# 1 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*.

articles.txt — Bearbeitet

```
Die Zeitung – Neueste Nachrichten
---------------------------------

- »PHP & MySQL Schnelleinstieg« erschienen! Mehr unter https://www.php-mysql-schnelleinstieg.de
- ElePHPant entdeckt! Mehr unter https://www.php.net/elephpant.php
- PHP 8 veröffentlicht! Mehr unter https://www.php.net/releases/8.0/de.php
```

**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.

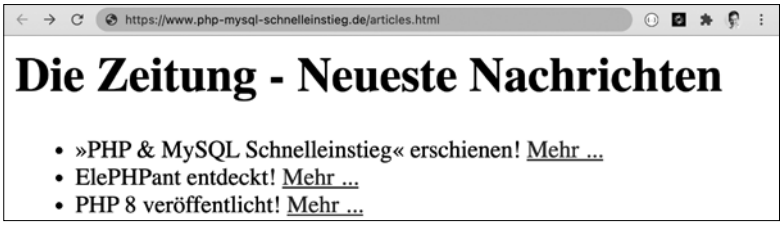


**Abb. 1.4:** Statische HTML-Webseite articles.html im Browser

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.

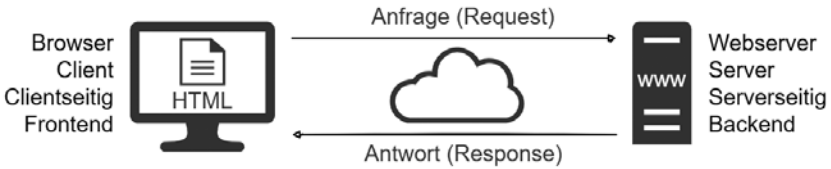


**Abb. 1.5:** Der Kreislauf aus HTTP-Anfrage und -Antwort

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` = **d**ay.**m**onth.**Y**ear = 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.

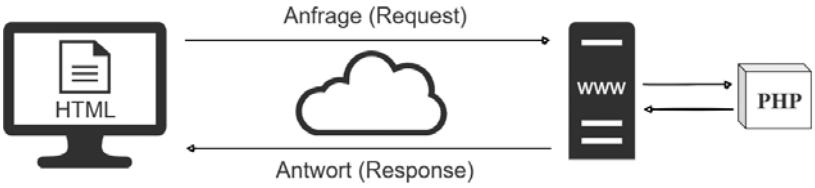


**Abb. 1.6:** Der Kreislauf aus Anfrage und Antwort mit PHP

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 »Nachrichten-Ticker« 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 |
| `$articles = ...` | Definiert die Variable `$articles` – eine Art Daten-Container für die Zeitungsartikel. Die Variable ist eine Referenz zum späteren Zugriff auf die Daten. |
| *`Abruf der Artikel aus Datenbank`* | 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 Verwendungszeck.

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 abruft.

Ein Webservice wird oft auch technischer als *Web-API* (Web Application Programming Interface) bezeichnet – eine Programmier-Schnittstelle 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-

hungsweise *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

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

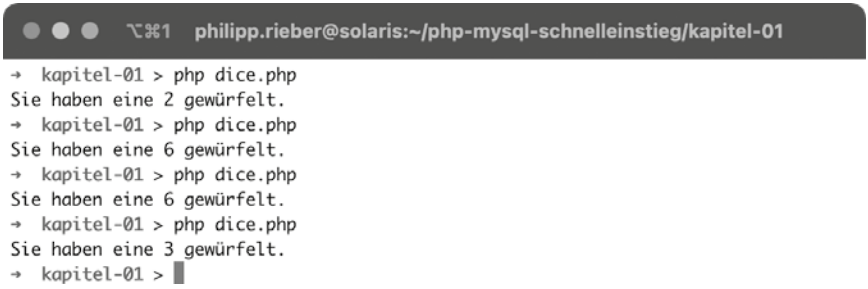


**Abb. 1.7:** Wiederholte Ausführung von dice.php

### 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.

# 2 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.

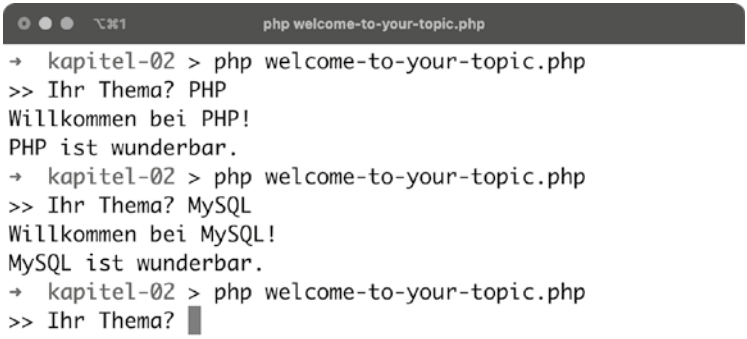


**Abb. 2.1:** Die Benutzereingabe beeinflusst dynamisch die Ausgabe

### 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 = '...'; // Enhält unzulässiges Zeichen
```

Für eine gute Lesbarkeit empfiehlt sich die Verwendung einer einheitlichen Schreibweise der Variablen-Bezeichner, zum Beispiel der sogenannten *camelCase*-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