i>) im gewünschten Bereich zwischen 1 und 6 zurückliefert.
<code>;</code>	Ende der Anweisung
<code>?></code>	Ende der PHP-Programmlogik

```

~%1 philipp.riever@solaris:~/php-mysql-schnelleinstieg/kapitel-01
→ kapitel-01 > php dice.php
Sie haben eine 2 gewürfelt.
→ kapitel-01 > php dice.php
Sie haben eine 6 gewürfelt.
→ kapitel-01 > php dice.php
Sie haben eine 6 gewürfelt.
→ kapitel-01 > php dice.php
Sie haben eine 3 gewürfelt.
→ kapitel-01 > █

```

Abb. 1.7: Wiederholte Ausführung von `dice.php`

Wiederkehrende Aufgaben mit Cronjobs erledigen

In Webprojekten spielen PHP-Kommandozeilenprogramme zur Ausführung wiederkehrender Aufgaben eine wichtige Rolle, sogenannte *Cronjobs*. Dabei wird das Programm entsprechend einem gewünschten Zeitplan auf dem Webserver aufgerufen.

Die fiktive Zeitungsredaktion aus dem vorigen Abschnitt möchte interessierten Lesern abends eine Übersicht der Tagesnachrichten per E-Mail anbieten. Statt eines mühsamen manuellen Versands richtet die Webagentur auf der Webseite ein Formular zur Anmeldung für den Service ein. Die eingegebenen E-Mail-Adressen werden durch PHP in einer Datenbank gespeichert. Die Agentur konfiguriert auf dem Webserver einen Cronjob, der jeden Abend um 20 Uhr ein PHP-Kommandozeilenprogramm aufruft. Das Programm liest die

Abonnenten und Artikel des letzten Tags aus der Datenbank und versendet die dynamisch erstellte Artikelübersicht per E-Mail.

Weitere Anwendungsbeispiele für Cronjobs:

- Erinnerungs-SMS an Kunden einer Autowerkstatt für einen bevorstehenden Termin.
- Nächtliche Generierung von Verkaufsberichten für einen Online-Shop.
- Regelmäßige Löschung nicht mehr benötigter Daten gemäß DSGVO.

Hilfsprogramme für die Entwicklung

Es gibt auch PHP-Kommandozeilenprogramme zur Unterstützung der Entwicklung mit PHP. Der Paketmanager *Composer* etwa ist ein in PHP geschriebenes Kommandozeilenprogramm, mit dessen Hilfe hunderttausende kostenlose PHP-Pakete zur Unterstützung eigener Projekte geladen werden können.

Im weiteren Verlauf des Buchs erlernen Sie zunächst die Grundlagen anhand kurzer PHP-Kommandozeilenprogramme. Anschließend kommt die serverseitige Web-Programmierung zur Erzeugung dynamischer Webseiten und Webservices zum Einsatz. Das Abschlussprojekt im letzten Kapitel verwendet Composer-Pakete zur PDF-Generierung und zum E-Mail-Versand.

1.2 Stärken von PHP

- PHP ist kostenlos.
- PHP läuft auf allen gängigen Plattformen, darunter Windows, macOS und Linux/Unix.
- PHP ist mit allen gängigen Webservern einsetzbar, darunter nginx, Apache und IIS.
- PHP kann an alle gängigen Datenbanken angebunden werden. Am beliebtesten ist die Kombination mit den kostenlosen Datenbanksystemen MySQL oder PostgreSQL.
- Breite Unterstützung durch Webhoster, dadurch große Auswahl und günstige Preise, um Projekte online zu stellen.
- Leichte Erlernbarkeit durch: Spezialisierung auf Webprogrammierung; Syntax ähnelt bekannten Sprachen; fehlertolerantes Verhalten; kostenlose Verfügbarkeit; zugängliches Wissen; Einbettung von PHP-Code ist intuitiv.
- Sehr umfangreiche eingebaute Funktionsbibliotheken, bei Bedarf erweiterbar.



Variablen, Datentypen und Konstanten

Computer-Programme verarbeiten Daten. Daten können dazu in zwei Arten von »Containern« erfasst werden: *Variablen* für veränderliche und *Konstanten* für unveränderliche Daten.

Variablen sind eine Art Platzhalter im Quelltext. Sie stehen stellvertretend für Daten, die erst bei der Ausführung eines Programms konkret werden, z.B. Benutzereingaben, Datenbankinhalte oder Zwischenergebnisse bei Berechnungen. Konstanten beinhalten Daten, die fester Bestandteil eines Programms sind, etwa das Zugangspasswort zu einer Datenbank oder eine feste mathematische Größe wie die Kreiszahl π (Pi).

Daten gehören immer einer bestimmten Art an, dem *Datentyp*. Jeder Datentyp weist andere Eigenschaften auf und kann unterschiedlich eingesetzt werden. Einfache Datentypen sind Zeichenketten (Texte), Zahlen und Wahrheitswerte (wahr oder falsch). Mit dem zusammengesetzten Datentyp *Array* werden Daten-Sammlungen dargestellt, z.B. eine Liste aus Zahlen.

Dieses Kapitel zeigt Ihnen den Umgang mit Variablen, Konstanten und Datentypen. Sie erfahren außerdem Grundlegendes zu Programmstrukturen.

2.1 Daten in einer Variablen erfassen

Das erste Programmbeispiel aus dem vorigen Kapitel kombinierte einen festen Text mit einem durch PHP ausgegebenen Text:

```
Willkommen bei <?php echo 'PHP'; ?>!
```

Die Ausführung erfolgt auf der Kommandozeile mit:

```
> php welcome.php  
Willkommen bei PHP!
```

Das Ergebnis kann natürlich auch ohne PHP-Code erreicht werden. Aber beobachten Sie im Folgenden, wie sich das Beispiel mit Hilfe einer Variablen in ein Programm mit dynamischer Ausgabe entwickelt. Um das bislang fest vorgegebene Thema der Begrüßung (»PHP«) dynamisch austauschbar zu machen, wird die Zeichenkette durch eine Variable als Platzhalter ersetzt. Eine Variable beginnt mit einem Dollarzeichen (\$), dem ein beliebiger Name folgt. »\$topic« (engl. *Thema*) erscheint als passender Variablenname:

```
Willkommen bei <?php echo $topic; ?>
```

Der Inhalt der Variable muss zuvor erfasst werden. Er wird der Variablen mit einem Gleichheitszeichen *zugewiesen*:

```
<?php $topic = 'PHP'; ?>  
Willkommen bei <?php echo $topic; ?>!
```

Die Variable lässt sich auch mehrfach im Programm einsetzen:

```
<?php $topic = 'PHP'; ?>  
Willkommen bei <?php echo $topic; ?>!  
<?php echo $topic; ?> ist wunderbar.
```

Listing 2.1: welcome-to-topic.php

Die Ausführung des Programms ergibt:

```
> php welcome-to-topic.php  
Willkommen bei PHP!  
PHP ist wunderbar.
```

Eine einzige Anpassung im Quelltext beeinflusst nun alle Ausgaben:

```
<?php $topic = 'MySQL'; ?> ...
```

Die Ausgabe lautet dann:

```
Willkommen bei MySQL!  
MySQL ist wunderbar.
```

Letztlich ist die Ausgabe des Programms immer noch statisch, die Variable \$topic enthält einen unveränderlichen Wert. Mit der in PHP eingebauten Funktion `readline()` kann es jedoch dem Aufrufer des Programms bei jeder Ausführung selbst überlassen werden, den Inhalt der Variable zu bestimmen. Der Funktionsaufruf unterbricht die Programmausführung zur Erfassung ei-

ner Benutzereingabe. Der in Klammern angegebene Text fordert zur passenden Eingabe auf:

```
<?php $topic = readline('>> Ihr Thema? '); ?>
Willkommen bei <?php echo $topic; ?>!
<?php echo $topic; ?> ist wunderbar.
```

Listing 2.2: welcome-to-your-topic.php

Nach Drücken der Eingabetaste fährt das Programm mit der Ausführung fort und weist die Eingabe der Variablen `$topic` zu. Abbildung 2.1 zeigt einige Beispielaufrufe.



```
→ kapitel-02 > php welcome-to-your-topic.php
>> Ihr Thema? PHP
Willkommen bei PHP!
PHP ist wunderbar.
→ kapitel-02 > php welcome-to-your-topic.php
>> Ihr Thema? MySQL
Willkommen bei MySQL!
MySQL ist wunderbar.
→ kapitel-02 > php welcome-to-your-topic.php
>> Ihr Thema? █
```

Abb. 2.1: Die Benutzereingabe beeinflusst dynamisch die Ausgabe

2.1.1 EVA: Grundprinzip der Datenverarbeitung

Mit dem Beispielprogramm aus dem vorigen Abschnitt haben Sie das Grundprinzip der Datenverarbeitung kennengelernt:



Abb. 2.2: EVA-Prinzip (Eingabe-Verarbeitung-Ausgabe)

Dieses EVA-Prinzip wiederholt sich ständig, auch in der Webprogrammierung:

- Das Programm nimmt Daten als Eingabe entgegen. Die Eingabekanäle können sehr unterschiedlich sein.
- Das Programm verarbeitet die Daten entsprechend seiner Anweisungen.
- Das Programm gibt das Ergebnis der Verarbeitung aus.

Auf der Kommandozeile können Daten wie gezeigt interaktiv vom Benutzer erfasst werden. Im Browser können die Daten z.B. aus einem Formular, Cookies oder der URL stammen.

Hinweis

Wo möglich verzichten die Code-Listings im weiteren Verlauf des Buchs aus Platzgründen auf die PHP-Tags (`<?php ... ?>`). PHP-Code ist immer im Kontext eines PHP-Bereichs zu verstehen. Die vollständigen Skripte finden Sie im Download zum Buch unter www.mitp.de/0395.

2.1.2 Variablen erzeugen und verwenden

Die Erzeugung einer Variablen heißt *Deklaration*. Die Deklaration ist eine Anweisung, bei der ein Name für die Variable festgelegt und ihr per *Zuweisungsoperator* (dem Gleichheitszeichen =) ein Ausgangswert zugewiesen wird:

```
$topic = 'PHP';
```

Im weiteren Programmverlauf dient die Variable als Referenz für den Zugriff auf die enthaltenen Daten, z.B. zur Ausgabe:

```
echo $topic; // => PHP
```

Hinweis

Einfache Skript-Ausgaben finden Sie direkt in einem Code-Kommentar – angedeutet mit einem Pfeil (=>). Das letzte Skript führt somit zur Ausgabe PHP.

Eine Variable kann überall verwendet werden, wo der enthaltene Wert und dessen Datentyp Sinn ergeben. Folgendes Beispiel mit der bereits bekannten `date()`-Funktion zeigt eine indirekte Übergabe des gewünschten Datumsformats mit einer Variablen:

```
$format = 'd.m.Y';  
echo date($format); // z.B. => 22.03.2022
```

Eine Deklaration kann auch mehrere Variablen initialisieren:

```
$topic = $language = 'PHP';
echo $topic . ', ' . $language; // => PHP, PHP;
```

Listing 2.3: define-multiple-variables.php

Eine neue Zuweisung überschreibt den Wert einer Variablen:

```
$format = 'd.m.Y';
echo date($format); // z.B. => 22.03.2022
$format = 'Y-m-d';
echo date($format); // z.B. => 2022-03-22
```

Listing 2.4: format-date.php

2.1.3 Regeln für Variablennamen

Variablen beginnen in PHP immer mit einem Dollarzeichen (\$), gefolgt von einem individuellen Namen, dem so genannten Variablen-*Bezeichner*. Der Bezeichner unterliegt folgenden Regeln:

- Er muss mit einem Buchstaben oder Unterstrich beginnen.
- Danach können Buchstaben, Zahlen und Unterstriche folgen.
- Groß- und Kleinschreibung wird unterschieden (engl. *case-sensitive*).

Beispiele für gültige Variablen:

```
$name = 'Eva';
$name = 'Tim'; // Ungleich $name!
$my_favorite_city = 'Neustadt';
$myFavoriteCity = 'Neustadt';
$_ = 's3cr3t';
$käse = 'Camembert';
$r2d2 = 'err-zwo deh-zwo';
```

Beispiele für ungültige Variablen:

```
name = '...'; // Kein Dollarzeichen
$4gewinnt = '...'; // Beginnt mit Zahl
$r2-d2 = '...'; // Enthält unzulässiges Zeichen
```

Für eine gute Lesbarkeit empfiehlt sich die Verwendung einer einheitlichen Schreibweise der Variablen-Bezeichner, zum Beispiel der sogenannten *camel-Case-Schreibweise* (»Kamelhöcker-Schreibweise«). Dabei wird grundsätzlich klein geschrieben, nur neue Wörter beginnen mit einem Großbuchstaben:

```
$theMostBeautifulCity = 'Neustadt';
```

Die gewählten Variablen-Bezeichner sollten aussagekräftig sein und den Inhalt der Variablen widerspiegeln. Eine Faustregel: nicht geläufige Abkürzungen sind zu vermeiden. »\$fn« zur Erfassung eines Vornamens (»first name«) mag während der Programmierung logisch erscheinen. Für Anpassungen des Programms in der Zukunft oder durch andere Programmierer ist »\$firstName« die verständlichere Wahl.

2.1.4 Datentypen und die »dynamische Typisierung«

Die bisherigen Beispiele wiesen einer Variablen eine Zeichenkette (d.h. einen Text) zu, eingefasst in einfache Anführungszeichen:

```
$topic = 'PHP'; // Zeichenkette; in Anführungszeichen!
```

Statt einer Zeichenkette ist es auch möglich, Zahlen zuzuweisen, die frei von Anführungszeichen stehen:

```
$number = 8; // Zahl; keine Anführungszeichen!
```

Mit Zahlen können Sie z.B. eine Addition durchführen:

```
$number1 = 5; $number2 = 3;
echo $number1 + $number2; // => 8
```

Knifflig wird es allerdings bei der Addition zweier Zeichenketten:

```
$topic1 = 'PHP'; $topic2 = 'MySQL';
echo $topic1 + $topic2; // => ???
```

An dieser Stelle kommt das Konzept der *Datentypen* ins Spiel. Jeder Wert im Programm kann einem Datentyp zugeordnet werden. Bislang sahen Sie die Typen *Zeichenkette* (engl. *String*) und *Ganzzahl* (engl. *Integer*). PHP erkennt Datentypen praktischerweise automatisch im Hintergrund. Kommt es zur Addition zweier Zahlen, kann PHP die Rechnung nach den Regeln der Mathematik lösen. Kommt es zu einer Addition der Zeichenketten PHP und MySQL wie im letzten Listing, reagiert PHP mit einem Fehler, da eine Addition zweier Wörter nicht sinnvoll möglich ist:

```
PHP Fatal error: Uncaught TypeError: Unsupported operand types:
string + string
```

Die Funktion `var_dump()` macht Datentypen (und Werte) sichtbar:

```
$topic = 'PHP'; $number = 8;
var_dump($topic);
var_dump($number);
```

Listing 2.5: var_dump.php

Die Ausgabe lautet:

```
string(3) "PHP" 1
int(8) 2
```

- **1** Der Datentyp der Variablen `$topic` ist *String* (Zeichenkette). Der String hat 3 Zeichen und lautet PHP.
- **2** Der Datentyp der Variable `$number` ist *Integer* (Ganzzahl, Abkürzung `int`), ihr Wert ist die Zahl 8.

Der Datentyp einer Variablen kann sich im Laufe eines Skripts durch eine neue Zuweisung auch einfach ändern:

```
$number = 8; // Datentyp: Integer
$number = 'Acht'; // Neuer Datentyp: String
```

Programmierer müssen den Datentyp bei der Deklaration nicht angeben und eine Variable ist nicht dauerhaft auf einen bestimmten Datentyp festgelegt. Dieses Merkmal von Skriptsprachen heißt *dynamische Typisierung* und macht den Umgang bequem und einsteigerfreundlich.

Die dynamische Typisierung sorgt in vielen Situationen auch für automatische Umwandlungen des Datentyps. Die Anpassungen erscheinen häufig völlig logisch, wie hier die Ausgabe einer Zahl:

```
$number = 8; echo $number; // => 8
```

Tatsächlich können jedoch nur Strings ausgegeben werden. PHP übernimmt die Typumwandlung der Zahl 8 in den String '8' im Hintergrund.

Reagierte PHP bei der Addition zweier Strings auf Grund inkompatibler Operanden mit einem Fehler, sieht es in diesem Fall anders aus:

```
$number1 = '5'; $number2 = '3'; // String-Variablen
$result = $number1 + $number2; // Ergebnis ist Integer
var_dump($result); // => int(8)
```

Listing 2.6: auto-type-conversion.php

Stichwortverzeichnis

Symbole

\$_COOKIE	197
\$_FILES	185
\$_GET	153
\$_POST	167
\$_SERVER	142, 200
\$_SESSION	203
\$this	217
127.0.0.1	133
200 OK	141
404 Not Found	141

A

Aggregatfunktion (SQL)	265, 268
Alias (SQL)	264
AND	262
Anführungszeichen	
doppelte	52
einfache	51
Anführungszeichen (SQL)	261
Anweisung	66
Anweisungsblock	73
API	183
Argument	
Funktion	106
Kommandozeile	29
Array	59
assoziatives	61
mehrdimensionales	62
AS	264
Ausdruck	66
Ausgabe-Maskierung	164
Ausnahmen	224
AUTO_INCREMENT	246

B

Backend	19
Bedingte Anweisung	74
Bedingte Wiederholung	91
Bedingungen	81
Benutzer-Accounts verwalten	278

Benutzereingabe	45
Benutzerklassen	191
Benutzer-Registrierung	280
Benutzersitzung	
<i>Siehe Session</i>	
Bezeichner (Datenbank)	247
Bibliothek	105
Block	
<i>Siehe Anweisungsblock</i>	
Blog	289
Boolean	57
Bootstrap	292
break	93
Browser-Konsole	140
Bug	97

C

Camel Case	47
Casting	
<i>Siehe Type Casting</i>	
CLI	
<i>Siehe Kommandozeile</i>	
Client	14, 131
clientseitig	20
Compiler	26
Composer	22, 293
composer.json	295
composer.lock	295
continue	93
Cookie	195
Ablaufdatum	201
lesen	197
löschen	202
setzen	196
Cookie-Banner	210
CREATE TABLE	243
Cronjob	21
Cross-Site-Scripting (XSS)	163
CSS	16
CSV	193

D

Data Control Language (DCL)	234
Data Definition Language (DDL)	234
Data Manipulation Language (DML)	234, 248
Datei	
Zugriffsrechte	190
Dateiberechtigung	192
Dateien	
schreiben und lesen	180
Datei-Uploads	184
dauerhaft speichern	186
eindeutiger Name	187
Einstellungen	188
Fehler	187
Datenbank	231, 241
löschen	242
non-relationale	232
Performance	269
relationale	231
Datenbankmanagementsystem ..	231, 232
Datenbankserver	233
Datenkapselung	
<i>Siehe Geheimnisprinzip</i>	
Datensatz	231
filtern	260
Datentyp	43, 48, 50
skalarer	50
DateTime	228
DBMS	
<i>Siehe Datenbankmanagementsystem</i>	
Debugging	97
DEFAULT	
<i>Siehe Vorgabewert (Datenbank)</i>	
Deklaration	46
Dekrement	56
DELETE	252
DESCRIBE	247
Dienst	131
Direktive (php.ini)	100
display_errors	103
Document-Root	134
Domain	130
Domain Name System (DNS)	130
Dompdf	294, 305
DROP DATABASE	242
DROP TABLE	247

DSN (Data Source Name)	275
Dynamischer Parameter (SQL)	277

E

echo	17, 39
Eigenschaft	214
Eingabeaufforderung	27
Einkaufswagen	210
Elternklasse	222
E-Mail versenden	306
Endlosschleife	93
Entitäten	
<i>Siehe HTML-Entitäten</i>	
Entwicklungsumgebung	24
Entwicklungszyklus	137
error_log	104
error_reporting	104
Escape-Sequenzen	52
EVA-Prinzip	45
Exception Handler	298
Exceptions	
<i>Siehe Ausnahmen</i>	
Exponentialschreibweise	57
Expression	
<i>Siehe Ausdruck</i>	

F

Fallunterscheidung	78
Fehlersichtbarkeit	103
Fehlerstufen	102
Flag	30, 161
Flash-Nachricht	300
Float	57
for 88	
foreach	86
Formular	
Inhalte wiederanzeigen	161
versteckte Felder	293
Formulareingabe	277
Framework	310
Frontend	19
FTP	148, 192
FULLTEXT	270
Funktion	105
Arrays	115
Datum und Zeit	118
Definition	123
eigene	122

mathematisch	112	INSERT INTO	249
native	105	Instanz	213
Rückgabewert	124	Instanziierung	213, 218
SQL	265	Integer	48, 54
Strings	113	IP-Adresse	129
Vorgabewert	124	IS NOT NULL	262
G		IS NULL	262
Ganzzahl		Iteration	85
<i>Siehe Integer</i>		J	
Geheimnisprinzip	220	JavaScript	24
GET-Methode	165, 169	JSON	19, 183
GET-Parameter		K	
<i>Siehe URL-Parameter</i>		Kindklasse	222
git	310	Klartext	284
Git Bash	27	Klasse	213
GitHub	310	Klausel (SQL)	251
Gleichheit	82	Kommandozeile	25, 26
Gleitkommazahl		Kommandozeilenprogramm	20
<i>Siehe Float</i>		Konstante	43, 67
GUI		benannte	68
<i>Siehe Grafische Benutzeroberfläche</i>		literale	67
H		magische	69
Header	141	vordefinierte	68
Homebrew	35	Konstruktor	218
Host	131	Kontrollstruktur	73
HTML	15, 138	alternative Syntax	94
Element	15, 139	Kopfzeile	
Tag	15	<i>Siehe Header</i>	
HTML-Entitäten	164	L	
HTTP	140	Laufzeit	98
I		Laufzeitfehler	98
Identität	82	Lerdorf, Rasmus	41
if-Anweisung		LIKE	262
<i>Siehe Bedingte Anweisung</i>		LIMIT	263
if-else-Verzweigung		Literal	51
<i>Siehe Verzweigung</i>		localhost	132
if-Klausel	74	Logging	103
Index	270	Login	204, 281
INDEX	270	Logische Fehler	99
Index-Datei	136	Logische Operatoren (SQL)	262
Indizes (SQL)	269	M	
Inheritance		mail()	306
<i>Siehe Vererbung</i>		MariaDB	235
Initialisierung	297	Maskierung	52, 164
Inkrement	55		

Maskierung (SQL)	287	Oracle	235
memory_limit	100	ORDER BY	251
Metadaten	144, 185	Output-Escaping	
Methode	216	<i>Siehe Ausgabe-Maskierung</i>	
statische	308	P	
verketteten	292	Package	294
MIME-Type	145, 189	Parameter	106, 123
MySQL	231, 234, 255, 273	optional	109
Konsole	239	Passwort-Hash	284
mysql-Befehl	239	Passwort speichern	284
MySQLi-Erweiterung	273	Payload	144, 167
MySQL-Version	278	PDF-Download	305
N		PDO::exec()	276
Nameserver	130	PDO::lastInsertId()	292
Namespace	306, 308	PDO (PHP Data Objects)	273, 274
NoSQL	232	PDO::prepare()	277
NOT	262	PDO::query()	276
NULL	58	Permissions	
Null coalescing operator	200	<i>Siehe Zugriffsrechte</i>	
NULL (Datenbank)	245	PHP	13
NULL (SQL)	262	Entstehungsgeschichte	41
O		Erweiterungen	102
Objekt	212, 213	installieren	31
Zustand	216	Interpreter	25
Objektorientierte Programmierung	212	PHP/FI	41
OFFSET	264	phpinfo()	137
OOP		php.ini	100
<i>Siehe Objektorientierte Programmie-</i>		phpMyAdmin	255
<i> rung</i>		Port	256
Operator		PhpStorm	38
arithmetisch	54	PHPUnit	311
Casting	64	Pipe	110
Dekrement	56	Platzhalter (SQL)	262
Inkrement	55	Port	131
instanceof	223	POST-Methode	166, 169, 293
logisch	83	Präzedenz	83
new	214	Präzedenz (SQL)	263
Objekt (->)	215	Prepared Statements	277
ternär	58, 77	Primärschlüssel	246
Vergleich	81	PRIMARY KEY	270
Verknüpfung	53	Produktionsumgebung	24
Zuweisung	46	Programmierparadigma	211
Zuweisung, kombiniert	55	Prompt	26
Zuweisung, verknüpfende	53	Protokoll	131
Option	30	Prozedurale Programmierung	211
OR	262	Punkt-Notation (Datenbank)	248

Q

Quelltext	25
Query-String	152

R**RDBMS**

Siehe Relationales Datenbankmanagementsystem

readline()	44
Regulärer Ausdruck	309
Relationales Datenbankmanagementsystem	233
Request	129, 140
require	176
Response	129, 140
Rückgabewert	106

S

Schleife	85
Schlüssel-Wert-Paar	59
SELECT	250
Semantische Strukturierung	15
Server	131
serverseitig	17
Session	202
starten	202
wieder aufnehmen	203
Session-ID	203
Sichtbarkeit	215, 217, 218
Signatur	108
Sitzung	
<i>Siehe Session</i>	
Skalare Funktion (SQL)	265
Datum	267
numerische	267
Zeichenketten	266
Skript	26
Skriptsprache	23, 26
Spaghetti-Code	212
Spaltenoption (Datenbank)	244
Spaltentyp (Datenbank)	243
Sprache wechseln	199
SQL	233, 275
Kommentare	241
SQL-Befehlskategorien	234
SQL-Injection	285
Standardsoftware	310

stateless	145
Statement	
<i>Siehe Anweisung</i>	
String	48, 51
Superglobals	142
Syntactic sugar	54
Syntax	98
Syntaxfehler	97

T

Tabelle (Datenbank)	231, 243
anlegen	243
löschen	247
Tag	39
Timestamp	
<i>Siehe Zeitstempel</i>	
try-catch	226
Type Casting	64
Type Hints	125
Type Juggling	63
Typisierung	
dynamische	49
statische	50

U

UNIQUE	270
Unit-Test	311
Unixzeit	118
UNSIGNED	245
UPDATE	251
URL	14, 129
URL-Parameter	153

V

Validierung	155
var_dump()	48
Variable	43, 46
Bezeichner	47
Geltungsbereich	126
Vererbung	221
Vergleichsoperatoren (SQL)	261
Verkettung	292
Versionsverwaltung	310
Verzweigung	75
Visual Studio Code	36
void	112
Volltextsuche	271
Vorgabewert (Datenbank)	245

W

Wahrheitswert	
<i>Siehe Boolean</i>	
Web-API	20, 183
Webhoster	146
Webhosting	259
Web Scraping	182
Webseite	
dynamische	14
statische	14
Webserver	14, 131, 133, 137
Webservice	18
Webpace	146
WHERE-Klausel	251, 260
while	91
Wildcard	262

X

XAMPP	309
XSS	
<i>Siehe Cross-Site-Scripting</i>	

Z

Zählschleife	88
Zeichenkette	
<i>Siehe String</i>	
Zeitstempel	118
Zeitzone	121
Zugriffsrechte	190
Unix	191
Zuweisung	44