

Konzepte des **skriptsprachenorientierten Programmierens**

Studienfach: Wirtschaftsinformatik

Prof. Dr. Fahri Yetim



Kurzvita

Studium der Informatik (FH Konstanz) und
Informationswissenschaft (Uni Konstanz)

Promotion an der Uni Konstanz. Thema der Dissertation:
„Erklärungen im Kontext der Mensch-Computer-Interaktion:
Ein Ansatz zur Integration der Methoden von Hypertext und
Künstlicher Intelligenz“

Systementwickler in Konstanz und Hamburg

DAAD-Dozentur in Istanbul

DAAD Stipendiat an der Uni Konstanz

Gastprofessur am New Jersey Institute of Technology:

Vertretungsprofessur an der FH Köln;

Senior Researcher an der Uni Siegen

Habilitation an der Oulu University in Finnland. Thema:
„Human-Centered Information Systems Design, in particular
Value Sensitive and Motivational Design“

**Professur für Wirtschaftsinformatik an der FOM seit 2014
(Studienzentrum Köln)**

Forschungsschwerpunkte

Mensch-Computer-Interaktion;
Persuasive Technologien und Verhaltensänderung;
Wertebasierte Analyse, Design & Evaluation von
Informationssystemen;
Kommunikations- und Partizipationsunterstützungssysteme
Personalisierung von Information, Wissensmanagement

Kurze Vorstellung der Teilnehmer

- Name
- Tätigkeit
- Erfahrung mit dem Thema

1 Einleitung: Modulziele, Prüfungsform & Organisatorisches

2 Einführung in Skriptsprachen

3 Einführung in PHP

4 Datentypen

5 Operatoren

6 Kontrollstrukturen

7 Funktionen

8 Objektorientierung

9 Ein- und Ausgabe (Dateien)

10 Web-Formularanbindung

11 Datenbankanbindung

12 Typische Anwendungen

Studierenden können nach erfolgreichem Abschluss des Moduls:

- die Eigenschaften der Programmierung von Sprachen der vierten Generation erklären,
- diese, gefestigt durch die praktischen Übungen, in Form der Lösung von programmiertechnischen Problemstellungen anwenden,
- erkennen, dass die Sprachen (beispielsweise PHP, Perl, Python, Ruby usw.) typischerweise interpreterbasiert sind und eine oft ausrichtungstypische bemerkenswerte Sprachmächtigkeit aufweisen,
- skriptsprachenorientierte Programmietechnik von den elementaren prozeduralen oder objektorientierten Programmiersystemen unterscheiden,
- die Vorteile skriptorientierter Sprachkonzepte herausstellen – z.B. oft flexible Sprachkonzepte bzw. in Form von Bibliotheken in der Regel mögliche Konstrukte auch für komplexe Probleme,
- die Potenz der skriptorientierten Systeme erklären (hohe Entwicklungsperformanz und Plattformunabhängigkeit) und die in der Regel untergeordnete Bedeutung der Ausführungsperformanz nachvollziehen,
- die Sprachmächtigkeiten darstellen und geschickt nutzen,
- anhand von elementaren und universellen Beispielen zeigen, wie die skriptorientierte Programmierung für Aufgaben aus dem administrativen Bereich bis hin zu hochkomplexen Applikationen sinnvoll eingesetzt werden kann.

Lehrmethodik

- Vorlesungen
- Themenbezogene Diskussionen
- Übungen und Fallstudien
- Hausaufgaben
- Online-Campus

Teilnahmevoraussetzungen und Vorkenntnisse

Die Teilnahme an folgenden Modulen wird empfohlen:

- Konzepte des prozeduralen Programmierens
- Konzepte des objektorientierten Programmierens

Prüfung und Benotung

- Klausur über 90 Minuten (90 Punkte)

- Im Rahmen der Klausur wird eine praktische Aufgabenstellung, die zuvor zur Verfügung gestellt wurde, abgeprüft.

- **Optional:** Bearbeitung eines Projektes (1-2 Personen):
 - Max. 25 Punkte für die Implementierung und Präsentation von Teilaufgaben (Komponenten) eines Projektes.
 - Die erworbenen Punkte gelten auch für die Nachschreibeklausuren.
 - Projektideen mit **Erwartungen und Voraussetzungen** werden zum Zeitpunkt der Behandlung des Kapitels „Typische Anwendungen“ online zur Verfügung gestellt. Studierende können auch eigene Projektideen vorschlagen.

Basisliteratur für die Veranstaltung

1. Florence Maurice. PHP 7 und MySQL: Ihr praktischer Einstieg in die Programmierung dynamischer Websites, dpunkt, 2018.
2. Thomas Theis. Einstieg in PHP 7 und MySQL. Rheinwerk Computing, 2018

Copyright-Hinweis: Die beiden Fachbücher decken die Themen der Vorlesung ab. Viele der im Skript verwendeten Abbildungen und Programmcodes unterliegen dem Copyright von Verlagen bzw. den Autoren der Fachbücher. Ich bedanke mich herzlich bei den Verlagen dpunkt und Rheinwerk Computing für die Unterstützung.

Weiterführende Literatur

1. Larry Ullman. PHP Advanced and Object-Oriented Programming: Visual Quickpro Guide. Third Edition. Peachpit Press, 2013.
2. Larry Ullman. Effortless E-commerce with PHP and MySQL. Second Edition. New Riders, 2013.

Hilfreiche und interessante Seiten im Internet

1. PHP-Handbuch, <http://php.net/manual/de/>

Ablauf der Vorlesung:

1. Kurze Wiederholung des Stoffes aus der vorangegangenen Veranstaltung
2. Neuer Stoff
3. Übungsaufgaben (in Kleingruppen)
4. Besprechung der Übungsaufgaben
 - Präsentation Ihrer Lösung
 - Präsentation meiner Lösung
 - Abschließende Diskussion
5. Zusammenfassung

1 Einleitung: Modulziele, Prüfungsform & Organisatorisches

2 Einführung in Skriptsprachen

3 Einführung in PHP

4 Datentypen

5 Operatoren

6 Kontrollstrukturen

7 Funktionen

8 Objektorientierung

9 Ein- und Ausgabe (Dateien)

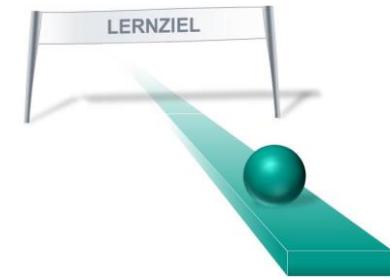
10 Web-Formularanbindung

11 Datenbankanbindung

12 Typische Anwendungen

Im Anschluss an diesen Themenblock sollen Sie:

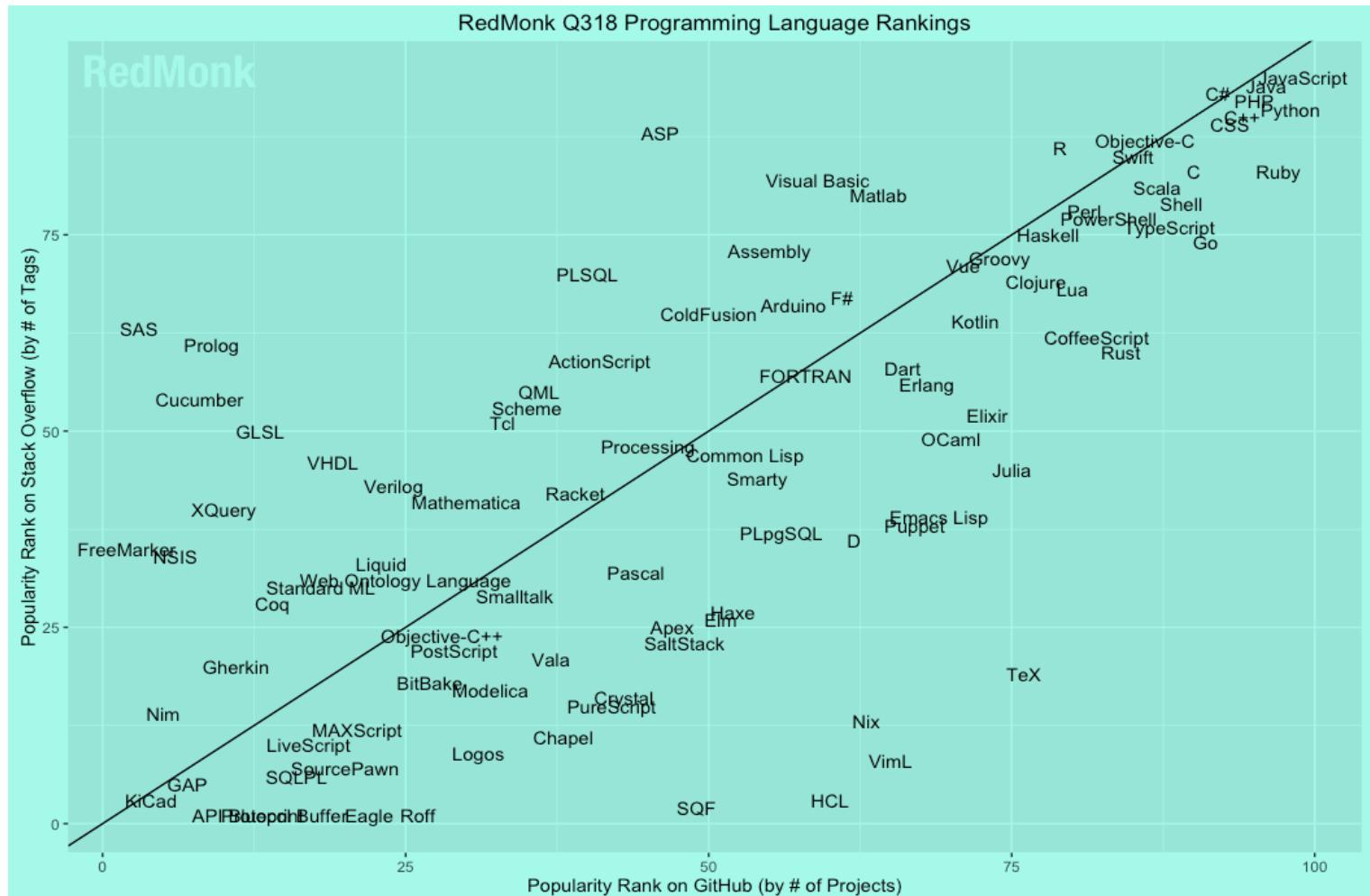
- Aktuelle Entwicklungen im Bereich Programmiersprachen kennen und wichtige Programmierparadigmen unterscheiden können
- Unterschiedliche Ansätze zur Übersetzung von Programmen kennen lernen (Compiler vs Interpreter)
- Merkmale von Skriptsprachen herausstellen können



RedMonk Language Ranking

RedMonk Top 10

1. Java
2. JavaScript
3. Python
4. PHP
5. C#
6. C++
7. CSS
8. Ruby
9. C
10. Objective-C

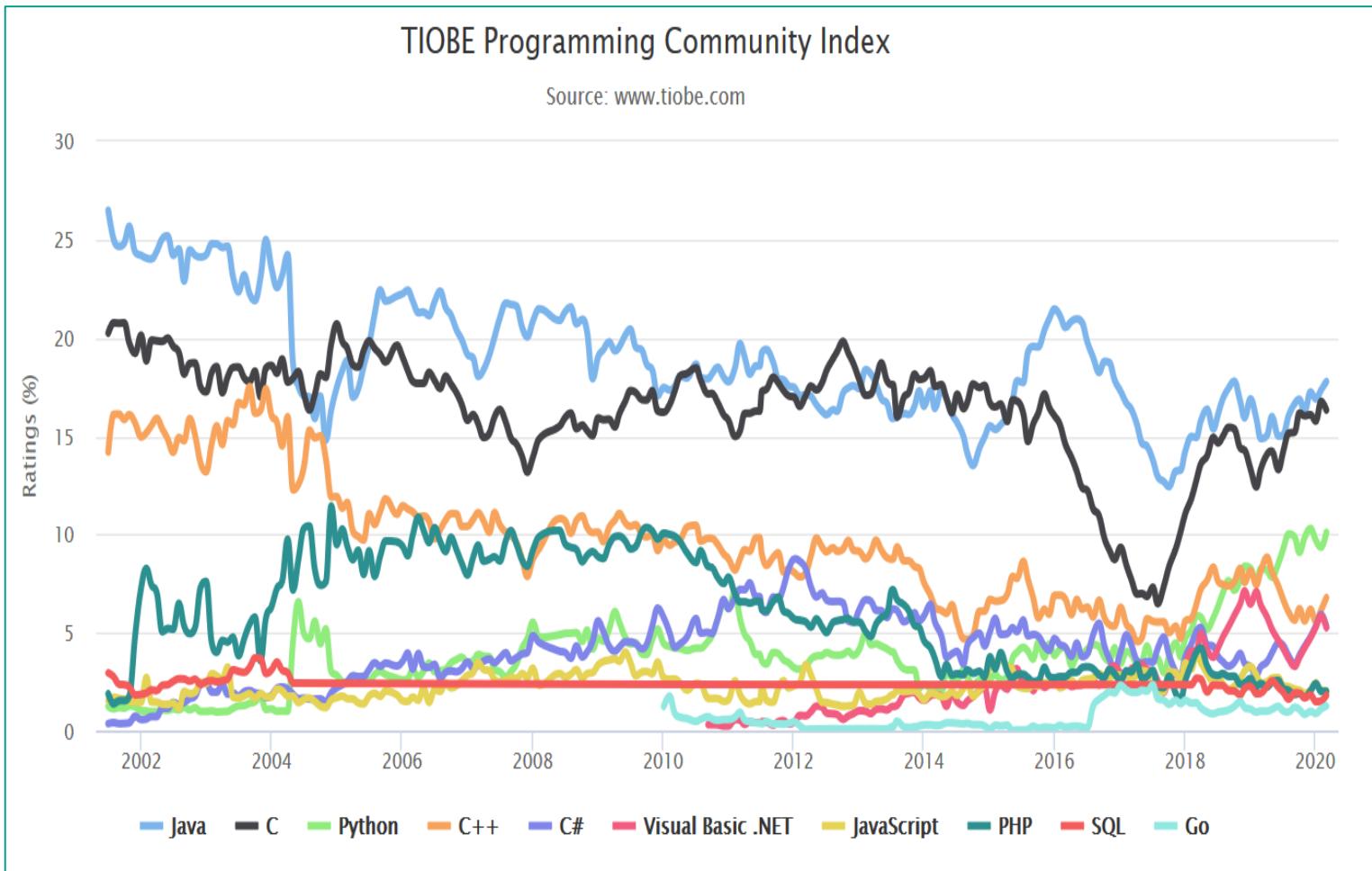


Quelle: <https://redmonk.com/sogrady/2018/08/10/language-rankings-6-18/> Zugriff 09.03.2020

TIOBE Programming Community Index

TIOBE-Index Top 10

1. Java
2. C
3. C++
4. Python
5. (Visual) Basic
6. C#
7. JavaScript
8. PHP
9. SQL
10. Objective-C



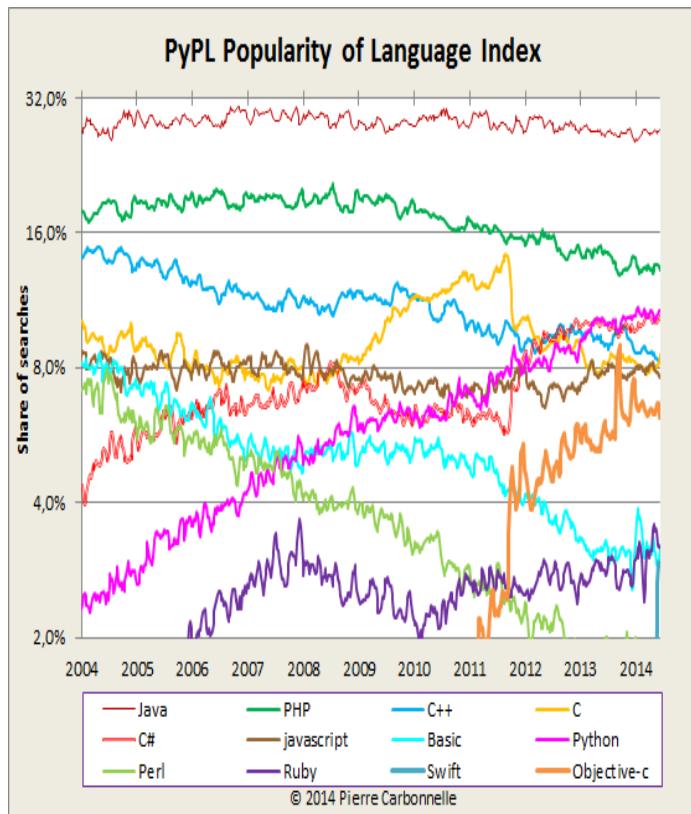
Quelle: <https://www.tiobe.com/tiobe-index/>

Zugriff: 09.03.2020

PYPL Popularity of Programming Language

PYPL-Index Top 10

1. Java
2. PHP
3. C#
4. Python
5. C++
6. C
7. JavaScript
8. Objective-C
9. Visual Basic
10. Ruby



Quelle: <https://sites.google.com/site/pydatalog/pypl/PYPL-Popularity-of-Programming-Language> Zugriff: 31.08.2014

The PYPL PopularitY of Programming Language Index is created by analyzing how often language tutorials are searched on Google.

März. 2020

Rank	Change	Language	Share	Trend
1		Python	30.09 %	+3.9 %
2		Java	18.84 %	-1.7 %
3		Javascript	8.1 %	-0.1 %
4		C#	7.27 %	-0.0 %
5		PHP	6.08 %	-1.1 %
6		C/C++	5.86 %	-0.2 %
7		R	3.73 %	-0.2 %
8		Objective-C	2.42 %	-0.5 %
9		Swift	2.28 %	-0.1 %
10		Matlab	1.89 %	-0.1 %

Quelle: <http://pypl.github.io/PYPL.html>
Zugriff: 09.03.2020

Definition

- Ein **Paradigma** ist eine grundsätzliche Denkweise.
- **Programmierparadigmen** reflektieren grundlegende Ansätze/Stile, wie man Computer durch Programmierung steuern kann.

Beispiel.: Wie wird ein Sachverhalt modelliert?

- Datenhaltung, Funktionen, Befehle (imperativ)
 - Durch Regeln, Fakten und Schlussfolgerungen (deklarativ / logisch)
 - Durch Klassen, Objekte, Methoden, Vererbung (objektorientiert)
-
- Der Programmierung liegen je nach der Programmiersprache verschiedene Prinzipien zugrunde.

Wichtige Programmierparadigmen

Imperative Programmierparadigmen

- Ein Programm besteht aus einer Folge von Befehlen, die vorgeben, *wie* das Programm seine Ergebnisse erzeugt.
- Strukturierte Programmierung (Pascal) / Prozedurale Programmierung (C)

Deklarative Programmierparadigmen

- Ein Programm gibt die gewünschten Ergebnisse (das *Was*) und ihre Bedingungen an. Der Lösungsweg wird bei der Übersetzung festgelegt.
- logische Programmierung (PROLOG), Datenbankabfrage (SQL)

Objektorientierte Programmierparadigmen

- Die Bausteine, aus denen ein objektorientiertes Programm besteht, werden als Objekte bezeichnet. Daten und Befehle in Objekten zusammengefasst.
- Java, C++

Weitere Paradigmen:

- komponentenorientiert, aspektorientiert, generisch, ...

=>Sprachen unterstützen i.d.R. mehrere Paradigmen gleichzeitig

Beispiel:

„In einer unter MS Access und mit VBA entwickelten Anwendung sind die funktionalen Komponenten *ereignis- und objektorientiert* angelegt (Bsp.: 'beim Open' von 'Formular X'). Der VBA-Code ist *strukturiert/modular/prozedural* (denn er besteht aus Modulen, Makros, Prozeduren etc.); und er ist gleichzeitig *imperativ*, weil er 'Befehle' enthält, die (innerhalb der Prozeduren ...) exakt in der codierten Folge ausgeführt werden. Formulare und Berichte sowie die SQL-Aufrufe sind *deklarativ*, weil der Entwickler hier nur das WAS und nicht das WIE festlegt.“

Quelle: <http://de.wikipedia.org/wiki/Programmierparadigma> Zugriff 31.07.2014

Compiler-basierter Ansatz

- Der Compiler übersetzt ein Programm (Quellcode) vor dessen Ausführung in Maschinencode

Charakteristiken:

- Programme haben schnelle Ausführungsgeschwindigkeit
- Programme ohne zusätzliche Software ausführbar
- Programme sind plattformabhängig (Betriebssystem, Hardware)
- Quellcode nicht offen

Beispiele:

- C, Pascal

Interpreter-basierter Ansatz

- Der Interpreter liest Quellcode zur Laufzeit des Programms ein und führt den Quellcode aus

Charakteristiken

- Programme haben langsame(re) Ausführungsgeschwindigkeit
- Programme leicht änderbar;
- Programme sind plattformunabhängig (Interpreter auf Plattform vorausgesetzt)
- Quellcode offen

Beispiele:

- PHP, Perl

Hybrider Ansatz: Bytecode-Interpreter

- Compiler übersetzt das Programm in einen Zwischencode (Bytecode)
- Interpreter („virtuelle Maschine“) liest Bytecode zur Laufzeit ein und führt den Code aus.

Charakteristiken

- Ausführungsgeschwindigkeit geringer im Vgl. mit reinem Compiler
- Bytecode ist plattformunabhängig (VM vorausgesetzt)
- Quellcode kann durch Dekompilierung (mehr oder weniger) offen gelegt werden

Beispiele:

- Java, C#

Hybrider Ansatz: Just-in-time Compiler

- Programm wird zur Laufzeit (teilweise) in Maschinencode übersetzt
- Compiler nutzt zur Laufzeit verfügbare Informationen zur Optimierung und zur Entscheidung, ob Programmteile kompiliert und ggf. für weitere Aufrufe gespeichert werden sollen.
- Einsatz meist innerhalb von virtuellen Maschinen

Charakteristiken

- Ausführungsgeschwindigkeit höher im Vgl. mit Interpreter
- dynamische Optimierungen möglich

Beispiele:

- Java VM Hotspot von Sun Microsystems
- JavaScript Engine Firefox Browser: JIT Compiler JaegerMonkey
- PHP Beschleuniger, HHVM

Skriptsprachen weisen oft die folgenden Merkmale auf

- Mehrheitlich Interpreter-basiert (Programm zur Laufzeit interpretiert),
- Quellcode ist sichtbar
- Plattformunabhängigkeit (Interpreter vorausgesetzt)
- Deklaration von Variablen ist meist nicht nötig
- Dynamische, automatische Typumwandlung (Interpreter ermittelt den Typ während der Laufzeit)
- Automatische Speicherverwaltung

Beispiele:

- PHP, JavaScript, Perl, Python, Ruby, Linux-Shell

Skripte (Skripts):

- Programme, die in Skriptsprachen geschrieben sind
- beinhalten Befehle in einer logischen, ausführbaren Reihenfolge.

Skriptsprachen im WWW

(a) Clientseitige Skriptsprachen

- Sprachen, die u.a. zum Erstellen webbasierter Programme verwendet werden, die auf Seite des Webbrowsers ausgeführt werden (Bestandteil von Dynamic HTML).
- Beispiel: JavaScript

(b) Serverseitige Skriptsprachen

- Sprachen, mit denen man Skripte schreibt, die auf dem Webserver ausgeführt (interpretiert) werden
- Beispiel: PHP, Perl

Einsatzgebiete

- Automatisierung von Administrationsaufgaben (Backup, Monitoring)
- Steuerung von Programmen (Shell-Skripte)
- Kommunikation zwischen Systemen
- Makros für Editoren, Anwendungsprogramme (Office, ...)
- Datenauswertung und Konvertierung
- Programmierung dynamischer Webseiten
- Administration von Datenbanken
- Erweiterung von Unternehmens-Anwendungen

1 Einleitung: Modulziele, Prüfungsform & Organisatorisches

2 Einführung in Skriptsprachen

3 Einführung in PHP

4 Datentypen

5 Operatoren

6 Kontrollstrukturen

7 Funktionen

8 Objektorientierung

9 Ein- und Ausgabe (Dateien)

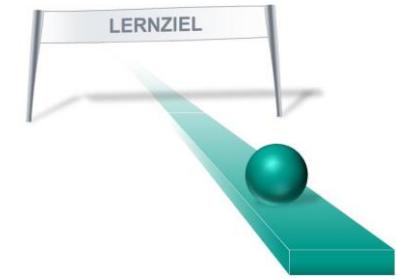
10 Web-Formularanbindung

11 Datenbankanbindung

12 Typische Anwendungen

Im Anschluss an diesen Themenblock sollen Sie:

- Die Vorteile von PHP benennen können
- Die grundlegende Arbeitsweise mit PHP kennen lernen
- Erste Erfahrungen mit PHP sammeln



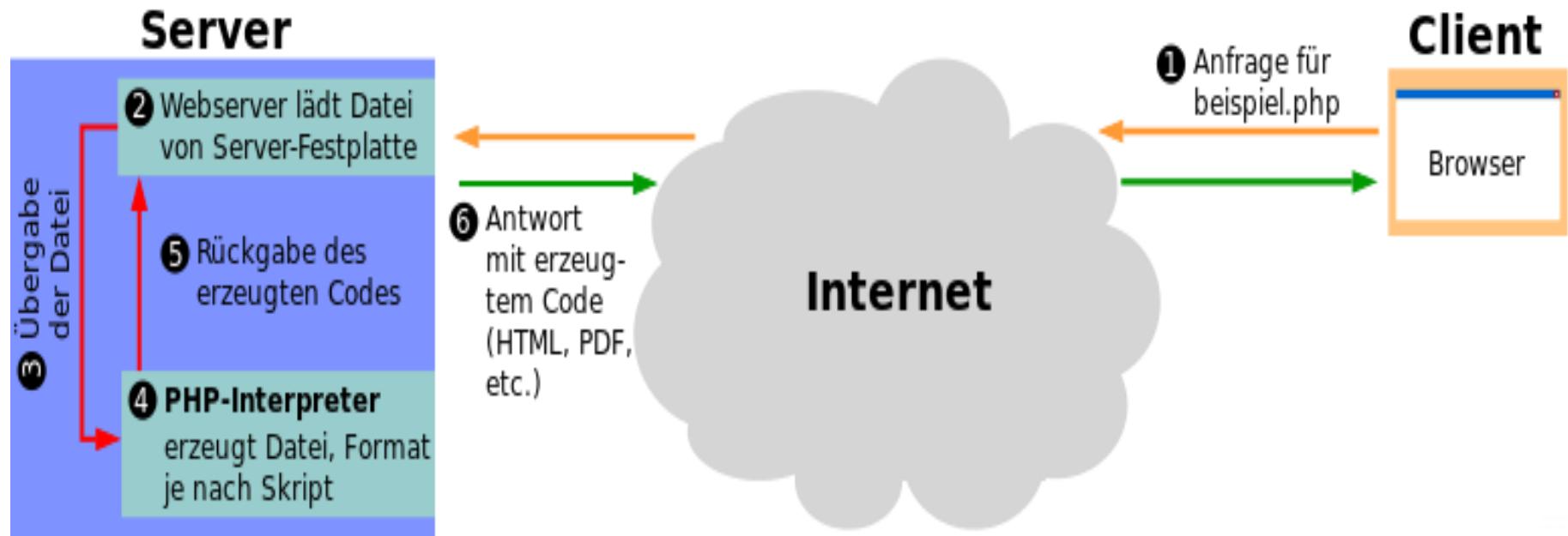
PHP hat viele Vorteile (Maurice, 2014):

- Es ist relativ einfach zu erlernen ... und trotzdem ausgereift: PHP liegt derzeit in **Version 7.x.** vor.
- Alles, was Sie zur Arbeit mit PHP brauchen, steht frei zur Verfügung. Im Internet finden Sie auch bei spezielleren Fragen Hilfe
- Es ist speziell für dynamische Webseiten entwickelt worden – das bedeutet, alle Funktionen sind genau darauf zugeschnitten.
- PHP ermöglicht das Arbeiten mit Datenbanken Dateien, aber auch vieles mehr (PDF-Erstellung, Bildbearbeitung)
- PHP kann sowohl prozedural als auch objektorientiert programmiert werden und ist damit auch für den Einsatz bei größeren Projekten geeignet.
- Viele bekannte Anwendungen basieren auf PHP, wie z.B.:
 - Content-Management-Systeme (WordPress, Joomla!, Drupal und TYPO3, ...).
 - Customer-Relationship-Management-Systeme (SugarCRM, ...)
 - E-Commerce-Applikationen (osCommerce, ...).
 - ...

PHP – Historie

- Entwickelt 1994/95 von Rasmus Lerdorf: „Personal Home Page Tools“
- Später umbenannt in „PHP: Hypertext Preprocessor“
- Skriptsprache mit einer an C und Perl angelehnten Syntax
- Objektorientiert seit Version 5.0
- Aktuelle **Version 7.x**
- Haupteinsatzgebiet: Erstellung von dynamischen Webseiten oder Webanwendungen
- Homepage: <http://php.net/>

Funktionsweise in Client-Server-Umgebung



Voraussetzungen für PHP-Programmierung

a) PHP-Interpreter

- XAMPP (<http://www.apachefriends.org/de/index.html>)
- eine Distribution von Apache, MySQL, PHP

b) Editoren (Windows)

- Notepad ++
- PHPEd
- Ultraedit
- PHPStorm IDE (kommerziell)
- Zend Studio (<http://www zend com/>)
- NetBeans IDE (<https://netbeans org/>)
- **Eclipse for PHP Developers (empfohlen)**

1. Server und PHP installieren

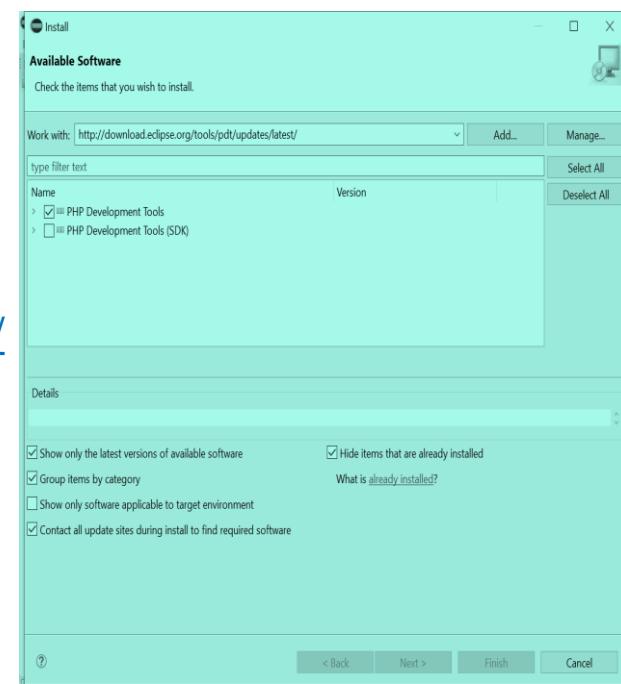
- XAMMP installieren: www.apachefriends.org/de/index.html.
- Installationsanweisungen: s. Homepage des Servers:

2. Installation von Entwicklungsumgebung:

2.1 Eclipse installieren: <http://www.eclipse.org/downloads/>

2.2 PHP IDE (ein Plugin) über Eclipses Plugin Manager installieren

- In Eclipse, click *Help -> Install New Software* und work with: <http://download.eclipse.org/tools/pdt/updates/latest/>
- Check die “**PHP Development Tools**” und click “Next”, um die Installation zu starten



Hello World in PHP!

(1) Neue Skript-Datei mit Dateiendung „.php“ erstellen

Inhalt: <?php **echo** "Hello World"; ?>

(2) Skript ausführen: Aufruf des PHP-Interpreters (php) mit Angabe der Quelldatei.

```
<?php  
  
echo 'Hallo Welt!'; // Dies ist ein einzeiliger Kommentar  
  
?>  
  
<?php  
  
/* Dies ist ein mehrzeiliger  
Kommentar  
  
*/  
  
echo 'Hallo Welt!';  
  
?>  
  
Doc-Kommentare: /** Das ist ein Doc-Kommentar .... */
```

Variablen

- müssen nicht deklariert werden

Variablennamen

- beginnen mit einem **\$-Zeichen**
- enthalten keine Zahl am Anfang
- keine Sonderzeichen bis auf _

Beispiele:

`$VARIABLE`

`$_variable`

`$variable`

`$Variable`

Bei Variablen

- wird zwischen Groß- und Kleinschreibung unterschieden

Schlüsselwörtern (*echo, while*)

- keine Berücksichtigung von Groß-/Kleinschreibung

Beispiele:

echo "Hallo Welt";

EchO "Hallo Welt";

Welche Variablennamen sind korrekt?

1. Variable
2. \$variable2
3. \$2variable
4. \$variable
5. \$_Variable
6. \$variable!

Den Inhalt von Variablen ausgeben

```
$name = "Lola";  
$alter = 2;  
echo "$name ist $alter Jahre alt." //Lola ist 2 Jahre alt
```

Echo versus Print:

- Ob Sie echo oder print wählen, ist im Wesentlichen Geschmackssache.
- Es gibt allerdings kleinere Unterschiede, die im Normalfall nicht relevant sind: So gibt print einen Rückgabewert zurück, echo hingegen nicht.

Variablennamen über {} kennzeichnen

Wie kann man direkt an den Wert etwas dranhängen, z.B. ein Genitiv-s (Aminas Jacke)?

```
$vorname= "Amina";  
echo "{$vorname}s Jacke"; // Ausgabe: Aminas Jacke
```

Variable Variablen

Variablennamen können selbst in Variablen gespeichert werden (zwei Dollarzeichen)

```
$varname = "beispiel";  
$$varname = "php";  
echo $beispiel; // Ausgabe: "php".
```

Konstanten werden definiert über

- die Funktion **define()** oder
- das Schlüsselwort **const**

```
const gruss = "Guten Morgen";  
echo gruss;
```

```
define("MAXWERT", 10);  
echo MAXWERT;           // gibt 10 aus
```

Aufgabe 3.1:

Erstellen Sie ein Skript, in dem Sie mehrere Variablen für Ihren Vornamen, Ihren Nachnamen und Ihren Wohnort definieren. Lassen Sie dann »X Y wohnt in Z« ausgeben.

Beispiel:

```
<html>
    <head>
        <title>Beispiel</title>
    </head>
    <body>
        <?php
            echo "Hallo, ich bin ein PHP-Skript!";
        ?>
    </body>
</html>
```

Kürzere Version:

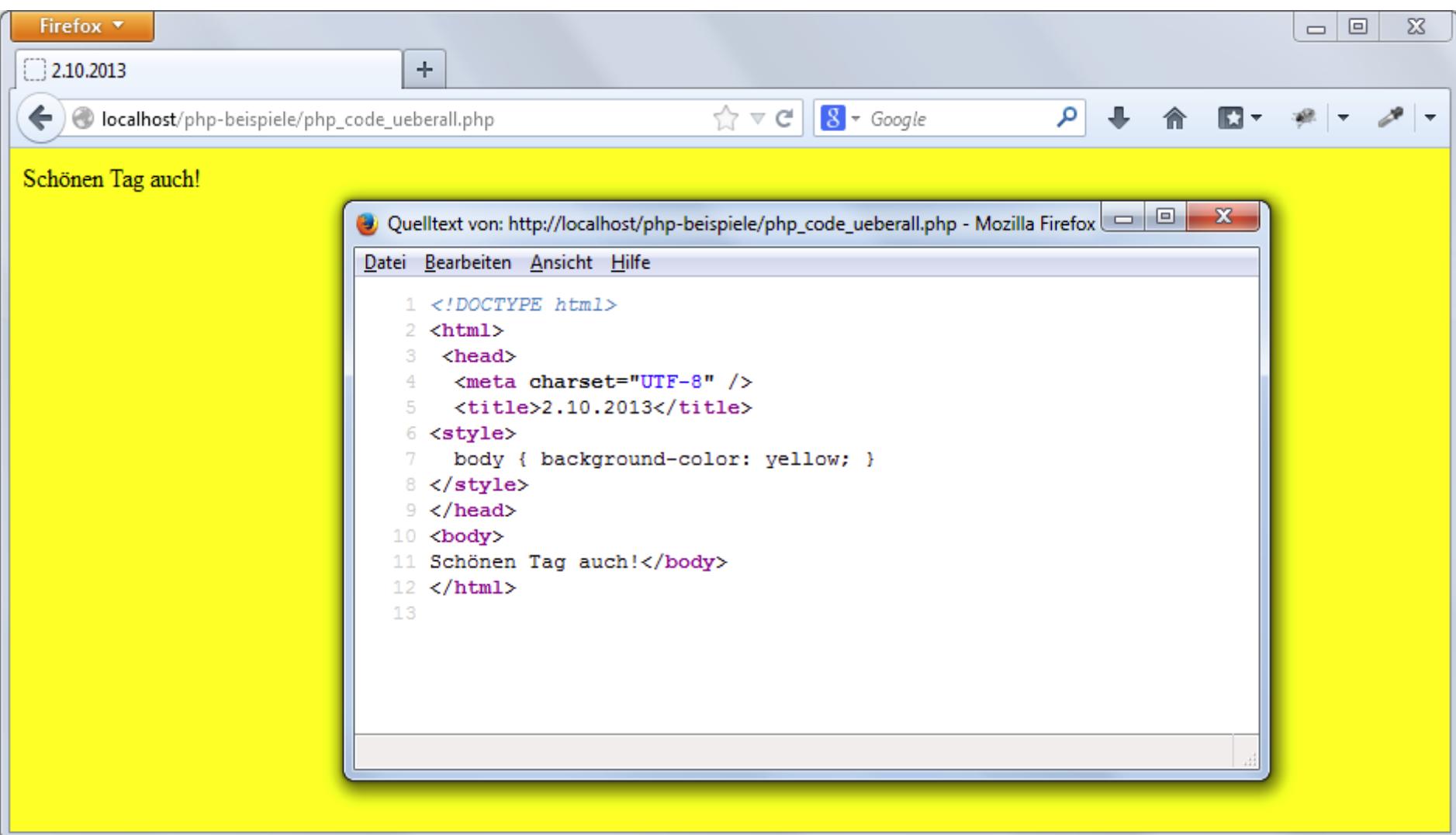
```
<?php ... ?>
```

Längere Version:

```
<script language="php"> PHP-Anweisungen </script>
```

Die PHP-Befehle können Sie an beliebigen Stellen in Ihrem HTML einfügen

```
01 <?php date_default_timezone_set("Europe/Berlin") ;?>
02 <!DOCTYPE html>
03 <html>
04 <head>
05     <meta charset="UTF-8" />
06     <title><?php echo date("j.n.Y") ; ?></title>
07 <style>
08     body { background-color: <?php echo "yellow"; ?>; }
09 </style>
10 </head>
11 <body>
12 <?php
13     echo "Schönen Tag auch!";
14 ?>
15 </body>
16 </html>
```



Mit Backslash maskieren

Möchten Sie z.B. innerhalb von doppelten Anführungszeichen wirklich ein Dollarzeichen ausgeben lassen, müssen Sie es *maskieren*

```
echo "Das Buch kostet 14 \$"; /* Dollarzeichen, keine Variable */
```

Das werden Sie häufig bei Attributwerten in HTML brauchen, die selbst in Anführungszeichen geschrieben werden:

```
echo "<img src=\"wiesen.jpg\" width=\"137\" height=\"103\"  
alt=\"Landschaft\" />";
```

ergibt als HTML-Code

```

```

Oder: echo "<img src='wiesen.jpg' width='137' height='103' "

```
alt='Landschaft' />";
```

\n und \t für einen übersichtlichen HTML-Quellcode verwenden

```
echo "Unser erstes \nphp-Dokument. \n"; // \n für einen Zeilenumbruch,  
echo "\tUnser erstes \tphp-Dokument. \n"; // \t für einen Tabulator
```

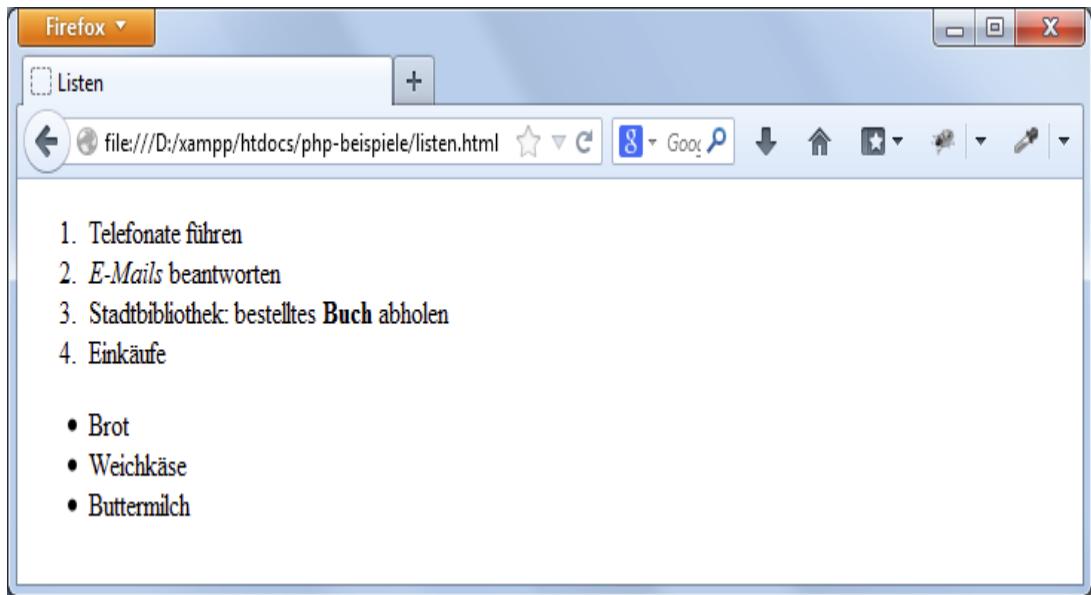
Alle möglichen Escapesequenzen

(i.e. die Kombination von Backslash plus Zeichen)

Kombination	Bedeutung
"\\"	\
"\n"	Neue Zeile
"\t"	Tabulator
"\\$"	Dollarzeichen
"\""	"
"\r"	Wagenrücklauf
"\v"	Vertikaler Tabulator
"\f"	Seitenvorschub
"\100"	Das Zeichen, das der angegebenen Oktalzahl in der Codetabelle des Zeichensatzes entspricht – hier @
"\X40"	Das Zeichen, das der angegebenen Hexadezimalzahl in der Codetabelle des Zeichensatzes entspricht – hier @
"\\\"	\
"\\."	.

//LISTEN

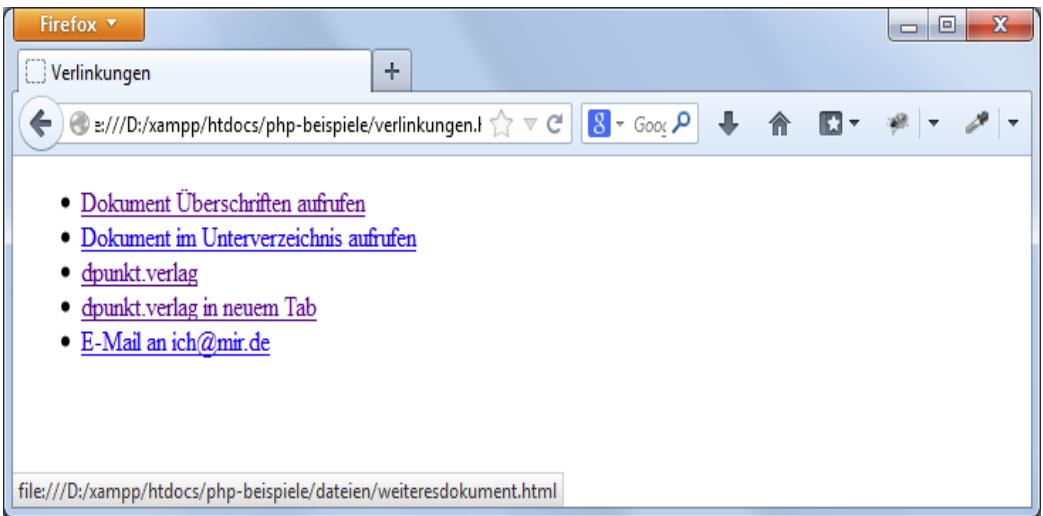
```
01 <!DOCTYPE html>
02 <html>
03 <head>
04 <meta charset="UTF-8" />
05 <title>Listen</title>
06 </head>
07 <body>
08   <ol>          //ordered List
09     <li>Telefonate führen</li>
10     <li><em>E-Mails</em> beantworten</li>
11     <li>Stadtbibliothek: bestelltes <strong>Buch</strong> abholen</li>
12     <li>Einkäufe</li>
13   </ol>
14   <ul>          //unordered List
15     <li>Brot</li>
16     <li>Weichkäse</li>
17     <li>Buttermilch</li>
18   </ul>
19 </body>
20 </html>
```



HTML Basics: Verknüpfungen (Links und Bilder)

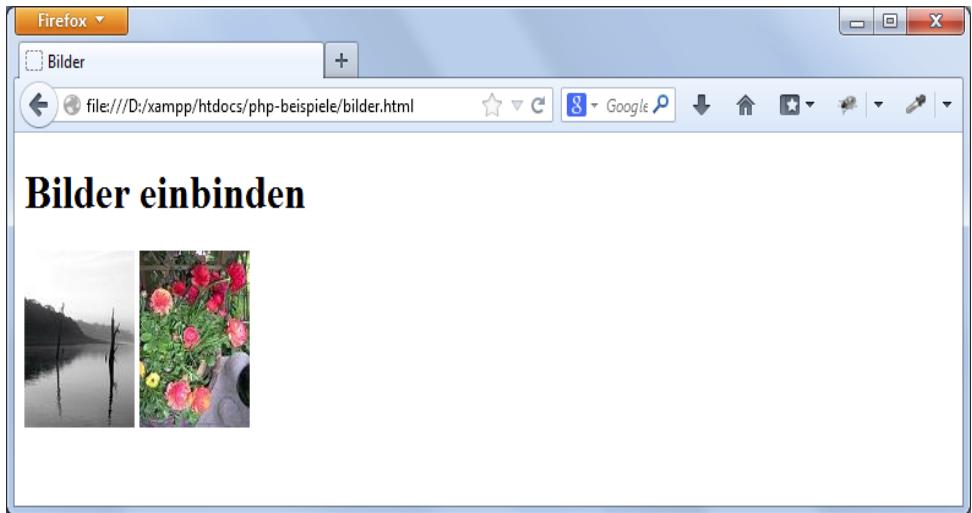
Eine Liste mit Links

```
01 <!DOCTYPE html>
02 <html>
03 <head>
04 <meta charset="UTF-8" />
05 <title>Verlinkungen</title>
06 </head>
07 <body>
08   <ul>
09     <li><a href="ueberschriften.html"> Dokument Überschriften
10       aufrufen</a></li>
11     <li><a href="dateien/weiteresdokument.html"> Dokument im
12       Unterverzeichnis aufrufen</a></li>
13     <li><a href="https://www.dpunkt.de/"> dpunkt.verlag </a></li>
14     <li><a href="https://www.dpunkt.de/" target="_blank">
15       dpunkt.verlag in neuem Tab </a></li>
16   </ul>
17 </body>
18 </html>
```



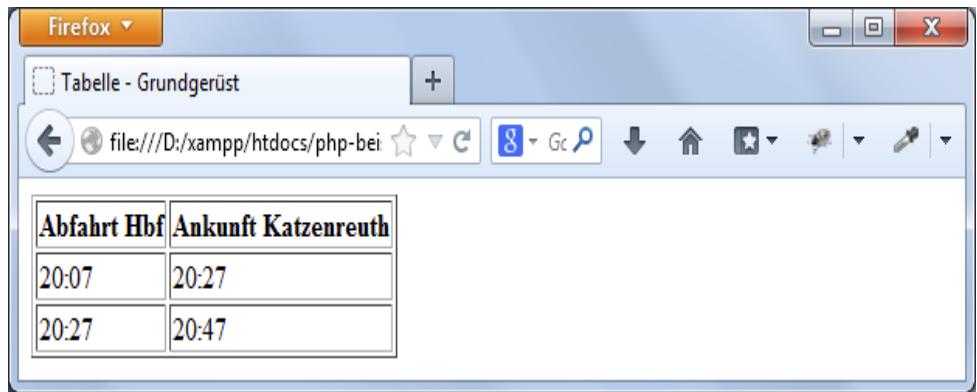
Bilder einbinden

```
01 <!DOCTYPE html>
02 <html>
03 <head>
04 <meta charset="UTF-8" />
05 <title>Bilder</title>
06 </head>
07 <body>
08   <h1>Bilder einbinden</h1>
09   
10   
11 </body>
12 </html>
```



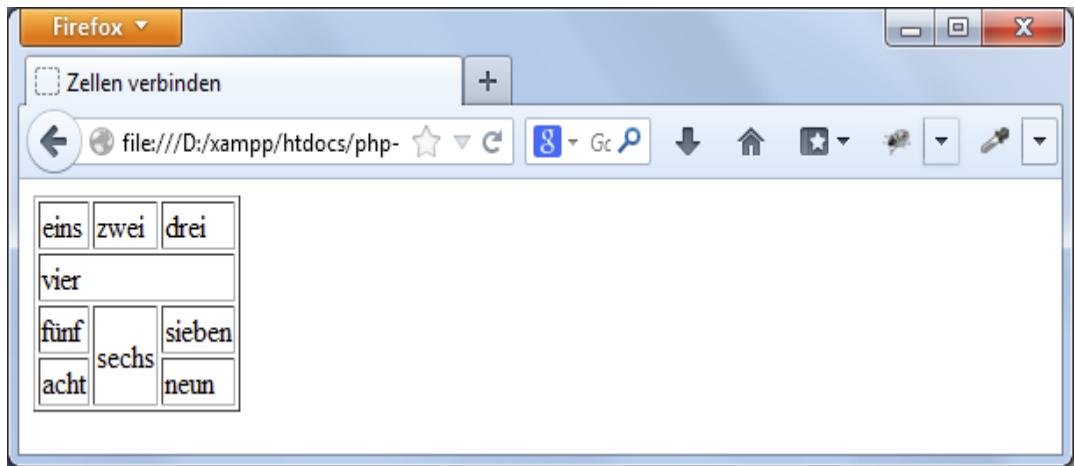
Eine zweispaltige Tabelle mit drei Zeilen

```
01 <!DOCTYPE html>
02 <html>
04 <head>
05 <meta charset="UTF-8" />
05 <title>Tabelle - Grundgerüst</title>
06 </head>
07 <body>
08   <table border="1">
09     <tr>
10       <th>Abfahrt Hbf</th><th>Ankunft Katzenreuth</th>
11     </tr>
12     <tr>
13       <td>20:07</td><td>20:27</td>
14     </tr>
15     <tr>
16       <td>20:27</td><td>20:47</td>
17     </tr>
18   </table>
19 </body>
20 </html>
```



Zellenverbinden

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<title>Zellen verbinden</title>
</head>
<body>
    <table border="1">
        <tr>
            <td>eins</td><td>zwei</td><td>drei</td>
        </tr>
        <tr><!--colspan verbindet Spalten -->      // <!-- Kommentar      -->
            <td colspan="3">vier</td>          // Zelle vier erstreckt sich über
        </tr>                                // drei Spalten
        <tr><!--rowspan verbindet Zeilen -->
            <td>fünf</td><td rowspan="2">sechs</td><td>sieben</td>
        </tr>                                // Zelle sechs erstreckt sich über
        <tr>
            <td>acht</td><td>neun</td>
        </tr>
    </table>
</body>
</html>
```



Aufgabe 3.2:

Erstellen Sie ein neues HTML-Dokument mit folgendem Inhalt:

- einer Überschrift (h1)
- einem Absatz
- einem Bild
- und einer ungeordneten Liste
- Außerdem soll ein Link dabei sein, der zu <http://php.net/> führt.

1 Einleitung: Modulziele, Prüfungsform & Organisatorisches

2 Einführung in Skriptsprachen

3 Einführung in PHP

4 Datentypen

5 Operatoren

6 Kontrollstrukturen

7 Funktionen

8 Objektorientierung

9 Ein- und Ausgabe (Dateien)

10 Web-Formularanbindung

11 Datenbankanbindung

12 Typische Anwendungen

Datentypen

Ein Datentyp beschreibt die Art der Informationen, die eine Variable oder ein PHP-Element enthalten kann. PHP unterstützt Datentypen für:

- Zahlen (Integer und Float: ganze Zahlen und Fließkommazahlen)
- Wahrheitswerte (der boolische Typ)
- Zeichenketten (Strings)
- Felder (Arrays: ein- und mehrdimensionale Felder von Variablen)
- Objekte (Klassen)
- Spezielle Typen: NULL und Resource Type

Unterschied zu vielen anderen Sprachen:

- Die Datentypen werden in PHP üblicherweise nicht vom Programmierer explizit gesetzt, sondern von PHP aus dem Kontext erkannt.
- Eine Variable kann ihren Datentyp innerhalb eines Programms wechseln.

Integer

Ganze Zahlen können positiv oder auch negativ sein.

```
$ganzezahl = 42;  
$nocheine = -13;
```

Zahlen definieren, die eine andere Basis als 10 haben. Bei Oktalzahlen (Basis 8) wird eine 0 vorangestellt, bei Hexadezimalzahlen (Basis 16) eine 0x:

```
$oktal = 012;      // entspricht 10  
$hexadezimal = 0xFF; // entspricht dezimal 255; für Farben verwendet
```

Float

Fließkommazahlen werden mit einem Punkt geschrieben:

```
$float = 1.5;
```

Ebenfalls möglich ist die wissenschaftliche Schreibweise für Fließkommazahlen:

```
$a = 1.2e3; /* entspricht 1200 */  
$b = 7e-2; /* entspricht 0.07 */
```

Wahrheitswert

kann nur true (wahr) oder false (falsch) annehmen.

```
$regnen = true;
```

Ressource

beinhaltet eine Referenz auf eine externe Ressource, wie z.B. auf eine geöffnete Datei oder auf eine Verbindung zu einer Datenbank

NULL

repräsentiert eine Variable ohne Wert (keinen Wert zugewiesen oder explizit auf NULL gesetzt).

Weitere Datentypen: Arrays und Objekte (Klassen)

Zu Arrays folgt gleich mehr und genaueres zu Objekten später

Eine Variable kann innerhalb eines Skripts beliebig den Wert wechseln:

```
$a = "Hallo"; // String  
$a = 7; // Integer  
$a = 3.5; // Float
```

→ PHP führt Konvertierungen automatisch durch.

TypeCasting:

Man kann auch die Umwandlung direkt durchführen

```
$string = "22";  
$zahl = (int) $string;
```

mit `var_dump()` kann man den Inhalt der Variablen und den Typ ausgeben.

```
var_dump($zahl); // Ausgabe: int(22)
```

Aufgabe 4.1:

Welche Ausgaben werden durch das folgende Programm erzeugt?

```
$str1 = "10 Eier";
$str2 = "Schachtel mit 10 Eiern";
$str3 = "3.5 Äpfel";
$erg1 = $str1 + 2;
var_dump($erg1);
echo "<br />\n";
$erg2 = $str2 + 2;
var_dump($erg2);
echo "<br />\n";
$erg3 = $str3 + 2;
var_dump($erg3);
```

Erstellen von Arrays

Arrays können durch oder ohne das Schlüsselwort array() erstellt werden

```
$antworten = array("nie", "manchmal", "oft");  
$antworten = ["nie", "manchmal", "oft"];  
  
$werte = array(42, 66, 3.5, 55, 7); // Array für Zahlen  
$antworten = array("nie", "manchmal", "oft", 42); // Kombination  
  
echo $antworten[0]; // Ausgabe: nie  
echo $antworten[2]; // Ausgabe: oft  
  
$antworten[] = "aus Prinzip nicht"; // ein weiteres Element anhängen  
echo $antworten[4]; // Ausgabe: aus Prinzip nicht
```

Informationen über Arrays ausgeben lassen,
um sich bei der Programmierung einen schnellen Überblick zu verschaffen

`echo $antworten; // schreibt "Array" auf den Bildschirm`

`Print_r($antworten); // zeigt den Inhalt`

`var_dump($antworten); /zeigt auch die Datentypen an`

The screenshot shows a Firefox browser window. The address bar displays "localhost/php-beispiele/arrays_v.php". The main content area shows the output of the PHP code `var_dump($antworten);`. The output is:

```
array ( [0] => nie [1] => manchmal [2] => oft [3] => 42 [4] => aus Prinzip nicht )
```

Below the browser window, a separate window titled "Quelltext von: http://localhost/php-beispiele/arrays_v.php - Mozilla Firefox" shows the source code of the PHP file:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8" />
5 <title>Arrays</title>
6 </head>
7 <body>
8 Array
9 (
10     [0] => nie
11     [1] => manchmal
12     [2] => oft
13     [3] => 42
14     [4] => aus Prinzip nicht
15 )
16 </body>
17 </html>
```

The screenshot shows a Firefox browser window. The address bar displays "localhost/php-beispiele/arrays_r.php". The main content area shows the output of the PHP code `Print_r($antworten);`. The output is:

```
array(5) {
[0]=>
string(3) "nie"
[1]=>
string(8) "manchmal"
[2]=>
string(3) "oft"
[3]=>
int(42)
[4]=>
string(17) "aus Prinzip nicht"
}
```

Ausgabe und Bearbeitung aller Elemente eines Arrays mit *foreach*

```
foreach ($antworten as $aw) {           // as Schlüsselwort
    echo "$aw <br />";                  // $aw temporäre Variable
}
echo $aw;      // außerhalb der Schleife, zuletzt gespeicherter Wert: "aus Prinzip nicht"
```

Anzahl der Elemente eines Arrays zu ermitteln

```
$anzahl = count($antworten);
echo $anzahl; // 5
```

Assoziative Arrays

Bisher haben wir die einzelnen Elemente über Nummern angesprochen. Manchmal möchte man aber die Arrayelemente über Namen ansprechen.

- Solche Schlüssel-Wert-Paare kann man einsetzen, um z.B. Vorwahlnummern Städten zuzuordnen.

Beispiel: Zuordnung von Farben zu hexadezimalen Farbbezeichnungen in HTML

```
$farben = array ("rot" => "#FF0000", "grün" => "#00FF00", "blau" =>  
    "#0000FF");
```

Oder einzeln ohne das Schlüsselwort definieren

```
$farben["rot"] = "#FF0000";  
$farben["grün"] = "#00FF00";
```

```
$farben["schwarz"] = "#000000"; // Elemente nachträglich ergänzen  
  
echo $farben["rot"];           // einzelne Werte ansprechen
```

Wie wird innerhalb von doppelten Anführungszeichen der Wert von Variablen ausgegeben?

- Wenn der Schlüssel eine Zahl ist, kann man den Wert des Arrayelements problemlos ausgeben:

```
echo "Sag niemals $antwort[0];"
```

- Probleme mit einem String als Schlüssel

```
echo "die Farbe ist $farben["rot"]"; // geht nicht
```

```
echo "die Farbe ist $farben['rot']"; // geht nicht
```

- Geschweifte Klammer zur Klammerung des Ausdrucks verwenden

```
echo "die Farbe ist {$farben['rot']}"; // geht
```

- Verknüpfungsoperator einsetzen,

```
echo "die Farbe ist " . $farben["rot"]; // geht auch
```

- den Schlüssel ohne Anführungszeichen schreiben:

```
echo "die Farbe ist $farben[rot]"; // geht auch
```

Zweidimensionale Arrays haben zwei Indizes

```
<html> <body>
<?php
    // 1. Zeile und 2. Zeile
    $pers = array(array("Maier", "Hans", 6714, 3500),
                  array("Schmitz", "Peter", 81343, 3750));
    // 3. Zeile
    $pers[2][0] = "Mertens";
    $pers[2][1] = "Julia";
    $pers[2][2] = 2297;
    $pers[2][3] = 3621.50;
    echo "<table border='1'>";
    for($i=0; $i<3; $i++) {
        echo "<tr>";
        for($k=0; $k<4; $k++)
            echo "<td>" . $pers[$i][$k] . "</td>";
        echo "</tr>";
    }
    echo "</table>";
?>
</body> </html>
```

http://localhost/num_zweidim.php +

localhost/num_zweidim.php

Maier	Hans	6714	3500
Schmitz	Peter	81343	3750
Mertens	Julia	2297	3621.5

Ein weiteres Beispiel:

```
$bilder = array(  
    array("pfad" => "blumen.jpg",  
          "alt" => "rote Blumen",  
          "titel" => "Strauß aus roten Blumen"),  
    array("pfad" => "landschaft.jpg",  
          "alt" => "Landschaft",  
          "titel" => "Landschaft im Nebel"),  
    array("pfad" => "stadt_am_meer.jpg",  
          "alt" => "Häuser",  
          "titel" => "Griechische Häuser am Abend"),  
        array("pfad" => "strand.jpg",  
              "alt" => "Strand",  
              "titel" => "Strand mit Bergen"),  
    array("pfad" => "boot.jpg",  
          "alt" => "Boot",  
          "titel" => "Boot auf einem Felsen")  
);
```

```
echo $bilder[0]["pfad"]; // blumen.jpg
```

Aufgabe 4.2:

Ändern Sie das Beispiel zufallsbilder.php so ab, dass zufällig einer von mehreren Texten angezeigt wird. Dafür müssen Sie natürlich zuerst ein Array mit mehreren Strings definieren!

```
// zufallsbilder.php
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Zufallsbilder</title>
  </head>
  <body>
    <?php
      $bilder = array("blumen.jpg", "boot.jpg", "landschaft.jpg", "stadt_am_meer.jpg", "strand.jpg");
      $max = count($bilder) - 1;
      $zufallszahl = rand(0, $max);
      echo "<img src='\$bilder[\$zufallszahl]" height='200' width='150' />";
    ?>
  </body>
</html>
```

1 Einleitung: Modulziele, Prüfungsform & Organisatorisches

2 Einführung in Skriptsprachen

3 Einführung in PHP

4 Datentypen

5 Operatoren

6 Kontrollstrukturen

7 Funktionen

8 Objektorientierung

9 Ein- und Ausgabe (Dateien)

10 Web-Formularanbindung

11 Datenbankanbindung

12 Typische Anwendungen

Operatoren

Operatoren dienen dazu, Ausdrücke zu erzeugen, Berechnungen durchzuführen, Variablen Werte zuzuweisen, sinnvoll miteinander zu verbinden oder zu vergleichen.

- Zuweisungsoperatoren
- Arithmetische Operatoren
- Logische Operatoren
- Vergleichsoperatoren
- Zeichenverkettungsoperator

Zuweisungsoperatoren, arithmetische Operatoren

Zuweisungsoperatoren

```
$nummer = 5;  
$zeichen = "Guten Tag";
```

Arithmetische Operatoren

```
$a + $b // Plus  
$a - $b // Minus  
$a * $b // Mal  
$a / $b // Geteilt  
$a % $b // Modulo
```

```
++$a // Pre-Inkrement  
$a++ // Post-Inkrement  
--$a // Pre-Dekrement  
$a-- // Post-Dekrement
```

Logische Operatoren

```
$a && $b    // Logisches UND
$a ||       // Logisches ODER
!$a         // Logisches NICHT
$a xor $b   // Exclusive-ODER (entweder $a oder $b)
```

Vergleichsoperatoren

<	// kleiner	
<=	// kleiner oder gleich	
>	// größer	
>=	// größer oder gleich	
==	// gleich	
!= oder <>	// ungleich	
==	// Identität	TRUE: gleiche Werte und gleicher Typ
!=	// Nichtidentität,	TRUE, ungleiche Werte oder ungleicher Typ

Zeichenverkettungsoperator

Zeichenverkettungsoperator

```
$vorname = 'Peter';  
$nachname = 'Kropff';  
echo 'Ich bin '.$vorname.' '.$nachname;
```

Rangfolge der Operatoren

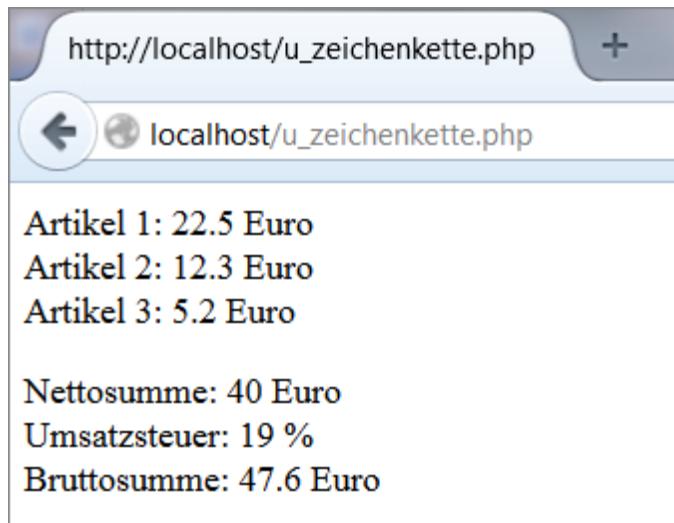
Operator	Assoziativität
! ++ --	rechts
* / %	links
+ - .	links
< <= > >=	keine Richtung
== != === !==	keine Richtung
&&	links
	links
= += -= *= /= .= %=	rechts
AND	links
XOR	links
OR	links

Aufgabe 5.1:

Berechnen Sie in einem PHP-Programm den Bruttoreis eines Einkaufs.

Es werden insgesamt drei Artikel eingekauft. Die Nettopreise der einzelnen Artikel betragen 22,50 €, 12,30 € und 5,20 €. Der Bruttoreis berechnet sich bekanntlich aus dem Nettopreis zuzüglich 19 % Umsatzsteuer. In die Berechnung muss also der Faktor 1.19 eingehen.

Die Ausgabe des Programms im Browser sollte wie folgt aussehen.



The screenshot shows a web browser window with the URL `http://localhost/u_zeichenkette.php` in the address bar. Below the address bar, there is a navigation bar with a back arrow, a refresh icon, and the URL again. The main content area displays the following text:
Artikel 1: 22.5 Euro
Artikel 2: 12.3 Euro
Artikel 3: 5.2 Euro

Nettosumme: 40 Euro
Umsatzsteuer: 19 %
Bruttosumme: 47.6 Euro

1 Einleitung: Modulziele, Prüfungsform & Organisatorisches

2 Einführung in Skriptsprachen

3 Einführung in PHP

4 Datentypen

5 Operatoren

6 Kontrollstrukturen

7 Funktionen

8 Objektorientierung

9 Ein- und Ausgabe (Dateien)

10 Web-Formularanbindung

11 Datenbankanbindung

12 Typische Anwendungen

Verzweigungen

dienen dazu, bestimmte Programmteile nur beim Vorliegen vorgegebener Bedingungen auszuführen.

- If
- switch

Schleifen

dienen dazu, das Programm anzuweisen, etwas mehrmals zu tun..

- while
- do-while
- for
- foreach

if

```
$i = 5;  
if ($i > 4) {  
    echo "$i ist größer als 4";  
    /* hier können weitere Anweisungen folgen */  
}
```

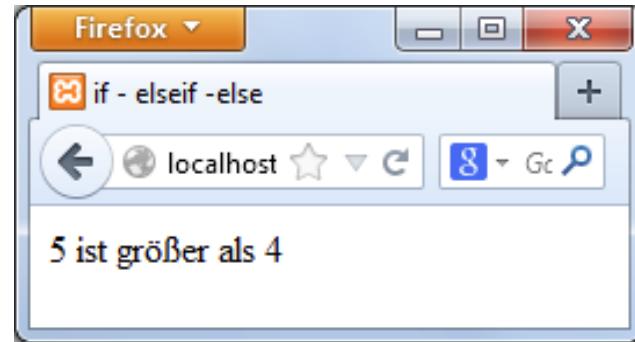
if - else

```
$i = 5;  
if ($i > 4) {  
    echo "$i ist größer als 4";  
    /* hier können weitere Anweisungen folgen */  
} else {  
    echo "$i ist nicht größer als 4";  
    /* weitere Anweisungen bei Bedarf */  
}
```

If-Abfrage

If - elseif - else

```
$i = 5;
if ($i > 4) {
    echo "$i ist größer als 4";
} elseif ($i == 4) {
    echo "$i gleich 4";
} else {
    echo "$i ist kleiner als 4";
}
```



Wechsel zwischen PHP- und HTML-Modus

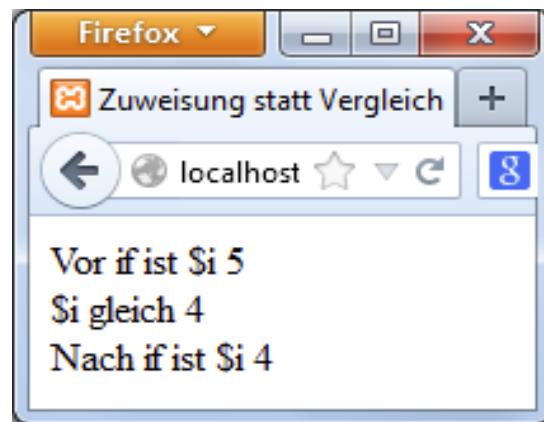
```
<?php
    $i = 5;
    if ($i > 4) {
        ?>
        <p>Die Bedingung ist wahr.</p>
        <?php
    } else {
        ?>
        <p>Die Bedingung ist nicht wahr.</p>
        <?php
    }
?>
```

Beachte: hier wird noch einmal in den PHP-Modus gewechselt, um die schließende Klammer zu setzen.

If-Abfrage

Beachte: Zuweisung anstelle von Vergleich

```
$i = 5;
echo "Vor if ist \$i $i<br />\n";
if ($i = 4) {
    echo "\$i gleich 4<br />\n";
}
echo "Nach if ist \$i $i<br />\n";
```



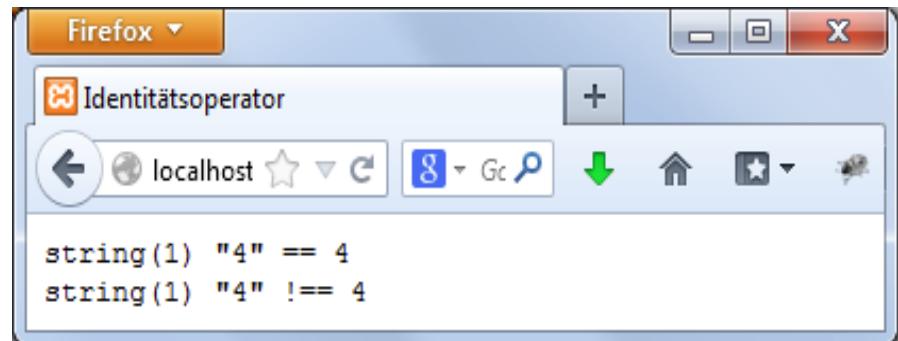
Folgende Werte werden als false behandelt:

- die Zahlen 0 und 0.0
- eine leere Zeichenkette "" oder auch eine Zeichenkette mit dem Element "0"
- ein Array ohne Elemente
- der spezielle Typ NULL
- und das Wort *false* selbst. Allerdings wird der String "false" selbst als true ausgewertet wie alle Strings, die nicht leer sind.

Alle anderen Ausdrücke werden zu true ausgewertet.

Beachte: Ist gleich-Zeichen: “==” vs. “===”

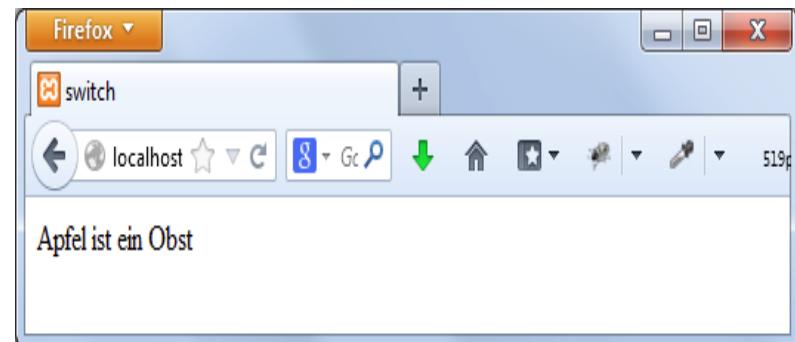
```
$i= "4";
if ($i == 4) {                                // automatische Typumwandlung
    var_dump($i);                            // Info über eine Variable: Typ & Wert
    echo " == 4<br />\n";
}
if ($i === 4) {                                // keine Typumwandlung
    var_dump($i);
    echo " === 4<br />\n";
} elseif ($i !== 4) {
    var_dump($i);
    echo " !== 4<br />\n";
}
```



Switch

Mehrere Fälle abdecken mit *switch*

```
$dasda = "Apfel";  
switch ($dasda) {  
    case "Apfel":  
        echo "$dasda ist ein Obst";  
        break;  
    case "Karotte":  
        echo "$dasda ist ein Gemüse";  
        break;  
    case "Käse":  
        echo "$dasda ist ein Milchprodukt";  
        break;  
    default:  
        echo "Kenne ich nicht";  
}
```



Aufgabe 6.1:

Erweitern Sie das folgende Programm mit den unterschiedlichen Begrüßungen je nach Uhrzeit so, dass immer der Seitenhintergrund unterschiedlich eingefärbt wird! Die Farbe des Seitenhintergrunds bestimmen Sie über CSS. Wenn Sie im head-Bereich Folgendes eingeben ...

```
<style>  
    body {  
        background-color: yellow;  
    }  
</style>
```

... wird beispielsweise der Hintergrund der Seite gelb.

→ Siehe das zu erweiternde Programm auf der nächsten Folie!

Aufgabe 6.1 (Fortsetzung):

```
// Das zu erweiternde Programm

<?php
date_default_timezone_set("Europe/Berlin");
$uhrzeit = date("H");
if ($uhrzeit < 5 || $uhrzeit > 20) {
    $gruss = "Gute Nacht";
} elseif ($uhrzeit < 11) {
    $gruss = "Guten Morgen";
} elseif ($uhrzeit < 15) {
    $gruss = "Guten Mittag";
} elseif ($uhrzeit < 18) {
    $gruss = "Guten Nachmittag";
} else {
    $gruss = "Guten Abend";
}
?>
```

```
// Fortsetzung

<!DOCTYPE html>

<html>
    <head>
        <meta charset="UTF-8" />
        <title>Unterschiedliche Begrüßung
        </title>
    </head>
    <body>
        <?php
            echo $gruss;
        ?>
    </body>
</html>
```

While

```
$i = 1;  
while ($i < 6) {  
    echo "hallo ";  
    $i++;  
}
```

Do-While

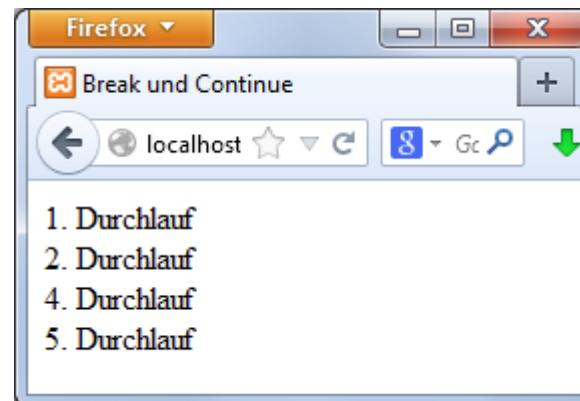
```
$i = 6;  
do {  
    echo "hallo ";  
    $i++;  
}  
while ($i < 6);
```

For

```
for ($i = 1; $i < 6; $i++) {  
    echo "$i. Hallo<br />\n";  
}
```

Schleife steuern über *break* und *continue*

```
for ($i = 1; $i <= 10; $i++) {  
    if ($i == 3) {  
        continue;  
    }  
    echo "$i. Durchlauf<br />\n";  
    if ($i == 5) {  
        break;  
    }  
}
```



foreach

es werden alle Elemente (oder Schlüssel-Wert-Paare) aus einem Array durchlaufen

- ***foreach (\$array_ausdruck as \$wert) Anweisung***

```
$stage = array("Montag", "Dienstag", "Mittwoch", "Donnerstag",
    "Freitag", "Samstag", "Sonntag");
foreach($stage as $tag) {
    echo $tag."<br />";
}
```

- ***foreach (\$array_ausdruck as \$schluessel => \$wert) Anweisung***

```
//jeder Spieler hat bestimmte Punktzahl
$spieler["Tomas"] = 30;
$spieler["Anna"] = 20;
$spieler["Sarah"] = 25;
foreach($spieler as $schluessel => $wert) {
    echo $schluessel." hat ".$wert." erreicht <br />";
}
```

Alternative Syntax

Für if-else und Schleifen mit while, for, foreach und switch gibt es eine alternative Syntax.

- anstelle der öffnenden geschweiften Klammer: ein Doppelpunkt
- anstelle der schließenden Klammer: endif, endwhile, endfor, endforeach oder endswitch.

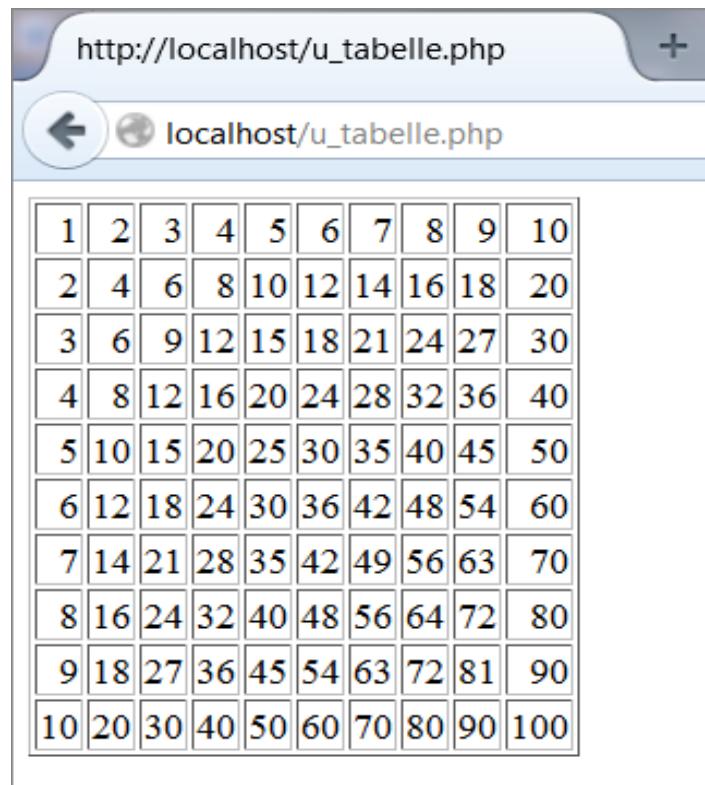
Beispiel:

```
$i = 5;  
if ($i > 4):  
    echo "$i ist größer als 4<br />\n ";  
endif;  
$j = 1;  
while ($j < 6):  
    echo "$j: hallo<br />\n";  
    $j++;  
endwhile;
```

Wenn HTML- und PHP-Code intensiv gemischt werden, dann ist ein weiter unten auftretendes <?php endwhile; ?> besser zuzuordnen als nur ein <?php } ?>.

Aufgabe 6.2:

Schreiben Sie ein Programm, in dem mithilfe zweier geschachtelter for-Schleifen das »kleine Einmaleins« in einer Tabelle ausgegeben wird. Die Ausgabe soll wie folgt aussehen.



The screenshot shows a web browser window with the URL `http://localhost/u_tabelle.php`. The page displays a 10x10 multiplication table where each cell contains the product of its row and column indices. The numbers are color-coded: the first row and column are in blue, while all other numbers are in orange. The table is as follows:

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

Aufgabe 6.3:

Erstellen Sie ein kleines Computerspiel. Zwei Spieler würfeln gegeneinander (Zufallszahlengenerator). Die Würfe jedes Spielers sollen addiert werden. Sobald einer der beiden Spieler oder beide Spieler nach einer Spielrunde den Wert 25 erreicht oder überschritten haben, ist das Spiel zu Ende.

1 Einleitung: Modulziele, Prüfungsform & Organisatorisches

2 Einführung in Skriptsprachen

3 Einführung in PHP

4 Datentypen

5 Operatoren

6 Kontrollstrukturen

7 Funktionen

8 Objektorientierung

9 Ein- und Ausgabe (Dateien)

10 Web-Formularanbindung

11 Datenbankanbindung

12 Typische Anwendungen

Funktionen

Funktionen fassen mehrere Anweisungen zusammen. Dadurch werden Quellcodes einfacher lesbar, und Fehler können an einem Ort behoben werden.

Die Syntax:

```
function funktionsname ([ parameter [, ...] ]) { Anweisungen }
```

- Funktionsname: Groß-/Kleinschreibung ist nicht relevant
- Parameterliste: Eine durch Kommas getrennte Liste von Ausdrücken
- Parameterliste ist optional

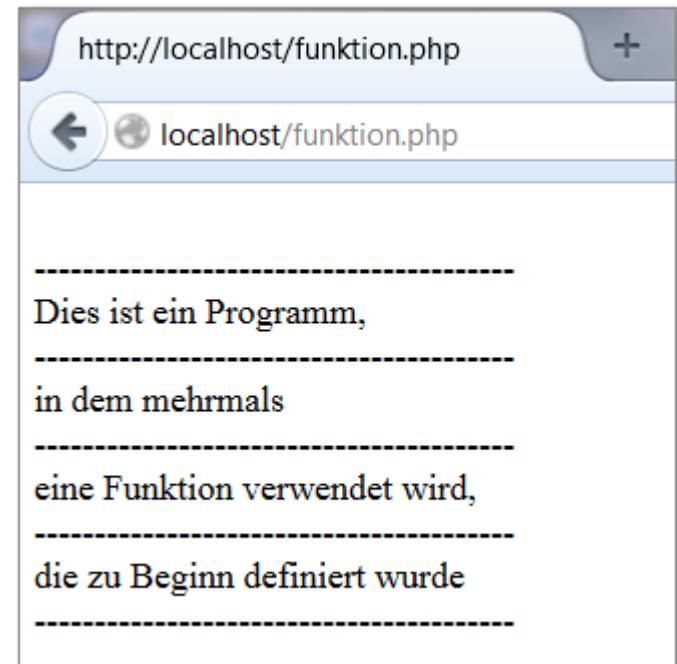
Manche Programmiersprachen (z.B. Pascal) unterscheiden zwischen Funktionen, die einen Wert zurückgeben, und Prozeduren, die keinen Wert zurückgeben. PHP macht hier, genau wie C und C++, keinen Unterschied.

Funktionstypen

- Benutzerdefinierte Funktionen
- Vordefinierte Funktionen (wie z.B. `print_r()`)

Benutzerdefinierte Funktion ohne Rückgabe

```
<html>
<head>
<?php
function trennstrich() {
    echo "<br />";
    for ($i=1; $i<=40; $i=$i+1)
        echo "-";
    echo "<br />";
}
?>
</head>
<body>
<?php
    trennstrich();
    echo "Dies ist ein Programm,";
    trennstrich();
    echo "in dem mehrmals";
    trennstrich();
    echo "eine Funktion verwendet wird,";
    trennstrich();
    echo "die zu Beginn definiert wurde";
    trennstrich();
?>
</body>
</html>
```



Rückgabewert einer Funktion

Eine Funktion kann auch einen Wert zurückgeben. Der zurückgelieferte Wert muss entweder in einer Variablen gespeichert oder direkt ausgegeben werden

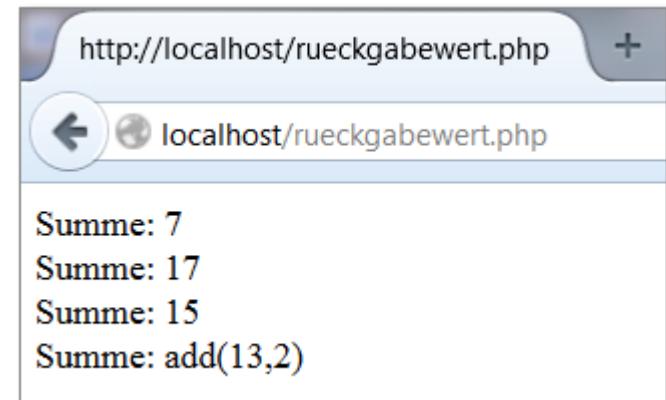
```
function add($z1, $z2) {
    $summe = $z1 + $z2;
    return $summe;
}
```

```
$c = add(3,4); // aufrufende Stelle
echo "Summe: $c <br />";
```

```
$x = 5;
$c= add($x,12); // aufrufende Stelle
echo "Summe: $c <br />";
```

```
// aufrufende Stelle innerhalb der Ausgabe
echo "Summe: " . add(13,2) . "<br />";
```

```
// Ausgabe in Zeichenkette: falsch!
echo "Summe: add(13,2)";
```



Hinweis

- über *return* lässt sich auch ein Array mit mehreren Werten zurückgeben.
- *return* beendet die Funktion, d.h. Anweisungen dahinter werden nicht ausgeführt. Man kann eine Funktion auch vorzeitig verlassen.

```
function copyright() {  
    return "Copyright 2014";  
    echo "Hallo, hallo, warum hört mich denn niemand";  
}
```

Aufgabe 7.1:

Erstellen Sie eine Funktion *vermerk()*, die einen Entwicklervermerk erzeugt. Die Funktion soll von verschiedenen Entwicklern genutzt werden können.

Vorname, Nachname und Abteilung werden als Parameter an die Funktion übergeben.

Jedes Mal, wenn die Funktion aufgerufen wird, erscheint eine Ausgabezeile mit diesen Informationen und der E-Mail-Adresse. Die E-Mail-Adresse setzt sich wie folgt zusammen: *vorname.nachname@abteilung.phpdevel.de*.

Testen Sie Ihre Funktion mit einem geeigneten Programm, in dem die Funktion mehrmals mit verschiedenen Informationen aufgerufen wird.

Eine mögliche Ausgabe sehen Sie im Folgenden:

```
Programmteil von Bodo Berg, Abteilung FE2  
E-Mail: Bodo.Berg@FE2.phpdevel.de  
  
Programmteil von Hans Heim, Abteilung SU3  
E-Mail: Hans.Heim@SU3.phpdevel.de
```

Parameterübergabe

PHP bietet hier zwei Möglichkeiten an:

- **Call-by-Value:** Übergabe der Parameter als Kopie.
 - Der Wert einer Variable wird in die Parametervariable kopiert
 - Innerhalb der Funktion wird auf der Kopie gearbeitet
 - Veränderung der Kopie hat keine Rückwirkung auf das Original.
- **Call-by-Reference:** Übergabe der Parameter als Referenz auf das Original.
 - Es wird ein Zeiger auf eine Variable übergeben (Parameter werden durch den &-Operator gekennzeichnet)
 - Innerhalb der Funktion wird auf der Variable außerhalb der Funktion gearbeitet
 - Eine Veränderung hat Rückwirkung auf das Original.

Parameterübergabe: Kopie und Referenz

Beispiel:

```

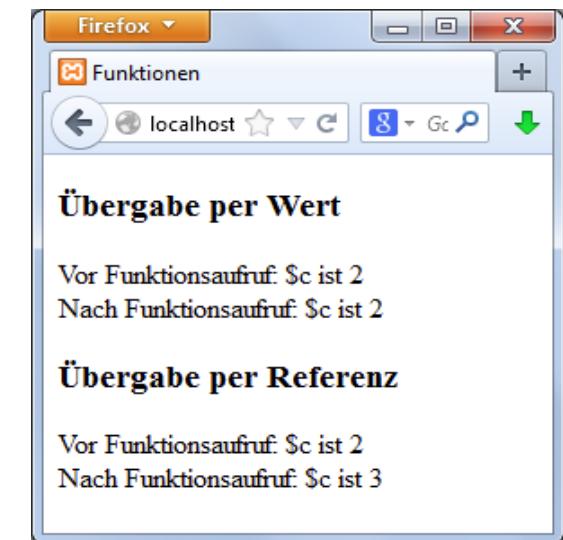
echo "<h3>Übergabe per Wert</h3>\n";
function veraendern($a) {
    $a++;
}

$c = 2;
echo "Vor Funktionsaufruf: \$c ist $c<br />\n";
veraendern($c);
echo "Nach Funktionsaufruf: \$c ist $c<br />\n";

echo "<h3>Übergabe per Referenz</h3>\n";
function veraendern2(&$a) {
    $a++;
}

echo "Vor Funktionsaufruf: \$c ist $c<br />\n";
veraendern2($c);
echo "Nach Funktionsaufruf: \$c ist $c<br />\n";

```



// Aufruf der Funktion per Referenz, z.B. veraendern(&\$c), veraltet

Defaultwerte für Parameter

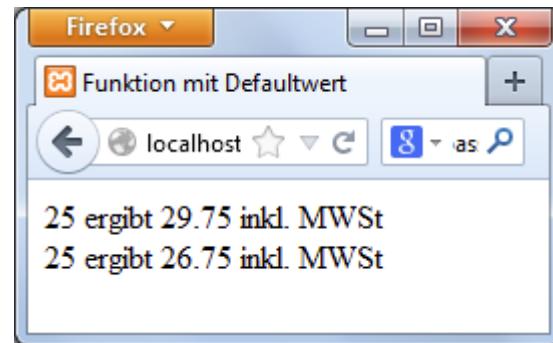
Defaultwerte für Parameter

Bei Funktionen kann man auch Defaultwerte für Parameter angeben, die genommen werden, wenn der entsprechende Parameter nicht angegeben ist.

```
function brutto ($netto, $mwstSatz = 19) {  
    return $netto * (100 + $mwstSatz) / 100;  
}
```

```
$betrag = 25;  
$bruttowert = brutto ($betrag);  
echo "$betrag ergibt $bruttowert inkl. MWSt.<br />\n";
```

```
$bruttowert2 = brutto ($betrag, 7);  
echo "$betrag ergibt $bruttowert2 inkl. MWSt.<br />\n";
```



Geltungsbereiche von Variablen

▪ Lokale Variablen

- werden innerhalb einer Funktion definiert und stehen nur dort zur Verfügung.
- Ein Parameter, der als Kopie an eine Funktion übergeben wird, ist dort lokal.

▪ Globale Variablen

- werden außerhalb einer Funktion definiert und stehen nur außerhalb derselben zur Verfügung.
- Falls eine globale Variable innerhalb einer Funktion benutzt werden soll, dann muss sie dort entweder mit dem Schlüsselwort *global* bekannt gemacht oder als Parameter übergeben werden.

▪ Superglobale Variablen

- sind PHP-Systemvariablen, die innerhalb & außerhalb von Funktionen zur Verfügung stehen, wie z.B. das assoziative Feld `$_POST`, das die Namen und Werte von Formularfeldern zur Verfügung stellt.

Hinweis: Die Variablen der Funktionen sollten immer „so lokal wie möglich“ sein:

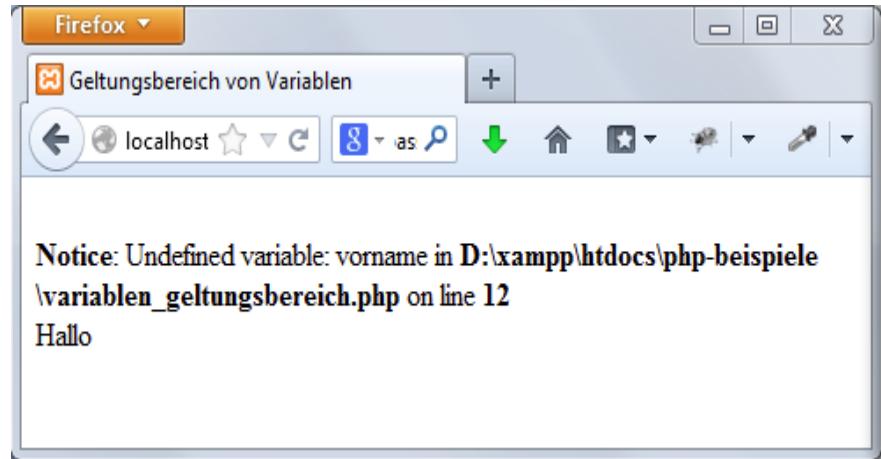
- Verbesserung der Modularisierung des Programms und
- Erleichterung der Wiederverwendbarkeit der Funktionen für andere Programme.

Geltungsbereiche von Variablen

Beispiele

```
$vorname = "Lilli";  
function gruss() {  
    echo "Hallo $vorname<br />\n";  
}  
gruss();
```

```
$vorname = "Lilli";  
function gruss() {  
    global $vorname;  
    echo "Hallo $vorname<br />\n";  
}  
gruss();
```



Zugriff per \$GLOBALS

- Ein vordefiniertes **assoziatives Array**, das als Elemente alle im globalen Geltungsbereich existierenden Variablen enthält.
- Der Schlüssel ist jeweils der Name der Variablen, und der Wert ist der Wert der Variablen,
- d.h., über `$GLOBALS["vorname"]` kann man innerhalb der Funktion auf den Inhalt der außerhalb der Funktion stehenden Variable `$vorname` zugreifen.

```
$vorname = "Lilli";  
function gruss () {  
    echo "Hallo {$GLOBALS['vorname']} <br />\n";  
}  
gruss();
```

Aufgabe 7.2:

Schreiben Sie ein PHP-Programm mit einer Funktion *rechne()*.

Dieser Funktion werden zwei Zahlen übergeben. Sie soll zwei Ergebnisse über die Parameterliste zurückliefern: zum einen die Summe der beiden übergebenen Zahlen und zum anderen das Produkt der beiden übergebenen Zahlen.

Alle beteiligten Zahlen sollen im Hauptteil des Programms, also außerhalb der Funktion, ausgegeben werden.

Verwenden Sie zur Übergabe die zweite Methode (Call-by-Reference). Nach einem Funktionsaufruf mit den Parametern 5 und 7 und der anschließenden Ausgabe erscheint eine Ausgabe wie im Folgenden.

Die Summe von 7 und 5 ist 12

Das Produkt von 7 und 5 ist 35

Rekursive Funktionen

```
<?php
    function recursion($a)
    {
        if ($a < 20)  {
            echo "$a\n";
            recursion($a + 1);
        }
    }

?>
```

Hinweis: Rekursive Funktions-/Methodenaufrufe können zu einem Stacküberlauf und damit zum Programmabbruch führen. Besonders unbegrenzte Rekursion wird als Programmierfehler erachtet.

Anonyme Funktionen

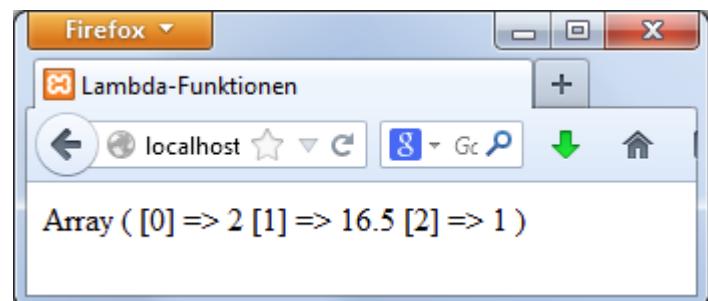
Anonyme Funktionen

Funktionen ohne Namen (auch **Lambda-Funktionen** genannt). Diese Funktionen kann man z.B. einer Variablen zuweisen und dann aufrufen:

```
$lambda = function($a) { return $a / 2; }; // Strichpunkt wichtig,  
$ergebnis = $lambda(5); // ansonsten Fehlermeldung.
```

- ermöglicht, Funktionen als Parameter zu übergeben, um bestimmte Verarbeitungsschritte durchzuführen, wie z.B. bei der Funktion `array_map()`
 - Als ersten Parameter erwartet `array_map()` eine Funktion, die bestimmt, was mit den einzelnen Elementen des Arrays geschehen soll; als zweiten Parameter das Array.

```
$lambda = function($a) { return $a / 2; };  
$zahlen = array (4, 33, 2);  
  
$ergebnis = array_map($lambda, $zahlen);  
print_r($ergebnis);
```



Variable Parameteranzahl

Variable Parameteranzahl

Die Anzahl der Parameter bei einem Funktionsaufruf muss nicht unbedingt genau der Anzahl der Parameter entsprechen, die bei der Definition der Funktion vorgegeben wurden.

Die folgenden Funktionen ermöglichen die Nutzung variabler Parameteranzahl.

func_num_args(): liefert die Anzahl der übergebenen Parameter.

func_get_arg(): liefert einen bestimmten Parameter aus der Parameterliste.

func_get_args() : (mit einem s am Ende) liefert ein numerisch indiziertes Feld mit allen übergebenen Parametern.

Variable Parameteranzahl

Beispiel: func_num_args() und func_get_arg():

```
<html>
<body>
<?php
function addiere() {
    $anz = func_num_args();
    echo "<p>Anzahl der Werte: $anz<br />";
    echo "Werte: ";
    $sum = 0;
    for($i=0; $i<$anz; $i++) {
        echo func_get_arg($i) . " ";
        $sum = $sum + func_get_arg($i);
    }
    echo "<br />Summe der Werte: $sum</p>";
}
addiere(2,3,6);
addiere(13,26);
addiere(65,-3,88,31,12.5,7);
?>
</body>
</html>
```

Anzahl der Werte: 3

Werte: 2 3 6

Summe der Werte: 11

Anzahl der Werte: 2

Werte: 13 26

Summe der Werte: 39

Anzahl der Werte: 6

Werte: 65 -3 88 31 12.5 7

Summe der Werte: 200.5

Parameter entpacken

Seit PHP 5.6 kann man Parameter einer Funktion auch »verpackt« übergeben.
Sie werden dann von der Funktion »entpackt«.

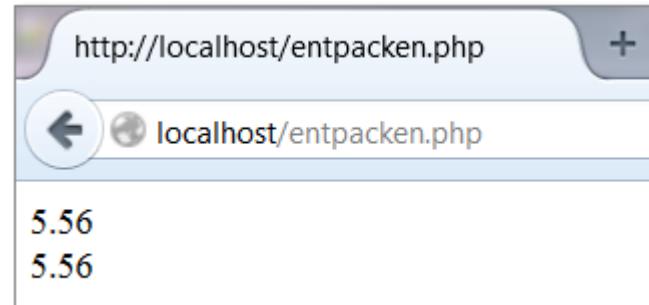
```
<html>
<body>
<?php

function mittelwert($a, $b, $c, $d, $e) {
    return ($a + $b + $c + $d + $e) / 5;
}

echo mittelwert(3.2, 14.5, 5.7, 4.2, 0.2) . "<br />";

$feld = array(5.7, 4.2, 0.2);
echo mittelwert(3.2, 14.5, ...$feld);

?>
</body>
</html>
```



Generatoren

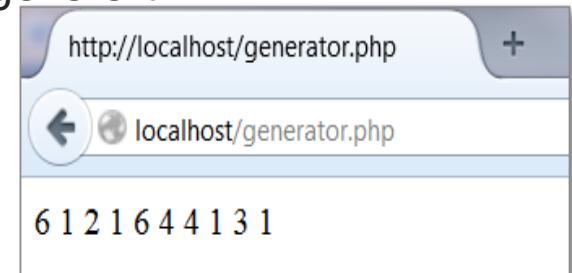
Ein besonderer Typ von Funktionen, liefern eine Reihe von Werten

- die Werte werden mithilfe des Schlüsselworts **yield** geliefert.

```
function wuerfelwertGenerator() {  
    for($i=1; $i<=10; $i++)  
        yield rand(1, 6);  
}
```

- Zu jedem Zeitpunkt steht nur ein Wert im Arbeitsspeicher (Optimierung der Nutzung des Arbeitsspeichers).
- Wird die Generatorfunktion innerhalb einer foreach-Schleife aufgerufen, endet sie nicht nach einem Durchlauf, sondern unterbricht nur. Auf diese Weise werden alle generierten Werte nacheinander geliefert.

```
foreach (wuerfelwertGenerator() as $wert)  
    echo "$wert ";
```



Include und require

Benutzerdefinierte Funktionen, die von mehreren Programmen genutzt werden sollen, können in externe Dateien ausgelagert werden.

- Einbindung mithilfe von *include* bzw. *require*
- Beide haben die gleiche Wirkung – mit einer Ausnahme. Falls eine einzubindende Datei nicht gefunden wird:
 - *require* beendet das Programm mit einem Fehler;
 - *include* gibt lediglich eine Warnung aus, und das Programm läuft weiter.
- Es wird empfohlen, einer solchen Datei die Endung *.inc.php* zu geben, um die Datei erkennbar zu machen

Funktionen im PHP-Manual

Firefox ▾

php PHP: date - Manual +

www.php.net/manual/de/function.date.php

php Downloads Documentation Get Involved Help Search

PHP-Handbuch / Funktionsreferenz / Datums- und zeitrelevante Erweiterungen / Datum/Uhrzeit / Datum/Uhrzeit Funktionen

Datum/Uhrzeit Funktionen

- [checkdate](#)
- [date_add](#)
- [date_create_from_format](#)
- [date_create_immutable_from_format](#)
- [date_create_immutable](#)
- [date_create](#)
- [date_date_set](#)
- [date_default_timezone_get](#)
- [date_default_timezone_set](#)
- [date_diff](#)
- [date_format](#)
- [date_get_last_errors](#)
- [date_interval_create_from_date_string](#)
- [date_interval_format](#)
- [date_isodate_set](#)
- [date_modify](#)
- [date_offset_get](#)
- [date_parse_from_format](#)
- [date_parse](#)
- [date_sub](#)

date

(PHP 4, PHP 5)

Beschreibung

```
string date ( string $format [, int $timestamp = time() ] )
```

Gibt einen formatierten String anhand eines vorzugebenden Musters zurück. Dabei wird entweder der angegebene **Timestamp** oder die gegenwärtige lokale Zeit berücksichtigt, wenn kein Timestamp angegeben wird. Mit anderen Worten ausgedrückt: der Parameter **Timestamp** ist optional und falls dieser nicht angegeben wird, wird der Wert der Funktion [time\(\)](#) angenommen.

Parameter-Liste

format

Das Muster des auszugebenden Datums-/Zeit-String. Unten sind die verfügbaren Formatierungsoptionen angegeben. Es gibt auch verschiedene [vordefinierte Konstanten](#), die verwendet werden können, z.B. enthält DATE_RSS das Formatierungsmuster 'D, d M Y H:i:s'.

*Die folgenden Zeichen werden im Parameter **Format** erkannt*

Format-Zeichen	Beschreibung	Beispiel für Rückgabewerte
<i>Tag</i>	---	---
<i>d</i>	Tag des Monats, 2-stellig mit führender Null	01 bis 31
<i>D</i>	Wochentag, gekürzt auf drei Buchstaben	Mon bis Sun

[Feedback & Ideas](#)

Vordefinierte Funktionen

- Funktionen für Variablen
- Funktionen für Strings (Zeichenketten)
- Funktionen für Arrays (Felder)
- Funktionen für Datum und Zeit

Funktionen für Variablen

- ### Funktionen für Variablen
- *print_r()* und *var_dump()* : um Informationen über Variablen herauszufinden.
 - *is_string()*: prüft ob etwas ein String ist
 - *isset ()*: prüft ob eine Variable gesetzt ist (z.B. `$z = 0;`)
 - *empty()*: prüft, ob ein String leer ist, liefert aber auch **wahr**,
 1. wenn eine nicht gesetzte Variable überprüft wird oder
 2. wenn eine Variable den Wert 0 hat.
 3. Allgemeinen gibt *empty()* wahr zurück, wenn der Wert false ergibt.

```
$z = 0;  
$falsch = false;  
if (empty($gibtsnicht)) { //wahr nach 1  
    echo " das ist zu wenig - ";  
}  
if (empty($z)) { //wahr nach 2  
    echo " von nichts kommt nichts - ";  
}  
if (empty($falsch)) { //wahr nach 3  
    echo " wieder nix ";  
}
```

Aufgabe 7.3:

Arrays lassen sich gut mit `print_r()` ausgeben. Die hilfreichen Einrückungen sieht man allerdings nur im HTML-Quellcode der Seite oder wenn man um den Aufruf von `print_r()` `<pre>` und `</pre>` ergänzt:

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

Schreiben Sie eine Funktion, die ein Array entgegennimmt und dieses Array mit `print_r()` und `<pre>` behandelt ausgibt.

Testen Sie dann Ihre Funktion mit einem Array.

Funktionen für Strings

Funktionen für Strings

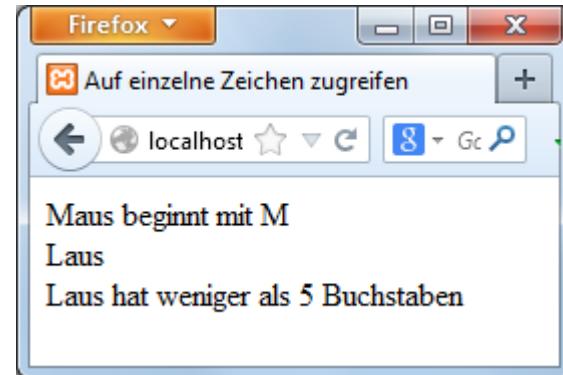
Zur Bearbeitung von Strings gibt es viele nützliche vordefinierte Funktionen.

Einzelne Zeichen eines Strings ausgeben und ändern

- Um einzelne Zeichen aus einem String zu ermitteln oder auch zu ersetzen, kann man den String so behandeln, als wäre er ein Array.

```
$tier = "Maus";
$erster = $tier[0];
echo "$tier beginnt mit $erster<br />\n";
$tier[0] = "L";
echo $tier;

if (!isset($tier[4])) {
    echo "<br />\n $tier hat weniger als 5 Buchstaben";
}
```



Funktionen für Strings

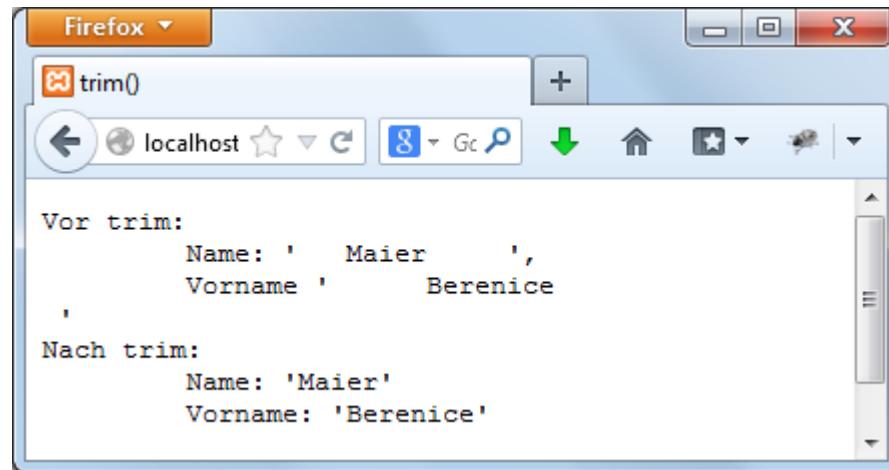
Die Länge eines Strings ermitteln

```
$name = " Hans-Heinerich ";           // 16 Zeichen
$laenge = strlen($name);
echo "$name ist $laenge Zeichen lang (inkl. Leerzeichen)<br />\n";
```

Leerzeichen entfernen

- trim() entfernt Leerzeichen am Anfang und Ende eines Strings (z.B. einer Eingabe)

```
01 $nn = " Maier ";
02 $vn = "\tBerenice\n ";
03 echo "<pre>";
04 echo "Vor trim:
05         Name: $nn,
06         Vorname $vn.";
07 $nn = trim($nn);
08 $vn = trim($vn);
09 echo "Nach trim:
10         Name: $nn.
11         Vorname: $vn.";
12 echo "</pre>";
```



Formatierte Ausgabe über printf()

- Erwartet als ersten Parameter einen Formatierungsstring (durch % gekennzeichnet), darauf können mehrere Parameter verschiedenen Typs folgen.
 - Z.B: %s für einen String und %d für einen Integer.

```
printf("%s ist %d Jahre alt", "Ben", 4); // Ben ist 4 Jahre alt.
```

```
printf("%d", 2.567); // 2 → konvertiert, da anderer Typ angegeben
```

```
printf("%.2f", 34.567); // 34.57 -> auf 2 Dezimalstellen beschränkt
```

- Ausgabe mit Bedingungen

```
$genau = true;  
$ergebnis = 9.123456;  
if ($genau) {  
    $format = "%.4f";  
} else {  
    $format = "%.2f";  
}  
printf($format, $ergebnis);
```

Funktionen für Strings

Zeichen suchen, finden und ersetzen

Es gibt viele Funktionen

- `strpos()` ermittelt das erste Vorkommen eines Zeichens in einem String.

```
$satz = "Der Hund hat einen Knochen";
$suche = "noch";
$pos = strpos($satz, $suche); // Parameter: wo wird was gesucht,
if ($pos === false) {           // false, wenn nichts gefunden
    echo "$suche ist nicht in $satz";
} else {
    echo "$suche befindet sich an Position $pos in '$satz'";
}
```

Warum wurde das dreifache `=` verwendet?

Beide müssen vom denselben Typ sein.

Der Grund: `$pos` würde 0 zurückliefern,
wenn der String an der 0. Position wäre.

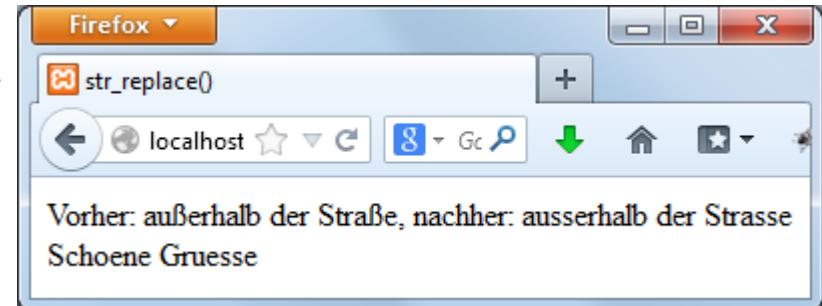
Dann würde `$pos == false` ebenfalls true zurückliefern, da 0 – in einen booleschen Wert
konvertiert – false ergibt. Durch das `==` erhält man auch in diesem Fall das richtige
Ergebnis (Testen mit „Der“ im Beispiel)



Funktionen für Strings

- str_replace() ersetzt im angegebenen String bestimmte Zeichenfolgen durch andere.

```
$satz = "außerhalb der Straße";
$erg = str_replace("ß", "ss", $satz);
echo "Vorher: $satz, nachher: $erg";
echo "<br />\n";
```

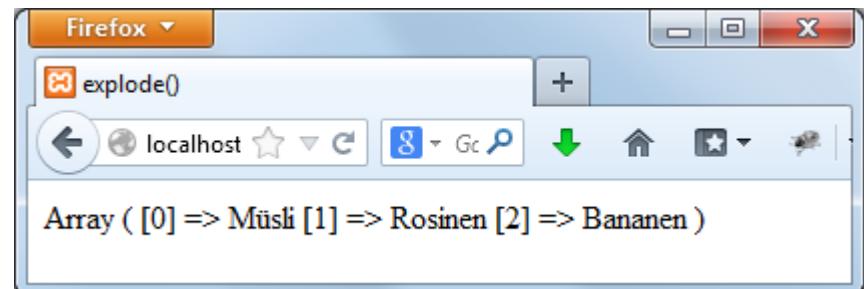


```
$text = "Schöne Grüße";
$sonder = array("ö", "ä", "ü", "ß", "Ö", "Ü", "Ä");
$ohne = array("oe", "ae", "ue", "ss", "Oe", "Ue", "Ae");
echo str_replace($sonder, $ohne, $text);
```

- String in Bestandteile zerlegen über explode()

- Parameter: (1) das Trennzeichen, an dem die Aufteilung stattfinden soll, (2) String, der aufgeteilt werden soll. Zurückgeliefert wird ein Array:

```
$str = "Müsli Rosinen Bananen";
$liste = explode(" ", $str);
print_r($liste);
```



Volle Freiheit mit regulären Ausdrücken

Manchmal wollen wir nach einem bestimmten Schema (Muster) suchen, um diesen herauszufiltern, zu überprüfen oder zu ersetzen.

- z.B. Telefonnummern, die einem bestimmten Schema – Zahlen, Klammern, Bindestriche und Schrägstriche.

Ein regulärer Ausdruck

- ist eine Zeichenkette auf Basis von syntaktischen Regeln, mit denen verschiedenste Arten von Textmustern beschreibbar sind
- dient zur Mustererkennung in Texten.
- wird üblicherweise mit Begrenzungszeichen / eingefasst.

```
$muster = "/und/";      // ein Muster für die Zeichenfolge "und"
```

- **preg_match()** prüft, ob ein Muster in einem String vorhanden ist

```
$muster = "/und/";  
$in = "Der Hund hat einen Knochen";  
if(preg_match($muster, $in)) {  
    echo "$muster passt auf '$in'";  
}
```

Mächtig und interessant werden die regulären Ausdrücke durch verschiedene Sonderzeichen.

- . (der Punkt) steht für ein beliebiges Zeichen:
/.und/ passt auf **Hund**, aber auch auf **Mund** oder **Kund**.
- *einen Punkt wörtlich verwenden*: maskieren (\.)

Muster »verankern«.

- ^: Treffer am Anfang des String
 - /^.und/ passt nicht auf **Der Hund**, aber auf **Hund**
 - /^.er/ passt auf **Der Hund**
- \$: Treffer am Ende des Strings
 - /.chen\$/ passt auf **Knochen**
 - /^hat\$/ passt auf **hat**, aber nicht auf **hatte** oder auf **anhat**
- ?: Vorheriges Zeichen ist optional
 - /iPhone[1-8]?>/ -Passt zu **iPhone**, **iPhone2** usw. bis **iPhone8**

Funktionen für Strings

- Zeichenklassen (in eckigen Klammern): an einer Stelle kann eines von mehreren Zeichen stehen
 - /[HM]und/ passt auf **Hund** oder **Mund**, nicht aber auf **Fund**
- ^-Zeichen: keines der in eckigen Klammern angegebenen Zeichen darf vorkommen.
 - /[^HM]und/ passt auf **Fund**, **Kund**, aber nicht auf **Hund** oder **Mund**
- Einen Zeichenbereich kann man über einen Bindestrich festlegen:
 - /[A-Z]und/ passt auf einen beliebigen Großbuchstaben, gefolgt von **und**.
 - /[0-9]-mal/ passt auf **1-mal**, **2-mal**, aber nicht auf **dreimal**
- Über Quantifizierer spezifiziert man, wie oft ein bestimmtes Zeichen vorkommen darf, z.B: + (mindestens einmal, aber beliebig häufig)
 - /^[A-Za-z]+und\$/ passt auf **Stund**, **Hund** und auch auf **Basketballbund**

Aufgabe 7.4:

Mit welchem regulären Ausdruck könnte man überprüfen, dass ein Dateiname ein JPEG-Bild ist? (Die übliche Endung für JPEG-Bilder kann sowohl *jpeg* als auch *jpg* sein!)

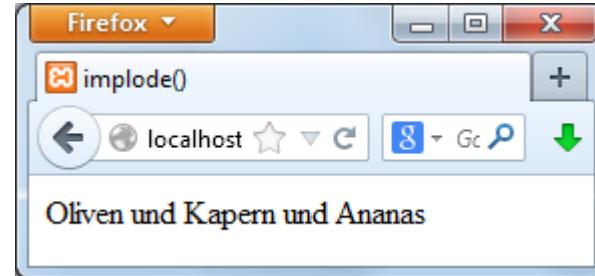
Funktionen für Arrays

Funktionen für Arrays

Arrays und String

- *implode()*: verbindet mehrere Arrayelemente zu einem String
 - nützlich, z.B. um ein Array in Cookies oder Datenbank zu schreiben

```
$liste = array ("Oliven", "Kapern", "Ananas");  
$str = implode(" und ", $liste);  
echo $str;
```



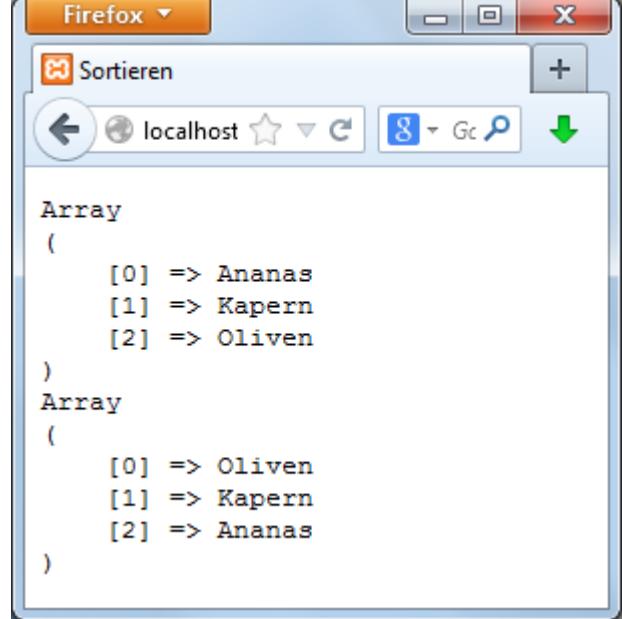
Arrays sortieren

Man kann bei Arrays sowohl Schlüssel als auch Werte sortieren, und das noch auf- oder absteigend etc.

Sortieren von indizierten Arrays:

- `sort()` und `rsort()`: beide erwarten als Parameter ein Array, das per Referenz übergeben wird (d.h. die Funktionen ändern das Array selbst).
- Funktionsbeschreibung im Manual: *bool sort (array &\$array)*
- `sort()` sortiert aufsteigend, `rsort()` absteigend.

```
echo "<pre>";  
$liste = array ("Kapern", "Oliven", "Ananas");  
  
sort($liste);  
print_r($liste);  
  
rsort($liste);  
print_r($liste);  
  
echo "</pre>";
```



```
Array  
(  
    [0] => Ananas  
    [1] => Kapern  
    [2] => Oliven  
)  
Array  
(  
    [0] => Oliven  
    [1] => Kapern  
    [2] => Ananas  
)
```

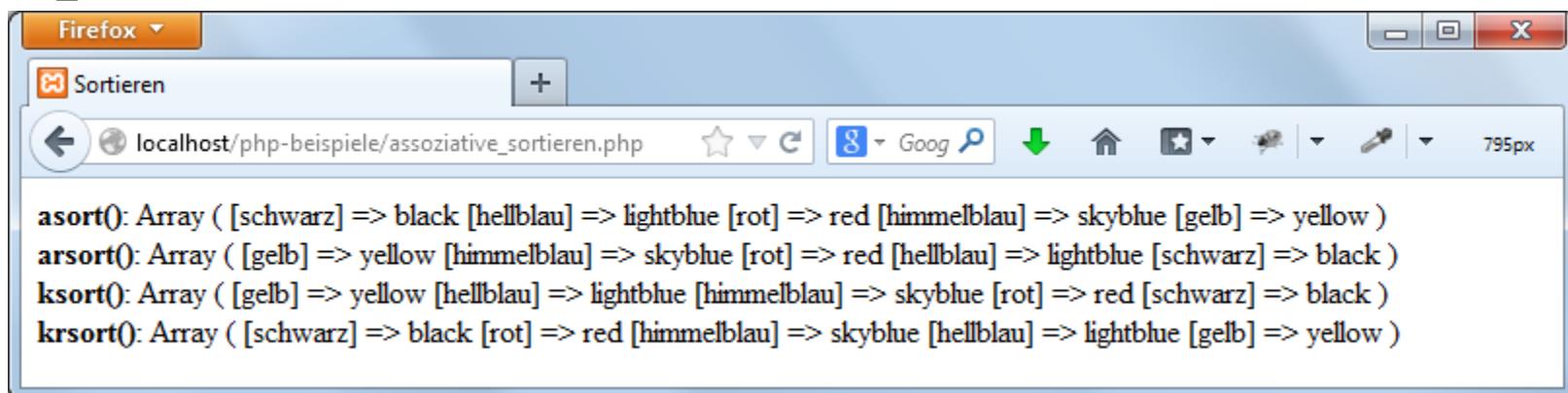
Sortieren von assoziativen Arrays

- `asort()`: sortiert ein Array nach Werten aufsteigend.
- `ksort()`: sortiert nach Schlüsseln aufsteigend.
- `arsort()`: sortiert nach Werten absteigend
- `krsort()`: sortiert nach Schlüsseln absteigend.

- **Beispiel:** s. nächste Folie
 - Zuerst wird vorwärts und rückwärts nach Werten sortiert, dann nach Schlüssel:
 - Die Zuordnung von Wert zu Schlüssel bleibt dabei erhalten.

Funktionen für Arrays

```
01 $farben = array ("hellblau" => "lightblue",
02                     "schwarz" => "black",
03                     "gelb" => "yellow",
04                     "himmelblau" => "skyblue",
05                     "rot" => "red");
06 echo "<strong>asort()</strong>: ";
07 asort($farben);
08 print_r($farben);
09 echo "<br /><strong>arsort()</strong>: ";
10 arsort($farben);
11 print_r($farben);
12 echo "<br /><strong>ksort()</strong>: ";
13 ksort($farben);
14 print_r($farben);
15 echo "<br /><strong>krsort()</strong>: ";
16 krsort($farben);
17 print_r($farben);
```



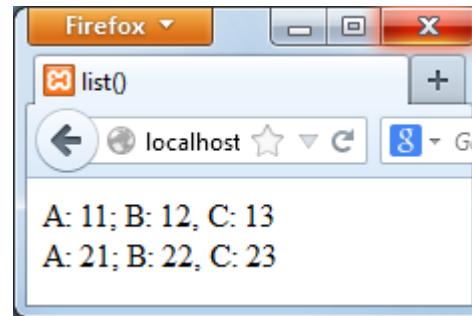
Weitere Arrayfunktionen

- `list()`: um die Elemente von Arrays einzelnen Variablen zuzuweisen

```
$liste = array ("Kapern", "Oliven", "Ananas");  
list($a, $b, $c) = $liste;  
echo "mit $a, $b und $c ...";
```

- `list()` kann auch in einer `foreach`-Schleife verwendet werden.

```
$array = [  
    [11, 12, 13],  
    [21, 22, 23]  
];  
foreach ($array as list($a, $b, $c)) {  
    echo "A: $a; B: $b, C: $c<br />\n";  
}
```



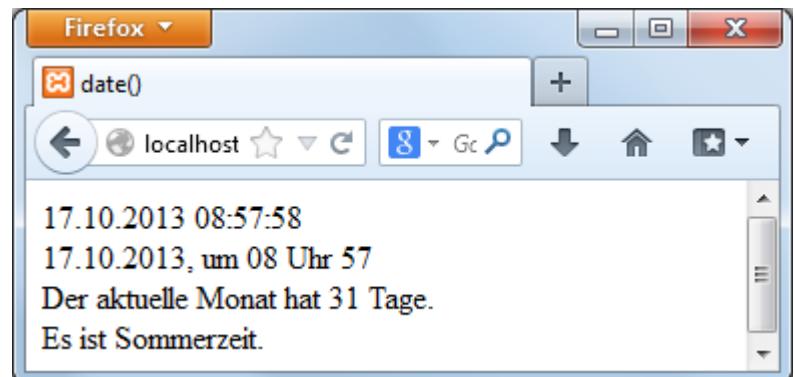
Funktionen für Datum und Zeit

Funktionen für Datum und Zeit

Es gibt mehrere Funktionen, um ein aktuelles Datum formatiert auszugeben.

- *date()*: das folgende Skript zeigt den Einsatz von *date()*

```
date_default_timezone_set("Europe/Berlin");
echo date("d.m.Y H:i:s");
echo "<br />\n";
echo date("d.m.Y, \u00m H \u00h\\r i"); // \\r, da \r = Wagenrücklauf
echo "<br />\nDer aktuelle Monat hat ";
echo date("t") . " Tage.<br />\n";
if (date("I") == 1) {
    echo "Es ist Sommerzeit.";
} else {
    echo "Es ist keine Sommerzeit.";
}
```



Mögliche Angaben bei Date()

Formatierungsstring	Erklärung	Beispiel
Tag		
d	Zweistelliger Tag des Monats	01 bis 31
D	Abgekürzter Wochentag	Mon bis Sun
j	Tag des Monats ohne führende 0	1 bis 31
I	Wochentag ausgeschrieben	Sunday bis Saturday
N	Eine der ISO-8601-Norm entsprechende Darstellung des Wochentags (erst seit PHP 5.1.0)	1 (Montag) bis 7 (Sonntag)
w	Numerische Darstellung des Wochentags	0 (Sonntag) bis 6 (Samstag)
z	Der wievielte Tag des Jahres ist es? Die Zählung beginnt bei 0.	0 bis 365
Woche		
W	Gibt an, welche Kalenderwoche im Jahr es ist. Begonnen wird mit Montag. Ist konform zur ISO-8601-Norm.	10
Monat		
F	Ausgeschriebener Monatsname (englisch)	January bis December
m	Monat zwischen 01 und 12	01 bis 12
M	Abgekürzter Monatsname	Jan bis Dec
n	Monat zwischen 1 und 12 (ohne führende 0)	1 bis 12
t	Anzahl der Tage in einem Monat	28 bis 31

Formatierungsstring	Erklärung	Beispiel
Jahr		
L	Schaltjahr	1, wenn Schaltjahr, sonst 0
Y	Vierstellige Jahreszahl	2009
y	Zweistellige Jahreszahl	99 oder 08
Zeit		
a	Kleingeschrieben am oder pm	am oder pm
A	AM oder PM großgeschrieben	AM oder PM
g	Stunde von 1 bis 12 ohne führende 0	1 bis 12
G	Stunde von 0 bis 23 ohne führende 0	0 bis 23
h	Stunde von 1 bis 12 mit führender 0	01 bis 12
H	Stunde von 0 bis 23 mit führender 0	00 bis 23
i	Minuten mit führender 0	00 bis 59
s	Sekunden mit führender 0	00 bis 59
u	Millisekunden (seit PHP 5.2.2)	54321
Zeitzone		
e	Identifiziert die Zeitzone (seit PHP 5.1.0)	UTC, GMT, Atlantic/Azores
I (großes i)	Sommerzeit oder nicht	Bei Sommerzeit 1, sonst 0
O	Unterschied zur mittleren Greenwich-Zeit (GMT) in Stunden	+0200
P	Unterschied zur GMT in Stunden mit Doppelpunkt zwischen Stunden und Minuten (seit PHP 5.1.3)	+02:00
T	Abkürzung der Zeitzone	EST, MDT ...
Z	Zeitzonenoffset relativ zur UTC	-43200 bis 50400
Vollständige Datum/Zeitangabe		
c	ISO-8601-Datum (seit PHP 5)	2009-04-27T08:30:21+02:00
r	Gemäß RFC 2822 formatiertes Datum, praktisch etwa für E-Mail- oder HTTP-Header	Thu, 21 Dec 2000 16:01:07 +0200
U	Seit der Unix-Epoche verstrichene Sekunden (1.1.1970 00:00:00 GMT)	

Aufgabe 7.5:

Verwenden Sie `date()`, um die Meldung »Schönes Wochenende« am Samstag und Sonntag ausgeben zu lassen. An anderen Wochentagen soll hingegen »Gute Woche« erscheinen.

1 Einleitung: Modulziele, Prüfungsform & Organisatorisches

2 Einführung in Skriptsprachen

3 Einführung in PHP

4 Datentypen

5 Operatoren

6 Kontrollstrukturen

7 Funktionen

8 Objektorientierung

9 Ein- und Ausgabe (Dateien)

10 Web-Formularanbindung

11 Datenbankanbindung

12 Typische Anwendungen

Objektorientierung

- einige grundlegende Funktionen von PHP sind objektorientiert konstruiert
- Größere PHP-Anwendungen wie Content-Management-Systeme sind häufig objektorientiert programmiert.

Objekte und Klassen

- In der objektorientierten Herangehensweise ist erst einmal alles ein Objekt
- Diese Objekte haben dann verschiedene *Eigenschaften (Datenfelder) und Methoden* (Funktionen).
 - Das Objekt Warenkorb hat z.B. als Eigenschaften die jeweilige Nummer des Kunden und eine Liste von Artikeln.
 - Außerdem hat es Methoden, um den Inhalt des Warenkorbs anzeigen zu lassen, einen Artikel zum Warenkorb hinzuzufügen oder zu löschen.
- Die Baupläne der Objekte sind die Klassen. Sie definieren, welche Eigenschaften und Methoden vorgesehen sind.

- Eine Klasse mit dem Schlüsselwort *class* erstellen

```
class Kunde {           // Üblich: Name beginnt mit Großbuchstaben
    public $name;        // Eigenschaft / Datenfeld
    public function halloSagen() { // Methode: function
        echo "Hallo";
    }
}
```

- Ein Objekt mit dem Schlüsselwort *new* erstellen

```
$neuerKunde = new Kunde();           // Rückgabe: eine Referenz
```

- Die einzelnen Eigenschaften setzen

```
$neuerKunde->name = "Anja";       // name ohne Dollarzeichen
```

- Auf die Methoden der Klasse zugreifen

```
$neuerKunde->halloSagen();
```

- Die Sichtbarkeit von Eigenschaften und Methoden lässt sich über die Schlüsselwörter *public*, *protected* und *private* festlegen.

Beispiel: Ein Objekt der Klasse Kunde

Das ganze Skript im Gesamtzusammenhang:

```
class Kunde
{
    public $name;
    public function halloSagen()
    {
        echo "Hallo";
    }
}
$neuerKunde = new Kunde(); // auch erlaubt: new Kunde
$neuerKunde->name = "Anja";
$neuerKunde->halloSagen();
echo " ";
echo $neuerKunde->name; // Ausgabe: Hallo Anja
```

- Es ist auch möglich, eine Methode oder Eigenschaft der Klasse direkt bei der Instanziierung aufzurufen

```
(new Kunde)->halloSagen();
```

Konstanten definieren

```
class Klasse {  
    const KONST_WERT = 42;  
    public function ausgabe() {  
        echo self::KONST_WERT;           // self: innerhalb der Funktion  
    }                                     // der Klasse  
}  
  
echo Klasse::KONST_WERT;                // Klassename außerhalb der Klasse  
echo "<br />\n";  
$obj = new Klasse();  
$obj->ausgabe();
```

Konstruktor

- eine Methode mit dem vorgegebenen Namen `__construct()`. // zwei Unterstriche
- wird automatisch aufgerufen, wenn ein neues Objekt erstellt wird.

```
class Kunde {  
    public $name;  
    public function __construct($name) {  
        $this->name = $name;    // Zugriff auf Eigenschaften  
    }                      // $this: Referenz auf das aktuelle Objekt  
  
    public function halloSagen() {  
        echo "Hallo $this->name";  
    }  
}  
  
$neuerKunde = new Kunde("Anja");  
$neuerKunde->halloSagen();           // Ausgabe: Hallo Anja  
echo $neuerKunde->name;              // Ausgabe: Anja
```

Aufgabe 8.1:

Definieren Sie eine Klasse mit dem Namen Fahrzeug, die die folgenden Eigenschaften hat:

- Farbe
- Hersteller

Und außerdem hat sie zwei Methoden, die jeweils eine Meldung ausgeben:

- Starten
- Stoppen

Erstellen Sie dann ein Objekt zu der Klasse!

Optionale Parameter

Sowohl Funktionen als auch Methoden können *optionale Parameter beinhalten*.

- erweitern die Möglichkeiten von Methoden und bieten auch alternative Aufrufmöglichkeiten für Methoden
 - Z.B. ihre Anwendung bei der Konstruktormethode ermöglicht, Objekte der Klasse auf verschiedene Art und Weise zu erzeugen.
- **Hinweis:** Die Möglichkeit des Überladens von Methoden, wie es aus anderen objektorientierten Sprachen bekannt ist, existiert in PHP nicht.
- Optionale Parameter sollten immer am Ende der Parameterreihe stehen.
- Falls bei einem Methodenaufruf optionale Parameter weggelassen werden, kann dies nur von rechts nach links innerhalb der Parameterreihe geschehen.

Beispiel: Optionale Parameter

```

class Fahrzeug {
    private $geschwindigkeit;
    private $bezeichnung;
    function __construct($bez = "xxx", $ge = 0) { // Konstruktor mit Opt. Par.
        $this->bezeichnung = $bez;
        $this->geschwindigkeit = $ge;
    }
    function beschleunigen($wert) {
        $this->geschwindigkeit += $wert;
    }
    function ausgabe() {
        echo "Name: $this->bezeichnung, Geschwindigkeit:
        $this->geschwindigkeit km/h<br />";
    }
}

$vespa = new Fahrzeug("Vespa Piaggio");
$scania = new Fahrzeug("", 62);
$jeep = new Fahrzeug("Jeep Cherokee", 45);
$hyundai = new Fahrzeug();

$vespa->ausgabe();
$scania->ausgabe();
$jeep->ausgabe();
$hyundai->ausgabe();

```

Name: Vespa Piaggio, Geschwindigkeit: 0 km/h
 Name: , Geschwindigkeit: 62 km/h
 Name: Jeep Cherokee, Geschwindigkeit: 45 km/h
 Name: xxx, Geschwindigkeit: 0 km/h

Statische Eigenschaften und Methoden

Statische Eigenschaften (Klassenvariablen) und statische Methoden (Klassenmethoden) können Sie aufrufen, ohne ein Objekt erstellt zu haben.

```
class Beispiel {  
    public static $zahl = 42;  
    public static function ausgeben() {  
        echo „Hallo“;  
    }  
}
```

```
Beispiel::ausgeben();      // Direkt aufrufen  
echo Beispiel::$zahl;
```

Aufgabe 8.2:

Definieren Sie eine Klasse *Math* mit mindestens zwei statischen Methoden:

- Die Methode *wurzel()* soll die Quadratwurzel liefern. Hierfür können Sie *sqrt()* benutzen (<http://php.net/manual/de/function.sqrt.php>).
- Die Methode *absolut()* soll den absoluten Wert einer Zahl berechnen, was Sie über die PHP-Funktion *abs()* bewerkstelligen können.
- Sie können gerne noch weitere Methoden integrieren.

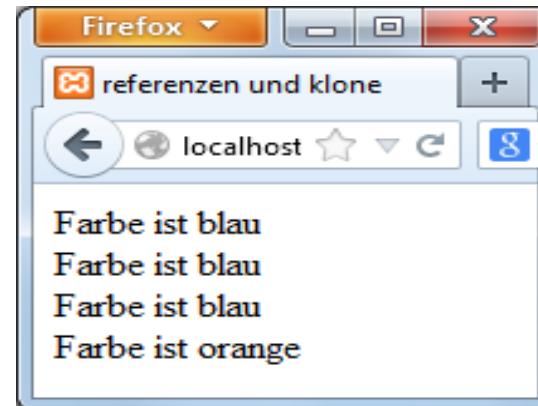
Rufen Sie die Methoden dann auf.

Referenzen und Klonen

- Bei der Zuweisung eines Objekts werden nicht die Werte des Objekts zugewiesen, sondern nur die Referenz auf das Objekt.
- Sollen auch die Werte zugewiesen werden, so ist **clone** Operator anzuwenden.
- Der **clone** Operator erzeugt ein Duplikat eines Objekts im Hauptspeicher

Beispiel

```
class Beispiel {  
    public $farbe;  
    public function info() {  
        echo "Farbe ist {$this->farbe}<br />\n";  
    }  
}  
  
$obj1 = new Beispiel(); //Referenz  
  
$obj2 = $obj1;  
$obj1->farbe = "blau";  
$obj1->info();  
$obj2->info();  
  
$obj3 = clone $obj1;  
$obj1->farbe = "orange";  
$obj3->info();  
$obj2->info();
```



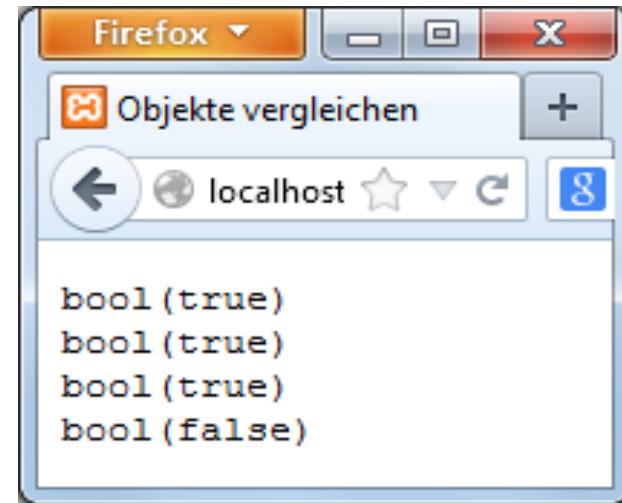
Objekte vergleichen

Zum Vergleichen von Objekten kann man `==` und `====` verwenden. Diese verhalten sich jedoch bei Objekten anders als bei normalen Variablen.

- `==` überprüft, ob die Eigenschaften identisch sind,
- `====` überprüft hingegen, ob der Objekt-Handler identisch ist, also auf dasselbe Objekt verweist.

Beispiel

```
class Beispiel {  
    public $farbe;  
    public function info() {  
        echo "Farbe ist {$this->farbe}<br />\n";  
    }  
}  
  
$obj1 = new Beispiel();  
$obj1->farbe = "blau";  
  
$obj2 = $obj1;  
$obj3 = clone $obj1;  
  
echo "<pre>";  
var_dump($obj1 == $obj2);  
var_dump($obj1 === $obj2);  
  
var_dump($obj1 == $obj3); // gleiche Eigenschaften  
var_dump($obj1 === $obj3); // unterschiedliche Objekte  
echo "</pre>";
```



Objekte verschachteln

```
class BeispielA {  
    public function ausgabe() {  
        echo "Ausgabe aus BeispielA";  
    }  
}  
class BeispielB {  
    public $a;  
    public function __construct() {  
        $this->a = new BeispielA(); // Instanz der Klasse BeispielA  
    }  
}  
  
$b = new BeispielB();  
$b->a->ausgabe(); //Ausgegeben wird: "Ausgabe aus BeispielA"
```

Vererbung

ermöglicht, Basisklassen (mit zusätzlichen Funktionen) zu erweitern.

```
class Fahrzeug {  
    private $geschwindigkeit = 0;  
    function beschleunigen($wert) {  
        $this->geschwindigkeit += $wert;  
    }  
    function ausgabe() {  
        echo "Geschwindigkeit: $this->geschwindigkeit<br />";  
    }  
}  
  
class PKW extends Fahrzeug {  
    private $insassen = 0;  
    function einsteigen($anzahl) {  
        $this->insassen += $anzahl;  
    }  
    function aussteigen($anzahl) {  
        $this->insassen -= $anzahl;  
    }  
    function ausgabe() { // überschriebene Methode  
        echo "Insassen: $this->insassen ";  
        parent::ausgabe(); // geerbte Methode  
    }  
}
```

Aufgabe 8.3:

In einer Übung (Aufgabe 8.1) haben Sie eine Klasse Fahrzeug definiert. Erstellen Sie jetzt dazu eine abgeleitete Klasse namens Auto.

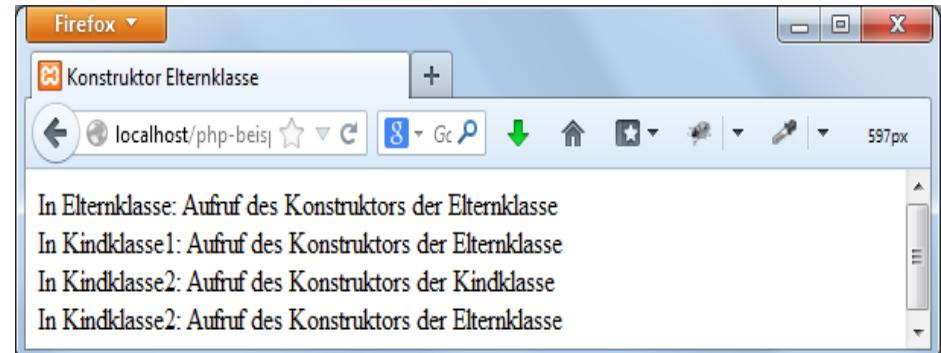
Diese Klasse hat folgende Unterschiede im Vergleich zur Basisklasse:

- Es gibt eine Eigenschaft zur Speicherung des Kilometerstands.
- Es gibt eine Methode zum Fahren, bei der der Kilometerstand um die gefahrenen Kilometer erhöht wird.

Konstruktoren in der Basisklasse und in der abgeleiteten Klasse

Auch der Konstruktor der Elternklasse kann in der abgeleiteten Klasse aufgerufen werden

```
class Elternklasse {  
    public function __construct() {  
        echo "In " . get_class($this);  
        echo ": Aufruf des Konstruktors der Elternklasse<br />\n";  
    }  
}  
  
class Kindklasse1 extends Elternklasse {}  
  
class Kindklasse2 extends Elternklasse {  
    public function __construct() {  
        echo "In " . get_class($this);  
        echo ": Aufruf des Konstruktors der Kindklasse<br />\n";  
        parent::__construct();  
    }  
}  
  
$objekt1 = new Elternklasse();  
$objekt2 = new Kindklasse1();  
$objekt3 = new Kindklasse2();
```



Schnittstellen (*interfaces*)

```
<?php
    interface Fahrbar {
        function rollen();
        function reifenwechseln();
    }
    interface Schwimmfaehig {
        function anlegen();
        function kentern();
    }
    class AmphiCar implements Fahrbar, Schwimmfaehig {
        function rollen() { echo "Es rollt<br />"; }
        function reifenwechseln() { echo "Es werden Reifen gewechselt<br />"; }
        function anlegen() { echo "Es legt an<br />"; }
        function kentern() { echo "Es kentert<br />"; }
        function fahren() { echo "Es fährt<br />"; }
    }
    $VwTyp166 = new AmphiCar();
    $VwTyp166->rollen();
    $VwTyp166->reifenwechseln();
    $VwTyp166->fahren();
    $VwTyp166->kentern();
    $VwTyp166->anlegen();
?>
```



Namensräume

Bei großen Projekten kann es leicht zu Namenskonflikten kommen,

- mehrere Funktionen mit einem vorgegebenen Namen (wenn prozedural programmiert) oder
- mehrere gleichnamigen Klassen (z.B. User) in externen Bibliotheken, die eingebunden sind (wenn objektorientiert programmiert)
- Was ist aber, wenn bei einem Projekt mehrere externe Bibliotheken zum Einsatz kommen, die zwei gleichnamige Klassen haben?

Genau an dieser Stelle kommen die Namensräume ins Spiel – die neben Klassen auch bei Funktionen und Konstanten relevant sind.

Das PHP-Manual vergleicht Namensräume mit Ordnern im Dateisystem.

Namensräume

Zuerst: das Schlüsselwort *namespace*. Danach: die gehörigen Klassen, Funktionen und Konstanten.

```
// Datei: namensraum.php
namespace meinprojekt;
class Benutzer {}
function wastun() {
    echo "getan";
}
const zahl = 42;
```

Die Funktion *wastun()* aus dem Namensraum *meinprojekt* kann benutzt werden

```
require "namensraum.php";
// wastun();      das geht nicht
meinprojekt\wastun(); // das hingegen geht
```

Fehlerbehandlung mit der Exception-Klasse

```
function teilen($x) {  
    if ($x==0) {  
        throw new Exception("Teilen durch 0! ");  
    } else {  
        return 1/$x;  
    }  
}  
  
try {  
    echo teilen(4) . "<br />\n";  
    echo teilen(0) . "<br />\n";  
    echo teilen(5) . "<br />\n";  
} catch (Exception $e) {  
    echo "Exception gefangen: " . $e->getMessage() . "<br />\n";  
}
```



Exceptions mit unterschiedlichen Fehlercodes

```
function teilen($x) {
    if (!is_numeric($x)) {
        throw new Exception("keine Zahl", 1);
    } elseif ($x == 0) {
        throw new Exception("Teilen durch 0!", 2);
    } else {
        return 1/$x;
    }
}

try {
    echo teilen(4) . "<br />\n";
    echo teilen("hallo") . "<br />\n";
} catch (Exception $e) {
    if ($e->getCode() == 2) {
        echo "Falscher Wert: " . $e->getMessage();
    } elseif ($e->getCode() == 1) {
        echo "Falscher Datentyp: " . $e->getMessage();
    }
}
```

Aufgabe 8.4:

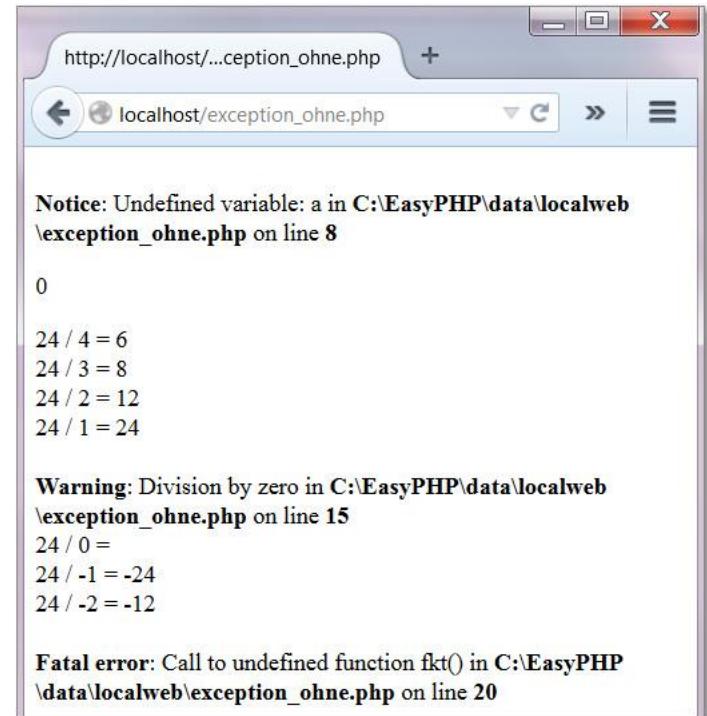
Behandeln Sie die Fehler im folgenden Programm. Ergänzen Sie es mit entsprechenden Codes für die Ausnahmebehandlung. Sie können hierzu die Funktionen *isset* und *function_exists* verwenden.

```
<?php
/* Für dieses Programm alle Fehler anzeigen */
ini_set("error_reporting", 32767);

/* Variable unbekannt */
$c = 2 * $a;
echo "<p>$c</p>";

/* Division durch 0 */
$x = 24;
for($y=4; $y>-3; $y--) {
    $z = $x / $y;
    echo "$x / $y = $z<br />";
}

/* Zugriff auf Funktion */
fkt();
echo "Ende";
?>
```



1 Einleitung: Modulziele, Prüfungsform & Organisatorisches

2 Einführung in Skriptsprachen

3 Einführung in PHP

4 Datentypen

5 Operatoren

6 Kontrollstrukturen

7 Funktionen

8 Objektorientierung

9 Ein- und Ausgabe (Dateien)

10 Web-Formularanbindung

11 Datenbankanbindung

12 Typische Anwendungen

In diesem Abschnitt werden die Funktionen zum

- Öffnen,
- Lesen,
- Schreiben und
- Schließen von Dateien behandelt.

Dateitypen

Bei der Ein- und Ausgabe von Daten in Dateien ist es wichtig zu wissen, welcher Dateityp vorliegt und welche Zugriffsart man verwenden kann.

Man unterscheidet zwischen den folgenden Zugriffsarten:

- **Sequenzieller Zugriff:**
 - *wird bei einer Datei bevorzugt, deren einzelne Zeilen unterschiedlich lang sind und jeweils mit einem Zeilenumbruch beendet werden.*
 - *Die Zeilen werden sequenziell gelesen bzw. geschrieben. Direkter Zugriff auf eine Zeile nicht möglich, da die Längen der Vorgängerzeilen nicht bekannt sind.*
- **Wahlfreier Zugriff:** *(wird hier nicht behandelt!)*
 - *Ist möglich bei einer Datei, die größtenteils gleich lange Datensätze beinhaltet. Es können Zeilenumbrüche existieren, müssen aber nicht.*
 - *Die Datensätze können direkt gelesen bzw. verändert werden, da Sie den Ort jedes Datensatzes berechnen können.*
- **Binärer Zugriff:** *(wird hier nicht behandelt!)*
 - *Ist bei jeder Datei möglich. Man arbeitet mit reinen Byte-Folgen.*

Lesen einer Zeile aus einer sequenziellen Datei

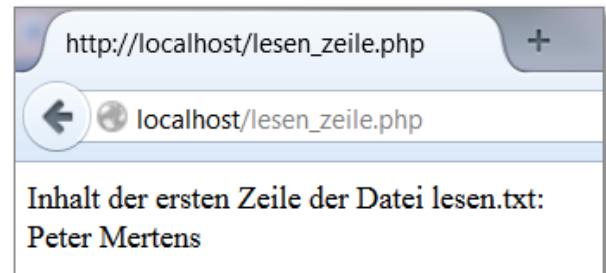
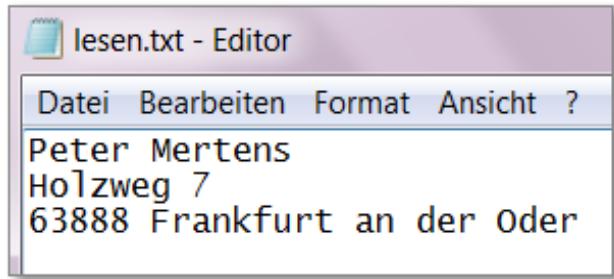
```

<html>
<body>
<?php
    if(!file_exists("lesen.txt"))
        exit("Datei konnte nicht gefunden werden");
    $fp = @fopen("lesen.txt","r"); // Öffnen einer Datei, r: Lesen
                                    // @ : keine Fehlermeldung ausgeben

    if(!$fp)
        exit("Datei konnte nicht geöffnet werden");

    $zeile = fgets($fp, 100);    // Datei, Länge der Zeichenkette
    echo "Inhalt der ersten Zeile der Datei lesen.txt:<br />$zeile";
    fclose($fp);
?>
</body>
</html>

```



Aus einer sequenziellen Datei lesen

- **fopen()** dient dem Öffnen einer Datei:
 - **@ (Silence-Operator):** sorgt dafür, dass keine Fehlermeldung auf dem Bildschirm erscheint, falls beim Ausführen der Funktion ein Fehler auftritt.
- **fgets()** dient dem Lesen einer Zeichenkette aus einer Datei.

Beschreibung im Manual (<http://php.net/manual/de/function.fgets.php>):

string fgets (resource \$handle [, int \$length])

- 1. Parameter: die Datei.
- 2. Parameter (optional): die Länge der einzulesenden Zeichenkette
- Es werden
 - entweder (Länge -1) Zeichen aus der Datei gelesen (hier 99)
 - oder bis zum Zeilenumbruch (Zeilenumbruch im Rückgabewert enthalten)
 - oder bis zum Ende der Datei.
- Dies gilt je nachdem, was zuerst eintritt.
- Der Rückgabewert der Funktion ist die gelesene Zeichenkette.
- **fclose()** dient dem Schließen einer Datei

Aus einer sequenziellen Datei lesen

Modus, Funktion, Dateizeiger, Datei anlegen

`$datei = fopen(Dateiname, Modus)`

Modus	Funktion	Dateizeiger	Datei anlegen
r	Lesen	Anfang	Nein
r+	Lesen und Schreiben	Anfang	Nein
w	Schreiben	Anfang	Ja
w+	Lesen und Schreiben	Anfang	Ja
a	Schreiben	Ende	Ja
a+	Lesen und Schreiben	Ende	Ja

Lesen aller Zeilen einer sequenziellen Datei

```
<html>
<body>
<?php
    if(!file_exists("lesen.txt"))
        exit("Datei konnte nicht gefunden werden");

    $fp = @fopen("lesen.txt","r");
    if(!$fp)
        exit("Datei konnte nicht geöffnet werden");

    while (!feof($fp)) {      // feof() zeigt das Ende einer Datei an
        $zeile = fgets($fp, 100);
        echo "Zeile: $zeile<br />";
    }

    fclose($fp);
?>
</body>
</html>
```



Hinweis

Im vorliegenden Fall wurde vorausgesetzt, dass die Datei *lesen.txt* mit einem Texteditor in der folgenden Weise erzeugt wurde:

Schreiben der ersten Zeile (Return-Taste)

Schreiben der zweiten Zeile (Return-Taste)

...

Schreiben der vorletzten Zeile (Return-Taste)

Schreiben der letzten Zeile (Abspeichern und Schließen der Datei)

(ohne Return-Taste)

Wenn Sie nach der letzten Zeile noch einen oder mehrere Zeilenumbrüche erzeugt und die Datei anschließend gespeichert und geschlossen haben, werden beim Lesen diese unnötigen leeren Zeilen ebenfalls erfasst und können zu Fehlausgaben führen (z.B. einer falschen Zeilenanzahl).

Vereinfachtes Lesen einer Datei

Die Funktionen `readfile()` und `file()` liefern eine Möglichkeit zum vereinfachten Lesen einer Datei:

- `readfile()` liest den vollständigen Inhalt einer Datei und gibt ihn auf dem Bildschirm aus.
 - Zeilenumbrüche werden im generierten HTML-Code als Zeilenumbrüche notiert.

```
if(!file_exists("lesen.txt"))
    exit("Datei konnte nicht gefunden werden");
readfile("lesen.txt");
```

- `file()` liest den vollständigen Inhalt einer Datei zeilenweise in ein eindimensionales Feld ein.
 - Die Elemente dieses Felds können anschließend bearbeitet werden.

```
$dfeld = file("lesen.txt");
for($i=0; $i<count($dfeld); $i++)
    echo $dfeld[$i] . "<br />";
```

Überschreiben einer sequenziellen Datei

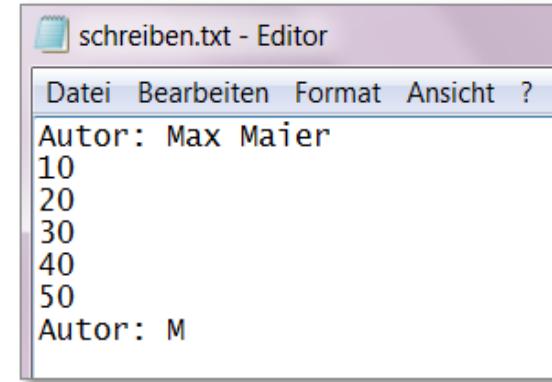
```
$fp = @fopen("schreiben.txt", "w"); // Öffnungsmodus w (für Write)
if (!$fp)
    exit("Datei konnte nicht zum Schreiben geöffnet werden");

fputs ($fp, "Autor: Max Maier\n"); // Zeichenausgabe in Datei

for ($i=10; $i<=50; $i+=10)
    fputs ($fp, "$i\n");

fputs ($fp, "Autor: Max Maier", 8);

echo "Ausgabe in Datei geschrieben";
fclose($fp);
```

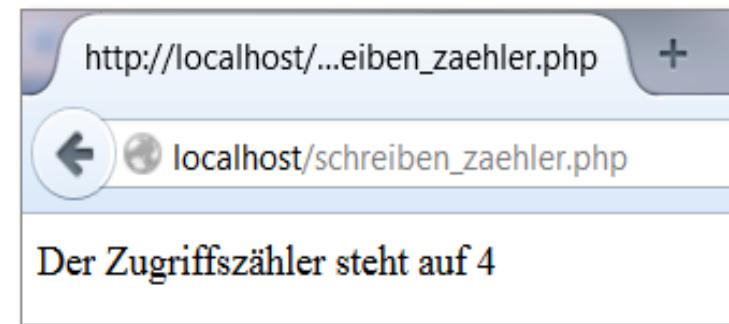


- **fputs()** dient der Ausgabe von Zeichenketten in eine Datei:
 - 1. Parameter: die Datei, in welche ausgegeben werden soll.
 - 2. Parameter: die auszugebende Zeichenkette.
 \n: für die Erstellung eines Zeilenumbruchs
 - 3. Parameter (optional): die Länge der Zeichenausgabe (hier: 8 Zeichen) .

Beispiel: ein einfacher Zugriffszähler

Zugriffszähler hält die Anzahl der Zugriffe auf eine Datei fest. Ein solcher Webcounter wird oft eingesetzt, um die Beliebtheit einer Webseite anzuzeigen.

```
if(file_exists("schreiben_zahler.txt")) {  
    $fp = @fopen("schreiben_zahler.txt", "r");  
    if($fp) {  
        $zahl = fgets($fp, 10);  
        fclose($fp);  
    }  
    else  
        $zahl = 0;  
}  
else  
    $zahl = 0;  
  
$zahl = $zahl + 1;  
echo "Der Zugriffszähler steht auf $zahl";  
$fp = @fopen("schreiben_zahler.txt", "w");  
if(!$fp)  
    exit("Der Zähler kann nicht geschrieben werden");  
fputs($fp, $zahl);  
fclose($fp);
```



Aufgabe 9.1:

Schreiben Sie mithilfe eines Texteditors mehrere Namen in eine Datei (*u_lesen.txt*). Jeder Name soll zweizeilig geschrieben werden: in der ersten Zeile der Vorname und in der zweiten Zeile der Nachname (wie die folgende Abbildung zeigt).

The screenshot shows a Windows Notepad window with the title bar 'u_lesen.txt - Editor'. The menu bar includes 'Datei', 'Bearbeiten', 'Format', 'Ansicht', and '?'. The content area contains the following text:
Peter
Mertens
Josef
Huber
Julia
Weber

Erstellen Sie ein PHP-Programm, das diese Datei öffnet und liest sowie die Namen (zusätzlich mit laufender Nummer) in einer HTML-Tabelle auf dem Bildschirm ausgibt (wie die folgende Abbildung zeigt).

The screenshot shows a web browser window with the URL 'http://localhost/u_lesen.php' in the address bar. The page content is an HTML table with the following data:

Nummer	Nachname	Vorname
1	Mertens	Peter
2	Huber	Josef
3	Weber	Julia

Aufgabe 9.2:

In einer Datei (*u_schreiben.txt*) stehen mehrere Datensätze in der folgenden Form (siehe Abbildung):

- Erste Zeile: laufende Nummer
- Zweite Zeile: Vorname
- Dritte Zeile: Nachname

u_schreiben.txt - Editor	
Datei	Bearbeiten
Format	Ansicht
1883	
Peter	
Mertens	
762	
Josef	
Huber	
2655	
Julia	
Weber	
699	
Wolfgang	
Schmitz	

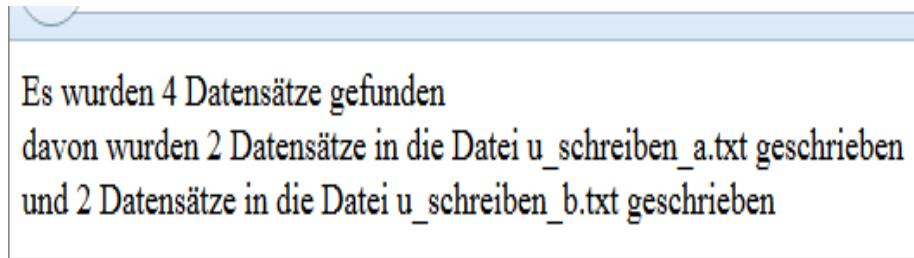
Schreiben Sie ein PHP-Programm, das diese Datei öffnet und liest sowie die Datensätze in der gleichen Form in zwei verschiedenen Dateien ausgibt:

- Datensätze mit einer laufenden Nummer kleiner als 1.000 in der Datei *u_schreiben_a.txt*
- alle anderen Datensätze in der Datei *u_schreiben_b.txt*



Aufgabe 9.2 (Fortsetzung):

Die Dateien sollen bei jedem Programmaufruf überschrieben werden. Auf dem Bildschirm soll eine Kontrollausgabe wie in folgender Abbildung erfolgen (hier mit den oben angegebenen Beispieldaten).



Hinweis: Die Funktion `fgets()` liest die Zeilen einschließlich des Zeilenumbruchs in eine Variable ein. Wird diese Variable in eine Datei ausgegeben, so wird auch der Zeilenumbruch ausgeführt. Er braucht nicht zusätzlich ausgegeben zu werden

1 Einleitung: Modulziele, Prüfungsform & Organisatorisches

2 Einführung in Skriptsprachen

3 Einführung in PHP

4 Datentypen

5 Operatoren

6 Kontrollstrukturen

7 Funktionen

8 Objektorientierung

9 Ein- und Ausgabe (Dateien)

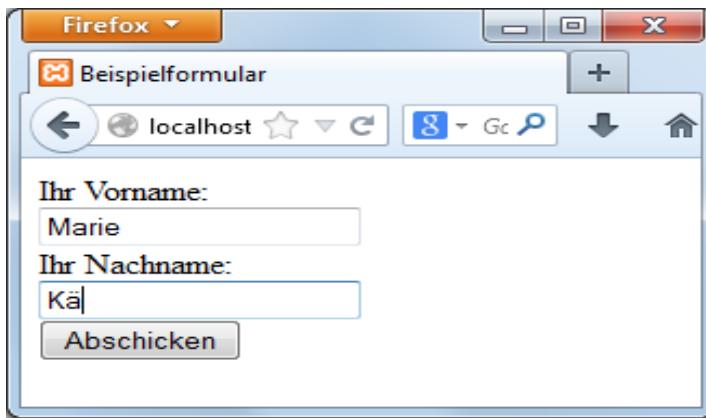
10 Web-Formularanbindung

11 Datenbankanbindung

12 Typische Anwendungen

Formulare

- Formulare sind eine zentrale Komponente von Webseiten, weil sie die komfortabelste Möglichkeit sind, Input von Benutzern zu erhalten.
- HTML stellt die Formularfelder zur Verfügung,
- aber für die Weiterverarbeitung braucht man PHP oder eine andere serverseitige Skriptsprache.
- In diesem Kapitel wird gezeigt, wie man Formulare erstellt und mit PHP auf die Daten zugreift.



// Datei: Formular.html

```
01 <form action="verarbeitung.php" method="get">
02 Ihr Vorname: <br />
03 <input type="text" name="vorname" size="20" maxlength="30" />
04 <br />
05 Ihr Nachname: <br />
06 <input type="text" name="nachname" size="20" maxlength="30" />
07 <br />
08 <input type="submit" value="Abschicken" />
09 </form>
```

Im Beispielformular werden mehrere HTML-Elemente eingesetzt:

- Alle Formularelemente werden innerhalb von `<form>` und `</form>` geschrieben.
- Im `form`-Starttag ist angegeben, was mit den Formulardaten geschehen soll und wie es geschehen soll:
 - Bei `action`: der Pfad zu einem Skript, das die Verarbeitung übernimmt.
 - Bei `method`: die Methode, wie die Formulardaten versendet werden können: GET oder POST (Genaues zu den Unterschieden später)
- Der Text in Zeile 2 und 5 ist die Beschriftung der Formularfelder
- In Zeilen 3 & 6 Textfelder erzeugt: `input`-Element mit dem Attribut `type="text"`.
 - Wichtig: dem Formularfeld einen Namen (`name`) zu geben. Über diesen kann man auf den Inhalt des Formularfelds zugreifen.
 - `size` = die sichtbare Größe und
 - `maxlength` = max. mögliche Zeichenlänge
- Zeile 8 erstellt den Absendebutton über ein `input`-Element mit `type="submit"`.
 - Der bei `value` angegebene Wert wird als Beschriftung im Browser angezeigt.

Mit PHP auf die Inhalte zugreifen:

PHP stellt alle Formulardaten in zwei assoziativen Arrays zur Verfügung:

- `$_POST` für per POST und `$_GET` für per GET versendete Formulardaten.
- `$_POST` und `$_GET` sind superglobal, d.h., Zugriff auf diese innerhalb von Funktionen möglich, ohne *global* zu benutzen
- Zugriff auf den Inhalt eines bestimmten Formularfelds über den Namen des Formularfelds als Schlüssel.
 - z.B.: `$_GET["vorname"]` und `$_GET["nachname"]`

In unserem Formular hatten wir auf die Datei *verarbeitung.php* verwiesen, die beim Abschicken des Formulars aufgerufen wird:

```
01 <form action="verarbeitung.php" method="get">
```

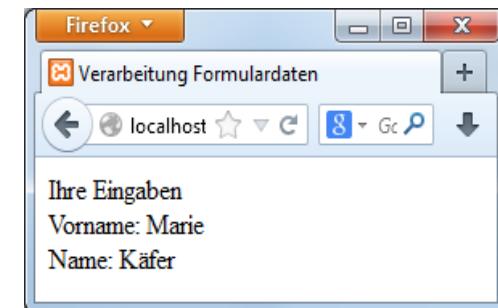
Datei verarbeitung.php zur Ausgabe des Inhaltes der Formularfelder

```
// Datei: verarbeitung.php

echo "Ihre Eingaben<br />\n";
echo "Vorname: {" . $_GET['vorname'] . "}<br />\n";
echo "Name: {" . $_GET['nachname'] . "}<br />\n";
```

Ergänzung:

- man muss alle Inhalte von Formularfeldern *bei der Ausgabe mit htmlspecialchars() behandeln*
- *htmlspecialchars()* wandelt Sonderzeichen in HTML-Codes um



```
echo "Ihre Eingaben<br />\n";
echo "Vorname: " . htmlspecialchars($_GET["vorname"]) . "<br />\n";
echo "Name: " . htmlspecialchars($_GET["nachname"]) . "<br />\n"
```

Aufgabe 10.1:

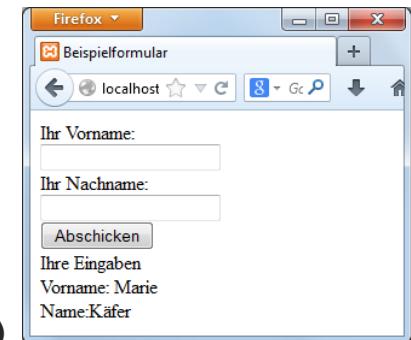
Ergänzen Sie das folgende Beispiel durch zwei weitere Felder für eine E-Mail-Adresse und eine Telefonnummer. Die Eingaben dieser beiden Felder sollen ebenfalls ausgegeben werden.

```
<form action="verarbeitung.php" method="get">  
Ihr Vorname: <br />  
<input type="text" name="vorname" size="20" maxlength="30" />  
<br />  
Ihr Nachname: <br />  
<input type="text" name="nachname" size="20" maxlength="30" />  
<br />  
<input type="submit" value="Abschicken" />  
</form>
```

- Häufig werden das Formular & die Verarbeitung in demselben Skript untergebracht
- Dafür: bei *action* im form-Starttag den Namen des aktuellen Skripts anzugeben.
- Vor der Ausgabe mit *isset()* überprüfen, ob das Formular abgesendet wurde.

formular_ausgabe.php

```
01 <form action="formular_ausgabe.php" method="get">
02 Ihr Vorname: <br />
03 <input type="text" name="vorname" size="20" maxlength="30" />
04 <br />
05 Ihr Nachname: <br />
06 <input type="text" name="nachname" size="20" maxlength="30" />
07 <br />
08 <input type="submit" value="Abschicken" />
09 </form>
10 <?php
11 if (isset($_GET["vorname"])) {
12 echo "Ihre Eingaben<br />\n"
13 . "Vorname: " . htmlspecialchars($_GET["vorname"])
14 . "<br />\n Name: " . htmlspecialchars($_GET["nachname"])
15 . "<br />\n";
16 }
17 ?>
```



- Im Beispiel soll beim Absenden das Skript selbst aufgerufen werden. Deswegen ist der Name der aktuellen Datei bei action eingetragen (Zeile 1).
 - Unpraktisch, wenn das Skript unter einem neuen Namen abgespeichert wird.
- Man kann auch von PHP selbst den Namen des Skripts ermitteln lassen, der in der Variablen `$_SERVER["PHP_SELF"]` steht.

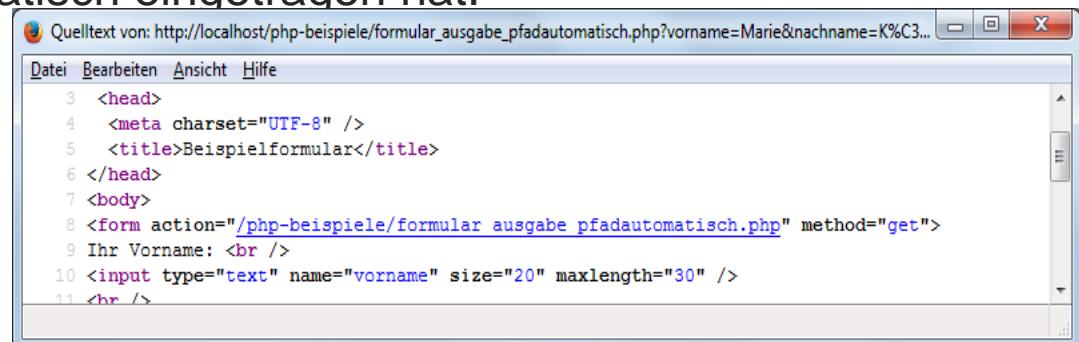
```
<form action="<?php echo $_SERVER ["PHP_SELF"] ;?>" method="get">
```

- Außerdem sollten Sie auch hier `htmlspecialchars()` benutzen:

```
<form action="<?php echo  
htmlspecialchars($_SERVER ["PHP_SELF"]);?>" method="get">
```

Testen Sie damit einmal Ihr Skript.

- Wenn Sie in den Quellcode wechseln, sehen Sie, dass PHP Ihnen den richtigen Pfad bei action automatisch eingetragen hat.



The screenshot shows a Windows Notepad window with the following content:

```
Quelltext von: http://localhost/php-beispiele/formular_ausgabe_pfadautomatisch.php?vorname=Marie&nachname=K%C3...  
Datei Bearbeiten Ansicht Hilfe  
3 <head>  
4 <meta charset="UTF-8" />  
5 <title>Beispielformular</title>  
6 </head>  
7 <body>  
8 <form action="/php-beispiele/formular_ausgabe_pfadautomatisch.php" method="get">  
9 Ihr Vorname: <br />  
10 <input type="text" name="vorname" size="20" maxlength="30" />  
11 <br />
```

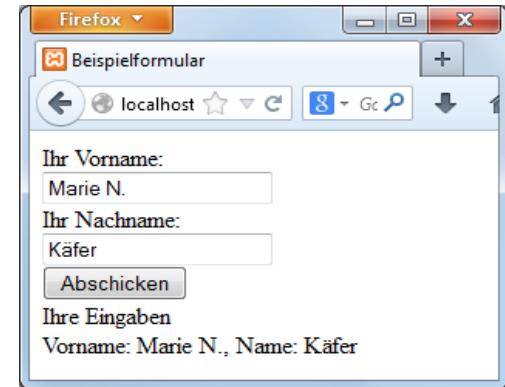
Formular oder Auswertung anzeigen

- verhindern, dass bei der Auswertung das Formular selbst erneut angezeigt wird:

```
01 <?php
02 if (!isset($_GET["vorname"])) {
03 ?>
04 <form action=<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>
    method="get">
05 Ihr Vorname: <br />
06 <input type="text" name="vorname" size="20" maxlength="30" />
07 <br />
08 Ihr Nachname: <br />
09 <input type="text" name="nachname" size="20" maxlength="30" />
10 <br />
11 <input type="submit" value="Abschicken" />
12 </form>
13 <?php
14 } else {
15     echo "Ihre Eingaben<br />\n"
16     . "Vorname: " . htmlspecialchars($_GET["vorname"])
17     . "<br />\n Name: " . htmlspecialchars($_GET["nachname"])
18     . "<br />\n";
19 }
20 ?>
```

Eingegebene Werte wieder eintragen

```
<?php
    if (isset($_GET["vorname"])) {
        $vorname = htmlspecialchars($_GET["vorname"]);
        $nachname = htmlspecialchars($_GET["nachname"]);
    } else {
        $vorname = "";
        $nachname = "";
    }
?>
<form action=<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>" method="get">
Ihr Vorname: <br />
<input type="text" name="vorname" size="20" maxlength="30"
           value=<?php echo $vorname; ?>" />
<br />
Ihr Nachname: <br />
<input type="text" name="nachname" size="20" maxlength="30"
           value=<?php echo $nachname; ?>" />
<br />
<input type="submit" value="Abschicken" />
</form>
<?php
    if (isset($_GET["vorname"])) {
        echo "Ihre Eingaben<br />\n";
        echo "Vorname: $vorname, Name: $nachname<br />\n";
    }
?>
```



- Bisher wurde GET als Übertragungsmethode eingesetzt. Wenn man das auf POST ändert, müssen auch die PHP-Variablen zum Zugriff auf die Formulareingaben angepasst werden.

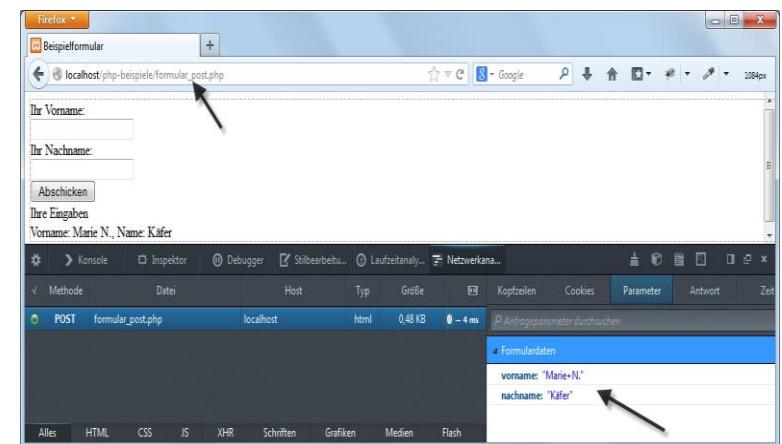
```
01 <form action="<?php echo  
    htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="post">  
02 <!-- Rest des Formulars wie gehabt --></form>  
03 <?php  
04 if (isset($_POST["vorname"])) {  
05     echo "Ihre Eingaben<br />\n"  
06     . "Vorname: " . htmlspecialchars($_POST["vorname"])  
07     . ", Name: " .htmlspecialchars($_POST["nachname"])  
08     . "<br />\n";  
09 }  
10 ?>
```

Der wesentliche Unterschied zwischen GET und POST:

- bei GET werden die Formulardaten über die URL übertragen. Ein Benutzer kann sie bspw. in die Lesezeichen aufnehmen, und die Parameter werden mit aufgenommen.
 - **Daten sind sichtbar. Zur Versendung von Login-Daten samt Passwort wäre GET eindeutig die falsche Wahl.**



- Bei der Datenübertragung per POST werden die Formulardaten nicht über die URL, sondern im Dokumentkörper mit übertragen.
 - **Sie sind auf normalem Wege nicht sichtbar.**
- Prinzipiell gilt:
 - **Für viele oder auch sensible Daten POST verwenden, ansonsten GET.**
 - **Trotzdem sind per POST versendete Daten nicht an sich geschützt – um sie zu schützen, kann man sie mit HTTPS verschlüsselt übertragen.**



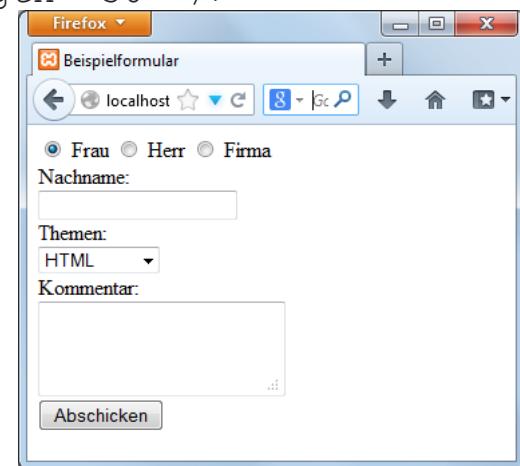
Aufgabe 10.2:

Modifizieren Sie das Beispiel aus Aufgabe 10.1 so, dass die folgenden Bedingungen erfüllt werden:

- Das Formular und die Auswertung des Formulars sollen innerhalb desselben Skripts stehen.
- Die Daten sollen per POST gesendet werden
- Außerdem soll entweder die Ausgabe oder das Formular angezeigt werden.

Radiobuttons, Auswahllisten und mehrzeilige Textfelder

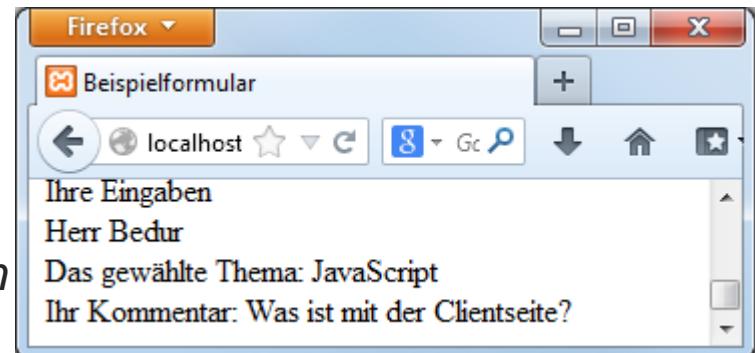
```
01 <form action=<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>"  
           method="get">  
02 <input type="radio" name="anrede" value="Frau" checked="checked"/> Frau  
03 <input type="radio" name="anrede" value="Herr" /> Herr  
04 <input type="radio" name="anrede" value="Firma" /> Firma <br />  
05 Nachname: <br />  
06 <input type="text" name="nachname" size="20" maxlength="30" />  
07 <br />  
08 Themen: <br />  
09 <select name="thema">  
10     <option value="HTML">HTML</option>  
11     <option value="CSS">CSS</option>  
12     <option value="JavaScript">JavaScript</option>  
13     <option value="PHP">PHP</option>  
14 </select>  
15 <br />  
16 Kommentar: <br />  
17 <textarea name="kommentar" rows="3" cols="20"></textarea>  
18 <br />  
19 <input type="submit" value="Abschicken" />  
20 </form>
```



//Nun zur Auswertung per PHP, die direkt darunter erfolgt:

```
21 <?php
22 if (!empty($_GET["nachname"])) {
23     echo "Ihre Eingaben<br />\n";
24     if (!empty($_GET["anrede"])) {
25         echo htmlspecialchars($_GET["anrede"]);
26     }
27     echo " " . htmlspecialchars($_GET["nachname"]) . "<br />\n";
28     if (!empty($_GET["thema"])) {
29         echo "Das gewählte Thema: " . htmlspecialchars($_GET["thema"]) .
30             "<br />\n ";
31     }
32     if (!empty($_GET["kommentar"])) {
33         echo "Ihr Kommentar: " . htmlspecialchars($_GET["kommentar"]);
34     }
35 ?>
```

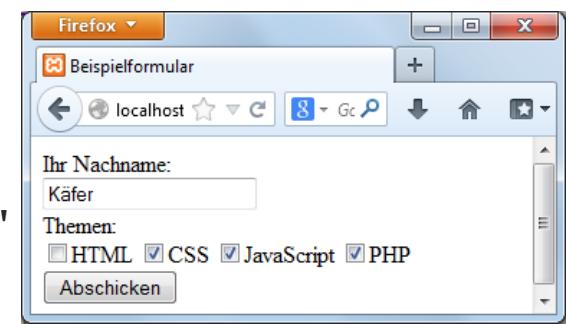
*Eine Beispieldausgabe
von eingegebenen Daten*



Checkboxen

- Bei Checkboxen kann man mehrere gleichzeitig ankreuzen.
- Damit PHP die Werte aller angekreuzten Checkboxen als Array abspeichert, müssen Sie beim Namen eckige Klammern schreiben.

```
01 <form action="<?php echo '  
        htmlspecialchars($_SERVER["PHP_SELF"]);?>"  
        method="get">  
02 Ihr Nachname: <br />  
03 <input type="text" name="nachname" size="20" maxlength="30" />  
04 <br />  
05 Themen: <br />  
06 <input type="checkbox" name="thema[]" value="HTML" />HTML  
07 <input type="checkbox" name="thema[]" value="CSS" />CSS  
08 <input type="checkbox" name="thema[]" value="JavaScript"  
                           />JavaScript  
09 <input type="checkbox" name="thema[]" value="PHP" />PHP<br />  
10 <input type="submit" value="Abschicken" />  
11 </form>
```



- Es folgt wieder die Auswertung per PHP:

```
12 if (!empty($_GET["nachname"])) {  
13     echo "Ihre Eingaben<br />\n"  
14     . "Name: ". htmlspecialchars($_GET["nachname"]) . "<br />\n";  
15     if (isset($_GET["thema"]) && is_array($_GET["thema"])) {  
16         echo "Die gewählten Themen: <br />\n ";  
17         foreach($_GET["thema"] as $th) {  
18             echo htmlspecialchars($th) . "<br />\n";  
19         }  
20     }  
21 }
```

Eine Ausgabe mit *print_r()* zeigt den Aufbau des verschachtelten *\$_GET*-Arrays:

```
25 Array  
26 ( [nachname] => Käfer  
27     [thema] => Array  
28         ( [0] => CSS  
29             [1] => JavaScript  
30                 [2] => PHP  
31             )  
32         )  
33     )  
34 )  
35 )
```

Aufgabe 10.3:

Das folgende Programm mit einer Funktion, die zu einem Nettobetrag den Bruttobetrag berechnet, soll so verändert werden, dass ein Benutzer den Nettobetrag über ein Formular eingeben kann (s. Abbildung). Außerdem kann er über eine Auswahlliste den Mehrwertsteuersatz wählen. Nach Absenden des Formulars findet die Umrechnung statt, und das Ergebnis wird ausgegeben.

```
// programm  
  
function brutto2($netto, $mwstSatz) {  
    return $netto * (100 + $mwstSatz) / 100;  
}  
  
$betrag = 23;
```

```
$bruttowert = brutto2($betrag, 7);
```

```
echo "$betrag ergibt $bruttowert inkl. MWSt<br />\n";
```

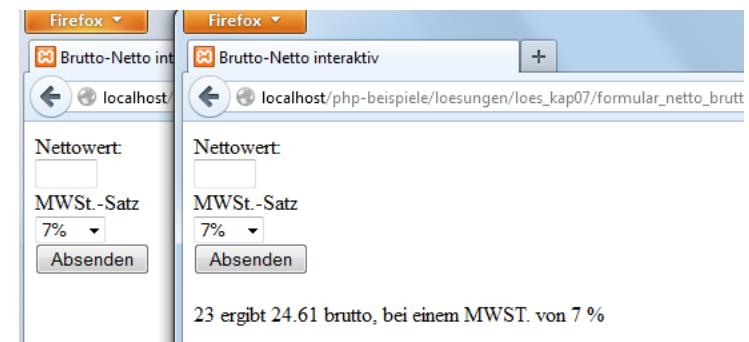
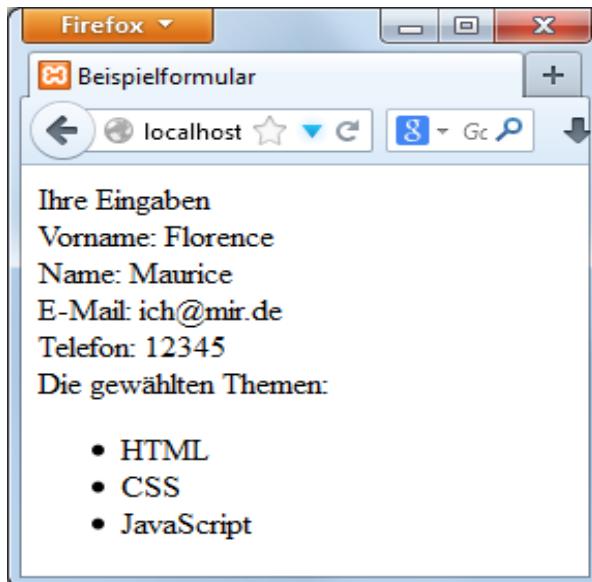


Abbildung: Brutto-Netto-Rechner interaktiv

```
$bruttowert2 = brutto2($betrag, 19);  
echo "$betrag ergibt $bruttowert2 inkl. MWSt<br />\n";
```

Aufgabe 10.4:

Ergänzen Sie beim Formular aus Aufgabe 10.2 Checkboxen, beispielsweise zur Auswahl von interessanten Themen. Lassen Sie die gewählten Themen dann als ungeordnete Liste ausgeben!



1 Einleitung: Modulziele, Prüfungsform & Organisatorisches

2 Einführung in Skriptsprachen

3 Einführung in PHP

4 Datentypen

5 Operatoren

6 Kontrollstrukturen

7 Funktionen

8 Objektorientierung

9 Ein- und Ausgabe (Dateien)

10 Web-Formularanbindung

11 Datenbankanbindung

12 Typische Anwendungen

Datenbankanbindung

- Im Zusammenhang mit der Programmiersprache PHP wird häufig mit MySQL-Datenbanken gearbeitet.
- In diesem Kapitel lernen Sie
 - die Struktur einer Datenbank
 - die Benutzeroberfläche phpMyAdmin und
 - das Zusammenspiel von PHP-Programmen mit MySQL kennen.

Datenbank

Eine Datenbank

- dient der Speicherung größerer Datenmengen und der übersichtlichen Darstellung bestimmter Daten aus diesen Datenmengen.
- beinhaltet verschiedene Tabellen.
 - Die Begriffe in der ersten Zeile nennt man die Datenfelder der Tabelle. Anschließend folgen die einzelnen Datensätze
 - Die Datenfelder haben bestimmte Datentypen (hier: Texte, Zahlen und Datumsangaben)

Name	Vorname	Personal-nummer	Gehalt	Geburts-tag
Maier	Hans	6714	3500,00	15.03.62
Schmitz	Peter	81343	3750,00	12.04.58
Mertens	Julia	2297	3621.50	30.12.59

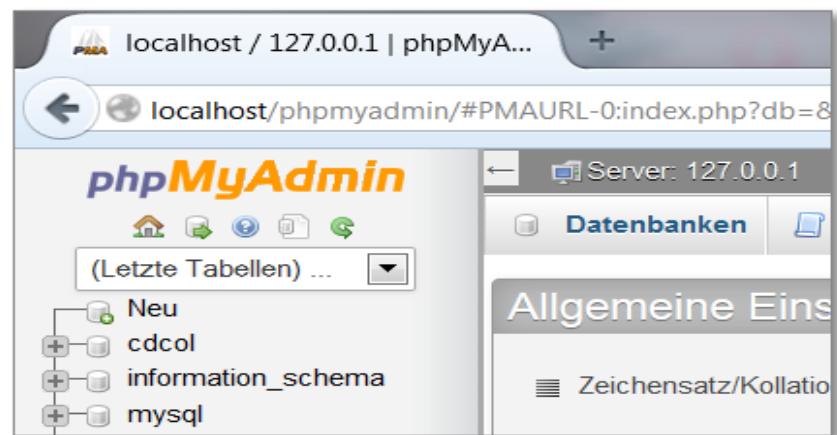
Vorgehensweise beim Erzeugen einer Datenbank

- Anlegen der Datenbank
- Anlegen von Tabellen durch Angabe der Struktur
- Eingeben der Datensätze in die Tabellen

MySQL und phpMyAdmin

MySQL und phpMyAdmin

- MySQL ist die Open-Source-Datenbank mit der größten Verbreitung.
- MySQL bietet SQL-Anweisungen zum Erzeugen einer Struktur von Datenbanken und Tabellen sowie zum Bearbeiten der Datensätze
- In der PHP-Welt wird zur Erzeugung der Struktur von MySQL-Datenbanken und Tabellen häufig die Bedienoberfläche *phpMyAdmin* verwendet
 - Bestandteil der Installationen mit XAMPP
- Mithilfe von phpMyAdmin können Sie unter anderem Datenbanken, Tabellen, Felder und eindeutige Indizes anlegen, verwalten und löschen.
- die Startseite von php-MyAdmin
[http://localhost/phpmyadmin.](http://localhost/phpmyadmin)



Beispieldatenbank und -tabelle

- Es soll eine Datenbank *firma* mit einer Datenbanktabelle *personen* erzeugt werden.
- In dieser Datenbanktabelle sollen die Daten zu einzelnen Personen gespeichert werden

Feldname	Datentyp
name	varchar(30)
vorname	varchar(25)
personalnummer	int
gehalt	double
geburtstag	date

Aufbau der Datenbanktabelle »personen«

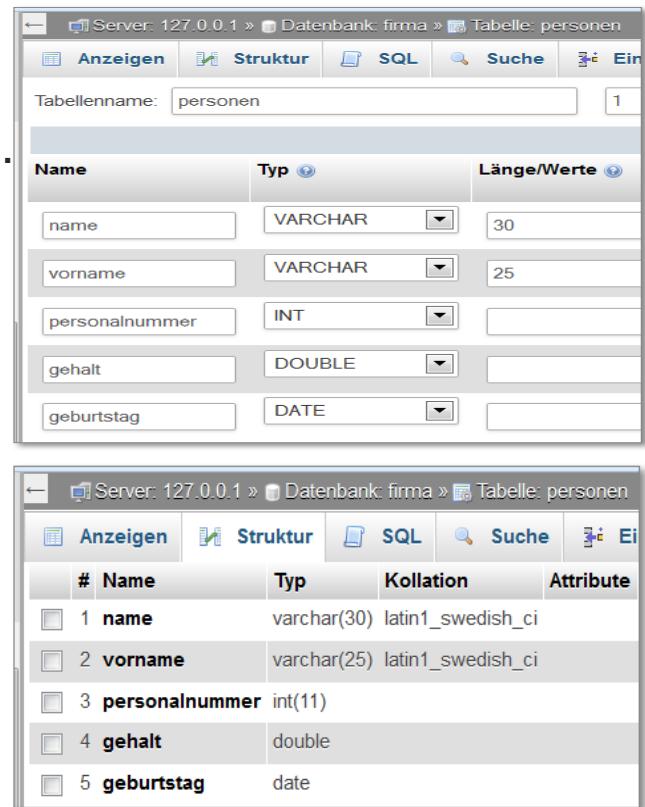
Datenbank und Tabelle erzeugen

Datenbank erzeugen

- auf die Registerkarte Datenbanken wechseln und die Datenbank *firma* anlegen.
- Bei den Namen von Datenbanken, Tabellen und Feldern:
 - keine Umlaute, kein ß (scharfes S), keine Leerzeichen oder Sonderzeichen

Tabelle erzeugen

- Auf den Namen der neu erzeugten Datenbank klicken.
 - es erscheint eine Seite mit der Überschrift **ERZEUGE TABELLE**.
- den Namen der Tabelle (personen) und die Anzahl der Spalten (5) eingeben.
 - es erscheint eine Seite zum Anlegen einer Tabelle mit fünf Spalten (sprich Datenbankfeldern)
- die Namen & Datentypen der Felder eintragen & speichern. Nun wird die Tabelle erzeugt.
- Nach Betätigen des Hyperlinks neben dem Namen der Tabelle erscheint die Tabellenstrukturansicht.



The screenshot shows two windows of a database management interface. The top window displays the table structure for 'Tabelle: personen' with columns: name (VARCHAR(30)), vorname (VARCHAR(25)), personalnummer (INT), gehalt (DOUBLE), and geburtstag (DATE). The bottom window shows the table structure in a grid view with columns: #, Name, Typ, Kollation, and Attribute. The columns are: 1 name, varchar(30), latin1_swedish_ci; 2 vorname, varchar(25), latin1_swedish_ci; 3 personalnummer, int(11); 4 gehalt, double; and 5 geburtstag, date.

#	Name	Typ	Kollation	Attribute
1	name	varchar(30)	latin1_swedish_ci	
2	vorname	varchar(25)	latin1_swedish_ci	
3	personalnummer	int(11)		
4	gehalt	double		
5	geburtstag	date		

Primärschlüssel erzeugen

- eine eindeutige Identifizierung der einzelnen Datensätze wird mithilfe eines Primärschlüssels realisiert.
- Z.B. das Feld *personalnummer* soll eindeutig sein und wird mit einem Primärschlüssel versehen.
 - Betätigen Sie dazu in der Tabellenstrukturansicht in der Zeile des Felds *personalnummer* den Link Primärschlüssel.
 - Lassen Sie sich die Tabellenstrukturansicht neu anzeigen.
 - Das Feld *personalnummer* ist nun in der Ansicht unterstrichen, und weiter unten erscheint im Bereich Indizes die Information über den Schlüssel mit dem Namen PRIMARY.
- Alle Datensätze können anhand des Werts *personalnummer* eindeutig voneinander unterschieden werden.
- Dies ist besonders beim Ändern und Löschen von Datensätzen wichtig.

Datensätze eintragen

- Auf der Registerkarte EINFÜGEN

Server: 127.0.0.1 » Datenbank: firma » Tabelle: personen

Spalte	Typ	Funktion	Null	Wert
name	varchar(30)			Maier
vorname	varchar(25)			Hans
personalnummer	int(11)			6714
gehalt	double			3500
geburtstag	date			1962-03-15

- Beim Eintragen von Datensätzen Folgendes beachten:
 - Bei Zahlen mit Nachkommastellen wird der Punkt anstelle des Kommas verwendet.
 - Bei Datumsangaben gilt das amerikanische Eingabeformat JJJJ-MM-TT.
 - Nach Eingabe aller Daten kann man den Hyperlink auf dem Namen der Tabelle **personen** betätigen und alle Datensätze sehen

name	vorname	personalnummer	gehalt	geburtstag
Maier	Hans	6714	3500	1962-03-15
Schmitz	Peter	81343	3750	1958-04-12
Mertens	Julia	2297	3621.5	1959-12-30

Aufgabe 11.1:

Legen Sie eine Datenbank mit dem Namen *hardware* an. Sie soll eine Tabelle *fp* mit der Struktur aus Abbildung 1 und den Daten aus Abbildung 2 enthalten. Das Feld *artnummer* soll einen Primärschlüssel haben. Die Datenbank enthält Hardware-Informationen, die Tabelle enthält Informationen zu Festplatten. Die Felder *hersteller*, *typ* und *artnummer* sind vom Typ Zeichenkette, das Feld *gb* gibt die Kapazität der Festplatte an, und das Feld *prod* gibt das erste Produktionsdatum der betreffenden Festplatte an.

The screenshot shows the MySQL Workbench interface with the following details:
- Server: 127.0.0.1
- Datenbank: hardware
- Tabelle: fp
The table structure is displayed in a grid with columns: #, Name, Typ, Kollation, Attribute, Null.
- Row 1: hersteller, varchar(25), latin1_swedish_ci, Ja
- Row 2: typ, varchar(25), latin1_swedish_ci, Ja
- Row 3: gb, int(11), , Ja
- Row 4: preis, double, , Ja
- Row 5: artnummer, varchar(15), latin1_swedish_ci, Nein
- Row 6: prod, date, , Ja

Abbildung 1: Tabelle "fp" Struktur

hersteller	typ	gb	preis	artnummer	prod
Quantum	Fireball CX	40	112	HDA-208	2008-10-01
Quantum	Fireball Plus	80	128	HDA-163	2008-03-15
Fujitsu	MPE 3136	160	149	HDA-171	2008-09-01
Seagate	310232A	60	122	HDA-144	2008-11-15
IBM Corporation	DJNA 372200	240	230	HDA-140	2008-06-15

Abbildung 2: Tabelle "fp" Daten

PHP und MySQL

- Die Datenbank und der Strukturentwurf werden vorher vom Entwickler mit phpMyAdmin erzeugt.
- Dem Benutzer wird mithilfe von PHP-Programmen eine komfortable Schnittstelle zum **Erzeugen, Anzeigen, Ändern und Löschen** von Datensätzen aus einer MySQL-Datenbank zur Verfügung gestellt.

Verbindung aufnehmen, Datensätze anzeigen

- Das folgende Programm illustriert die Anzeige aller Datensätze aus der Tabelle *personen* in der Datenbank *firma*:

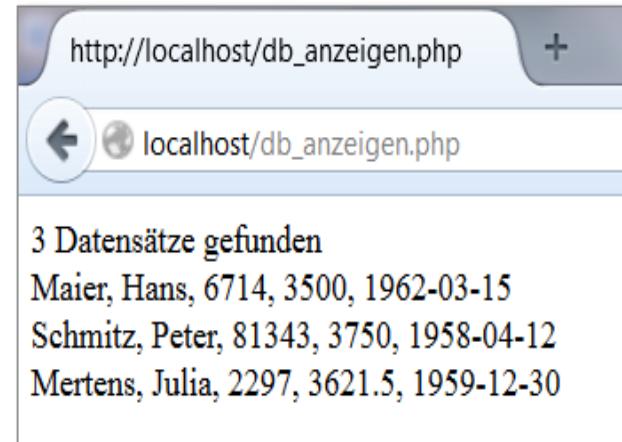
Datensätze anzeigen

Programm db_anzeigen.php

```
<?php
    // Verbindung zum MySQLDatenbankserver aufnehmen
$con = mysqli_connect("", "root"); // auch möglich: Host, Benutzer, Kennwort, DB
    // Datenbank auswählen
mysqli_select_db($con, "firma");
    // SQL-Abfrage ausführen
$res = mysqli_query($con, "select * from personen");
    // Anzahl Datensätze ermitteln und ausgeben
$num = mysqli_num_rows($res);

echo "$num Datensätze gefunden<br />";

/* Datensätze aus Ergebnis ermitteln,
   in Array speichern & ausgeben */
while ($dsatz = mysqli_fetch_assoc($res)) {
    echo $dsatz["name"] . ", "
    . $dsatz["vorname"] . ", "
    . $dsatz["personalnummer"] . ", "
    . $dsatz["gehalt"] . ", "
    . $dsatz["geburtstag"] . "<br />";
}
    // Verbindung schließen
mysqli_close($con);
?>
```



Erläuterung des Programms

- `mysqli_connect()` öffnet eine Verbindung zum MySQLDatenbankserver.
 - Bis zu drei Parameter möglich: Hostname, Benutzername und Kennwort.
 - Hostname ist localhost, der Standardbenutzer ist root (ohne Kennwort).
 - Rückgabewert der Funktion ist ein Verweis auf die Verbindung.
- `mysqli_select_db()`: DB auswählen
- `mysqli_query()` : Abfrage.
 - Gibt eine Ergebniskennung im Erfolgsfall zurück oder FALSE im Fehlerfall
- `mysqli_num_rows()` wird aufgerufen, wenn die Anzahl der Datensätze, die ermittelt wurden, ausgegeben werden soll.
- `mysqli_fetch_assoc()` wird verwendet, um einen Datensatz des Ergebnisses zu ermitteln und ihn in einem assoziativen Feld (hier \$dsatz) zu speichern.
- **Hinweis:**
 - Funktionen, deren Namen mit `mysql_` statt mit `mysqli_` beginnen, werden inzwischen als veraltet (*deprecated*) gekennzeichnet .
 - Das angehängte **i** bei `mysqli_` steht für *improved* (dt. verbessert).

Datensätze auswählen

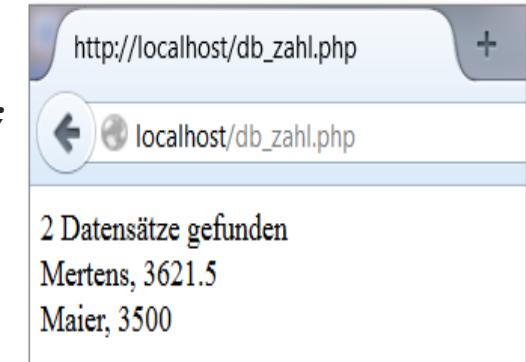
- In einem PHP-Programm können Sie Datensätze mithilfe von SQL-Anweisungen auswählen.
- Bei der Auswahl durch *where* innerhalb der *select*-Anweisung kann man Vergleichsoperatoren anwenden.
- Es lassen sich auch mehrere Auswahlbedingungen logisch miteinander verknüpfen.
- Der Operator *like* ist sehr nützlich beim Suchen nach Zeichenketten oder Teilen von Zeichenketten.
 - Dabei können auch Platzhalter (**Wildcards**) eingesetzt werden.
 - Ein % (Prozentzeichen) steht für eine beliebige Anzahl unbekannter Zeichen, und ein _ (Unterstrich) steht für genau ein unbekanntes Zeichen.
 - Die untersuchte Zeichenkette muss dabei weiterhin in einfache Hochkommata gesetzt werden.
 - Zusätzlich lässt sich die Reihenfolge der Ausgabe mithilfe von *order by* beeinflussen.
 - Der Zusatz *desc* steht für *descending* (= absteigend). Im Normalfall wird aufsteigend sortiert, d.h., die Angabe *asc* für *ascending* nicht nötig

Beispiel 1

Ein Beispiel mit ausgewählten Feldern, where-Klausel, Vergleichsoperator, logischem Operator und sortierter Ausgabe:

```
// Datei db_zahl.php

<html>
<body>
<?php
$con = mysqli_connect("", "root");
mysqli_select_db($con, "firma");
$sql = "select name, gehalt from personen";
$sql .= " where gehalt >= 3000 and gehalt <= 3700";
$sql .= " order by gehalt desc";
$res = mysqli_query($con, $sql);
$num = mysqli_num_rows($res);
echo "$num Datensätze gefunden<br />";
while ($dsatz = mysqli_fetch_assoc($res))
    echo $dsatz["name"] . ", " . $dsatz["gehalt"] . "<br />";
mysqli_close($con);
?>
</body>
</html>
```



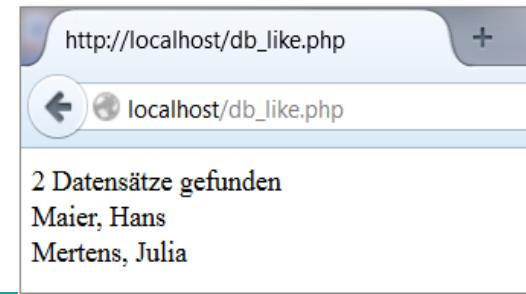
Beispiel 2

Ein Beispiel mit dem like-Operator:

```
//Datei db_like.php

<html>
<body>
<?php
$con = mysqli_connect("", "root");
mysqli_select_db($con, "firma");
$sql = "select name, vorname from personen";
$sql .= " where name like 'M%' order by name";
$res = mysqli_query($con, $sql);
$num = mysqli_num_rows($res);
echo "$num Datensätze gefunden<br />";
while ($dsatz = mysqli_fetch_assoc($res))
    echo $dsatz["name"] . ", " . $dsatz["vorname"] . "<br />";
mysqli_close($con);

?>
</body>
</html>
```



Aufgabe 11.2:

Schreiben Sie ein PHP-Programm zur Anzeige aller Datensätze aus der Tabelle *fp* der Datenbank *hardware*. Das Programm soll die folgende Ausgabe haben, basierend auf den ursprünglichen Daten.

5 Datensätze gefunden

Quantum, Fireball CX, 40, 112, 2008-10-01, HDA-208

Quantum, Fireball Plus, 80, 128, 2008-03-15, HDA-163

Fujitsu, MPE 3136, 160, 149, 2008-09-01, HDA-171

Seagate, 310232A, 60, 122, 2008-11-15, HDA-144

IBM Corporation, DJNA 372200, 240, 230, 2008-06-15, HDA-140

Aufgabe 11.3:

Zeigen Sie mit einem PHP-Programm aus der Tabelle fp der Datenbank hardware nur noch bestimmte Informationen an. Es sollen alle Festplatten mit den Angaben zu Hersteller, Typ, Artikelnummer und erstem Produktionsdatum angezeigt werden, die im ersten Halbjahr 2008 erstmalig produziert wurden, und zwar sortiert nach Datum. Das Programm soll die folgende Ausgabe erzeugen, basierend auf den ursprünglichen Daten.

2 Datensätze gefunden

Quantum, Fireball Plus, 2008-03-15, HDA-163

IBM Corporation, DJNA 372200, 2008-06-15, HDA-140

Ausgabe in eine HTML-Tabelle

```

<html>
<body>
<?php
$con = mysqli_connect("", "root");
mysqli_select_db($con, "firma");
$res = mysqli_query($con, "select *
from personen");

// Tabellenbeginn
echo "<table border='1'>";

// Überschrift
echo "<tr> <td>Lfd. Nr.</td>
<td>Name</td>";
echo "<td>Vorname</td>
<td>Personalnummer</td>";
echo "<td>Gehalt</td>
<td>Geburtstag</td> </tr>";
$l1f = 1;

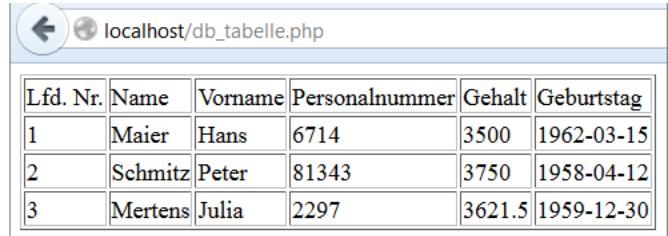
```

```

while ($dsatz = mysqli_fetch_assoc($res)) {
    echo "<tr>";
    echo "<td>$l1f</td>";
    echo "<td>" . $dsatz["name"] . "</td>";
    echo "<td>" . $dsatz["vorname"] .
                      "</td>";
    echo "<td>" . $dsatz["personalnummer"] .
                      "</td>";
    echo "<td>" . $dsatz["gehalt"] .
                      "</td>";
    echo "<td>" . $dsatz["geburtstag"] .
                      "</td>";
    echo "</tr>";
    $l1f = $l1f + 1;
}

// Tabellenende
echo "</table>";
mysqli_close($con);
?>
</body>
</html>

```



The screenshot shows a browser window with the URL "localhost/db_tabelle.php". The page displays a table with the following data:

Lfd. Nr.	Name	Vorname	Personalnummer	Gehalt	Geburtstag
1	Maier	Hans	6714	3500	1962-03-15
2	Schmitz	Peter	81343	3750	1958-04-12
3	Mertens	Julia	2297	3621.5	1959-12-30

Auswahl von Daten über ein Suchformular

- eine typische Internetdatenbankanwendung:
 - Der Benutzer gibt eine Anfrage (zum Beispiel eine Suchanfrage) ein, indem er Daten in ein Formular einträgt und diese Daten an den Webserver sendet.
 - Beim Webserver werden die Daten von einem PHP-Programm ausgewertet und mithilfe einer SQL-Anweisung an den Datenbankserver gesendet.
 - Der Datenbankserver ermittelt eine Antwort zur SQL-Anweisung und sendet diese an den Webserver zurück.
 - Das PHP-Programm verarbeitet die Antwort und sendet dem Benutzer eine Antwort.

Auswahl von Daten über ein Suchformular

Beispiel 1

- Im diesem Beispiel hat der Benutzer die Möglichkeit, zwei Zahlen einzugeben, die bei einer Abfrage als Untergrenze bzw. Obergrenze für das Feld *gehalt* dienen.
 - Der Anwender kann also bei jeder Abfrage festlegen, welcher Gehaltsgruppe die angezeigten Personen angehören sollen.

Zunächst das Formular:

```
//db_eingabe.htm
```

```
<html>
<body>
<p>Anzeige der Personen mit einem Gehalt zwischen:</p>
<form action = "db_eingabe.php" method = "post">
<p><input name="ug" /> Untergrenze</p><p>und</p>
<p><input name="og" /> Obergrenze</p>
<p><input type="submit" /> <input type="reset" /></p>
</form>
</body>
</html>
```

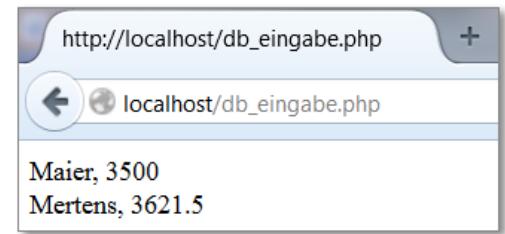
The screenshot shows a web browser window with the URL `localhost/db_eingabe.htm`. The page contains a form for selecting a salary range. It has two input fields: one labeled "Untergrenze" containing the value "3300" and another labeled "Obergrenze" containing the value "3700". Below the inputs are two buttons: "Daten absenden" (Send data) and "Zurücksetzen" (Reset). The page title is "Anzeige der Personen mit einem Gehalt zwischen:".

Auswahl von Daten über ein Suchformular

Das Programm sieht folgendermaßen aus:

```
// db_eingabe.php

<html>
<body>
<?php
$con = mysqli_connect("", "root");
mysqli_select_db($con, "firma");
$sql = "select name, gehalt from personen";
$sql .= " where gehalt >= " . $_POST["ug"]
. " and gehalt <= " . $_POST["og"];
$sql .= " order by gehalt";
$res = mysqli_query($con, $sql);
$num = mysqli_num_rows($res);
if ($num==0) echo "Keine passenden Datensätze gefunden";
while ($dsatz = mysqli_fetch_assoc($res))
    echo $dsatz["name"] . ", " . $dsatz["gehalt"] . "<br />";
mysqli_close($con);
?>
</body>
</html>
```



Auswahl von Daten über ein Suchformular

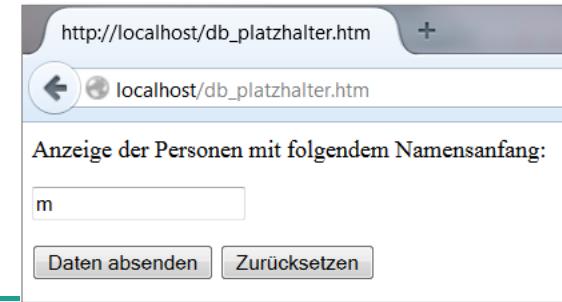
Beispiel 2

- Bei der Abfrage von Zeichenkettenfeldern muss man besonders auf die einfachen Hochkommata achten. Den Operator like und die Platzhalter % sowie _ kann man in gewohnter Weise verwenden.
- Mithilfe des Formulars aus dem folgenden Beispiel kann der Benutzer nach allen Personen suchen, deren Namen mit den eingegebenen Anfangsbuchstaben beginnen.

Hier zunächst das Formular:

//db_platzhalter.htm

```
<html>
<body>
<p>Anzeige der Personen mit folgendem Namensanfang:</p>
<form action = "db_platzhalter.php" method = "post">
<p><input name="anfang" /></p>
<p><input type="submit" /> <input type="reset" /></p>
</form>
</body>
</html>
```



Auswahl von Daten über ein Suchformular

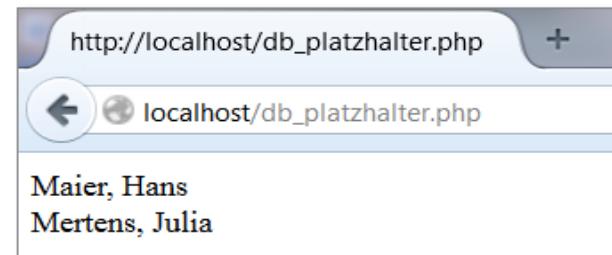
Das PHP-Programm sieht wie folgt aus:

```
// db_platzhalter.php

<html>
<body>
<?php
$con = mysqli_connect("", "root");
mysqli_select_db($con, "firma");
$sql = "select name, vorname from personen";
$sql .= " where name like '" . $_POST["anfang"] . "%'";
$res = mysqli_query($con, $sql);
$num = mysqli_num_rows($res);

if ($num==0) echo "Keine passenden Datensätze gefunden";

while ($dsatz = mysqli_fetch_assoc($res))
    echo $dsatz["name"] . ", " . $dsatz["vorname"] . "<br />";
mysqli_close($con);
?>
</body>
</html>
```



Auswahl von Daten über ein Suchformular

Beispiel 3

- Eine Abfrage durch die Verwendung von weiteren Formularelementen (Radiobutton-Gruppen, Kontrollkästchen, Auswahlmenüs usw.)
 - Mithilfe des folgenden Beispiels können Personen aus bestimmten Gehaltsgruppen angezeigt werden.

Zunächst das Formular:

```
<html>
<body>
<p>Anzeige der Personen aus der gewählten Gehaltsgruppe:</p>
<form action = "db_radio.php" method = "post">
<p><input type="radio" name="geh" value="1"/>
   bis 3000 &euro; einschl.<br />
<input type="radio" name="geh" value="2" />
   ab 3000 &euro; ausschl. bis 3500 &euro; einschl.<br />
<input type="radio" name="geh" value="3" checked="checked"/>
   ab 3 500 &euro; ausschl. bis 5000 &euro; einschl.<br />
<input type="radio" name="geh" value="4" />
   ab 5000 &euro; ausschl.</p>
<p><input type="submit" /> <input type="reset" /></p>
</form>
</body>
</html>
```



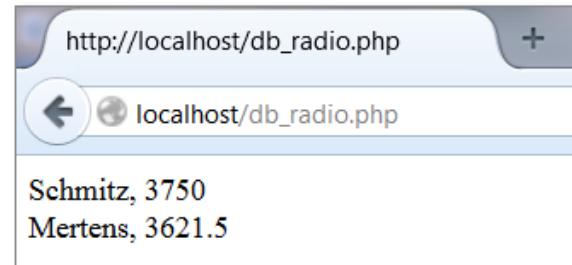
Auswahl von Daten über ein Suchformular

Das PHP-Programm lautet:

```
//db_radio.php

<html>
<body>
<?php
$con = mysqli_connect("", "root");
mysqli_select_db($con, "firma");
$sql = "select name, gehalt from
        personen where ";
switch($_POST["geh"]){
    case 1:
        $sql .= "gehalt <= 3000";
        break;
    case 2:
        $sql .= "gehalt > 3000 and
                  gehalt <= 3500";
        break;
    case 3:
        $sql .= "gehalt > 3500 and
                  gehalt <= 5000";
        break;
    case 4:
        $sql .= "gehalt > 5000";
}
```

```
$res = mysqli_query($con, $sql);
$num = mysqli_num_rows($res);
if ($num==0) echo "Keine passenden
                    Datensätze gefunden";
while ($dsatz =
        mysqli_fetch_assoc($res))
echo $dsatz["name"] . ", " .
        $dsatz["gehalt"] . "<br />";
mysqli_close($con);
?>
</body>
</html>
```



Aufgabe 11.4:

Zeigen Sie mit einem PHP-Programm aus der Tabelle fp der Datenbank hardware Festplatten aus bestimmten Preisgruppen an. Die Preisgruppen soll der Benutzer über Radiobuttons auswählen können (Dateien *u_db_radio.htm* und *u_db_radio.php*).

Es gelten die folgenden Preisgruppen:

bis 120 € einschließlich

ab 120 € ausschließlich und bis 140 € einschließlich

ab 140 € ausschließlich

Es sollen nur die Angaben zu Hersteller, Typ und Preis geliefert werden. Mithilfe eines Kontrollkästchens soll der Benutzer entscheiden können, ob er eine Sortierung der Ausgabe nach Preis wünscht.

Das Formular soll wie folgt aussehen.

Aufgabe 11.4 (Fortsetzung):

The screenshot shows a web browser window with the URL `http://localhost/u_db_radio.htm`. The page content is as follows:

Anzeige der Festplatten aus der gewählten Preisgruppe:

bis 120 € einschl.
 ab 120 € ausschl. bis 140 € einschl.
 ab 140 € ausschl.

Ausgabe nach Preis (absteigend) sortiert

Die Ausgabe zur Beispieleingabe zeigt die folgende Abbildung.

The screenshot shows a web browser window with the URL `http://localhost/u_db_radio.php`. The page content is as follows:

Quantum, Fireball Plus, 128
Seagate, 310232A, 122

Aufgabe 11.5:

Zeigen Sie mit einem PHP-Programm aus der oben angegebenen Tabelle nur noch Festplatten eines Herstellers an (Dateien *u_db_select.htm* und *u_db_select.php*). Der Benutzer soll den gewünschten Hersteller (Fujitsu, Quantum oder Seagate) in einem select-Menü auswählen. Die Daten sollen in Form einer HTML-Tabelle mit einer Überschrift angezeigt werden. Das Formular und die Ausgabe zur Beispielauswahl zeigen die folgenden Abbildungen.

Das Formular:

Anzeige der Festplatten des ausgewählten Herstellers:

Quantum ▾

Daten absenden Zurücksetzen

Ergebnis der Übung:

Hersteller	Typ	GB	Preis	Artnr.	Datum ...
Quantum	Fireball CX	40	112	HDA-208	2008-10-01
Quantum	Fireball Plus	80	128	HDA-163	2008-03-15

Datensätze erzeugen

- Bestimmten Benutzern kann es gestattet werden, weitere Datensätze zu erzeugen.
- Die Berechtigung hierzu lässt sich über den Benutzernamen und das Kennwort beim Aufbau der Datenbankverbindung oder über eine zusätzliche Passworteingabe festlegen.
- Es soll der Einfachheit halber angenommen werden, dass jeder Benutzer Datensätze hinzufügen, ändern und löschen kann.
- Es folgt ein Beispiel für eine Eingabeseite, die sich selbst aufruft .
 - **Das Formular und der PHP-Programmcode werden in einer Datei zusammengefasst:**

```
//db_erzeugen.php

<html>
<head>
<?php
if (isset($_POST["gesendet"])) {
    $con = mysqli_connect("", "root");
    mysqli_select_db($con, "firma");
    $sql = "insert personen"
        . "(name, vorname, personalnummer,"
        . " gehalt, geburtstag) values "
        . "(" . $_POST["na"] . ", "
        . "'". $_POST["vn"] . "', "
        . $_POST["pn"] . ", "
        . $_POST["ge"] . ", "
        . "'". $_POST["gt"] . "')";
    mysqli_query($con, $sql);
    $num = mysqli_affected_rows($con);
    if ($num>0) {
        echo "<p><font color='#00aa00'>";
        echo "Es wurde 1 Datensatz
            hinzugefügt";
        echo "</font></p>";
    }
}
```

```
else {
    echo "<p><font color='ff0000'>";
    echo "Es ist ein Fehler aufgetreten, ";
    echo "es wurde kein Datensatz
        hinzugefügt";
    echo "</font></p>";
}
mysqli_close($con);
?>
```

/* Die Funktion mysqli_affected_rows
ermittelt die Anzahl der von der Aktion
betroffenen Datensätze.

- Ist diese Anzahl größer als 0, dann
erfolgreich.

*/

//db_erzeugen.php Fortsetzung des Programmes

```
</head>
<body>
<p>Geben Sie bitte einen vollständigen Datensatz
  ein<br /> und senden Sie das Formular ab:</p>
<form action = "db_erzeugen.php" method = "post">
<p><input name="na" /> Name</p>
<p><input name="vn" /> Vorname</p>
<p><input name="pn" /> Personalnummer (eine ganze
  Zahl)</p>
<p><input name="ge" /> Gehalt (Nachkommastellen
  mit Punkt)</p>
<p><input name="gt" /> Geburtsdatum (in der Form
  JJJJ-MM-TT)</p>
<p><input type="submit" name="gesendet" />
<input type="reset" /></p>
</form>
<p>Alle Datensätze <a href="db_tabelle.php">
  anzeigen</a></p>
</body>
</html>
```

The screenshot shows a web browser window with the URL `http://localhost/db_erzeugen.php`. The page contains a form asking for a complete dataset. The fields are as follows:

- Name: Müller
- Vorname: Max
- Personalnummer (eine ganze Zahl): 3644
- Gehalt (Nachkommastellen mit Punkt): 3300
- Geburtsdatum (in der Form JJJJ-MM-TT): 1982-04-13

At the bottom are two buttons: "Daten absenden" and "Zurücksetzen". Below the form is a link "Alle Datensätze anzeigen".

The screenshot shows a web browser window with the URL `http://localhost/db_erzeugen.php`. A green success message "Es wurde 1 Datensatz hinzugefügt" is displayed. Below it is the same form as the previous screenshot, asking for a complete dataset.

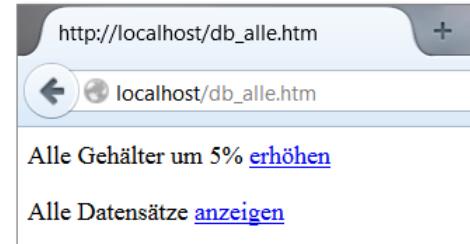
Ausgabe (Ausschnitt)

Ändern mehrerer Datensätze

- Nehmen wir an, dass aufgrund eines günstigen Geschäftsverlaufs die Gehälter aller Mitarbeiter um 5 % erhöht werden sollen. Die folgende HTML-Datei stellt zwei Möglichkeiten zur Verfügung:
 - Die Erhöhung soll durchgeführt werden (Aufruf des PHP-Programms über einen Hyperlink).
 - Alle Datensätze sollen zur Kontrolle angezeigt werden.

```
//db_alle.html
```

```
<html>
<body>
<p>Alle Gehälter um 5 % <a href="db_alle.php">erhöhen</a></p>
<p>Alle Datensätze <a href="db_tabelle.php">anzeigen</a></p>
</body>
</html>
```



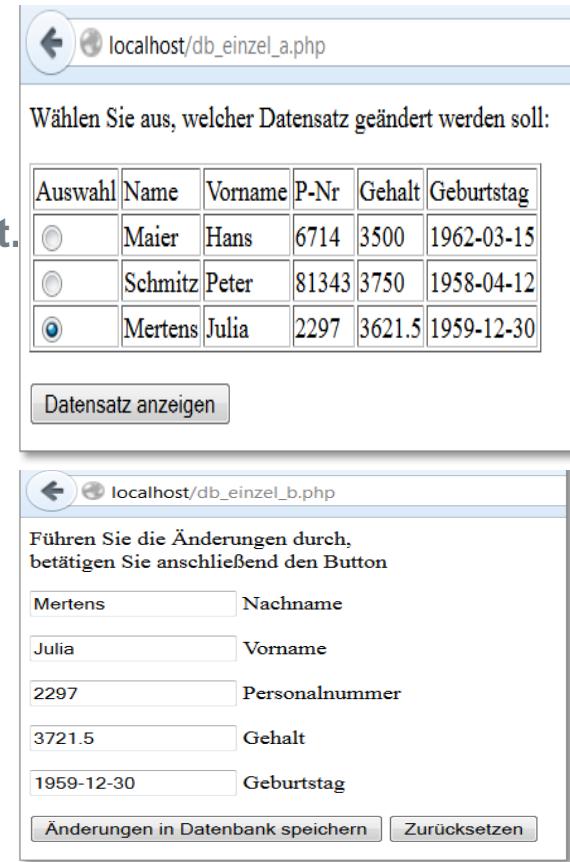
- Beim Aufruf des PHP-Programms wird die Änderung jedes Mal (!) durchgeführt:

```
// db_alle.php
```

```
<html>
<body>
<?php
$con = mysqli_connect("", "root");
mysqli_select_db($con, "firma");
$sql = "update personen set gehalt = gehalt * 1.05";
mysqli_query($con, $sql);
$num = mysqli_affected_rows($con);
echo "<p>Es wurden $num Datensätze geändert</p>";
mysqli_close($con);
?>
<p>Alle Datensätze <a href="db_tabelle.php">anzeigen</a></p>
</body>
</html>
```

Ändern eines bestimmten Datensatzes

- Um einen einzelnen Datensatz zu ändern, muss dieser zuvor identifiziert werden. Dies ist möglich, falls auf einem Feld der Tabelle ein Primärschlüssel liegt.
- Vorgehensweise
 - Dem Benutzer werden alle Datensätze angezeigt.
 - Er wählt den Datensatz aus, den er ändern möchte.
 - Der gewählte Datensatz wird in einem Formular angezeigt.
 - Der Benutzer gibt die Änderungen ein und führt sie aus.
- In der Tabelle personen liegt der eindeutige Index auf dem Feld personalnummer.
Die Vorgehensweise wird in den folgenden Dateien realisiert:
 - **db_einzel_a.php** zur Anzeige aller Datensätze und zur Auswahl
 - **db_einzel_b.php** zur Anzeige eines Datensatzes und zur Eingabe der Änderungen
 - **db_einzel_c.php** zur Durchführung der Änderungen



The image shows two screenshots of web forms for modifying a database record:

Screenshot 1: Selection Step (localhost/db_einzel_a.php)

Wählen Sie aus, welcher Datensatz geändert werden soll:

Auswahl	Name	Vorname	P-Nr	Gehalt	Geburtstag
<input type="radio"/>	Maier	Hans	6714	3500	1962-03-15
<input type="radio"/>	Schmitz	Peter	81343	3750	1958-04-12
<input checked="" type="radio"/>	Mertens	Julia	2297	3621.5	1959-12-30

Screenshot 2: Modification Step (localhost/db_einzel_b.php)

Führen Sie die Änderungen durch, betätigen Sie anschließend den Button

Mertens	Nachname
Julia	Vorname
2297	Personalnummer
3721.5	Gehalt
1959-12-30	Geburtstag

Buttons:

- Änderungen in Datenbank speichern
- Zurücksetzen

Ändern eines bestimmten Datensatzes

Anzeige aller Datensätze in Tabellenform
mit Radiobuttons zur Auswahl

```
//db_einzel_a.php

<html>
<body>

<p>Wählen Sie aus, welcher Datensatz  
geändert werden soll:</p>

<form action = "db_einzel_b.php" method =
"post">

<?php

$con = mysqli_connect("", "root");
mysqli_select_db($con, "firma");
$res = mysqli_query($con, "select * from
personen");

// Tabellenbeginn
echo "<table border='1'>";
// Überschrift
echo "<tr> <td>Auswahl</td> <td>Name</td>";
echo "<td>Vorname</td> <td>P-Nr</td>";
echo "<td>Gehalt</td> <td>Geburtstag</td>
</tr>";
```

```
while ($dsatz = mysqli_fetch_assoc($res))
{
echo "<tr>";
echo "<td><input type='radio' name='auswahl'";
echo " value=''" . $dsatz["personalnummer"] .
"' /></td>";
echo "<td>" . $dsatz["name"] . "</td>";
echo "<td>" . $dsatz["vorname"] . "</td>";
echo "<td>" . $dsatz["personalnummer"] .
"</td>";
echo "<td>" . $dsatz["gehalt"] . "</td>";
echo "<td>" . $dsatz["geburtstag"] . "</td>";
echo "</tr>";

// Tabellenende
echo "</table>";
mysqli_close($con);
?>
<p><input type="submit" value="Datensatz
anzeigen" /></p>
</form>
</body>
</html>
```

Ändern eines bestimmten Datensatzes

Anzeigen des ausgewählten Datensatzes mit allen Daten innerhalb eines Formulars

```
//db_einzel_b.php

<html>
<body>
<?php

if (isset($_POST["auswahl"])) {
$con = mysqli_connect("", "root");
mysqli_select_db($con, "firma");
$sql = "select * from personen where
personalnummer = " . $_POST["auswahl"];
$res = mysqli_query($con, $sql);
$dsatz = mysqli_fetch_assoc($res);
echo "<p>Führen Sie die Änderungen
durch,<br />";
echo "betätigen Sie anschließend den
Button</p>";
echo "<form action = 'db_einzel_c.php'
method = 'post'>";
echo "<p><input name='nn' value='"
. $dsatz["name"]
. "' /> Nachname</p>";
```

```
echo "<p><input name='vn' value='"
. $dsatz["vorname"]
. "' /> Vorname</p>";
echo "<p><input name='pn' value='"
. $_POST["auswahl"]
. "' /> Personalnummer</p>";
echo "<p><input name='ge' value='"
. $dsatz["gehalt"]
. "' /> Gehalt</p>";
echo "<p><input name='gt' value='"
. $dsatz["geburtstag"]
. "' /> Geburtstag</p>";
echo "<input type='hidden' name='oripn'
value='"
. $_POST["auswahl"]
. "' />";
echo "<p><input type='submit'
value='Änderungen in Datenbank speichern' />";

echo " <input type='reset' /></p>";
echo "</form>";
mysqli_close($con);

} else
echo "<p>Es wurde kein Datensatz
ausgewählt</p>";
?>
</body>
</html>
```

Ändern eines bestimmten Datensatzes

Nach der Durchführung wird die Änderung bestätigt

```
//db_einzel_c.php
```

```
<html>
<body>
<?php
$con = mysqli_connect("", "root");
mysqli_select_db($con, "firma");
$sql = "update personen set"
. " name = '" . $_POST["nn"] . "' , "
. " vorname = '" . $_POST["vn"] . "' , "
. " personalnummer = " . $_POST["pn"] . " , "
. " gehalt = " . $_POST["ge"] . " , "
. " geburtstag = '" . $_POST["gt"] . "' "
. " where personalnummer = "
. $_POST["oripn"];
mysqli_query($con, $sql);
$num = mysqli_affected_rows($con);
```

```
if ($num>0)
echo "<p>Der Datensatz wurde geändert</p>";
else
echo "<p>Der Datensatz wurde nicht
geändert</p>";

mysqli_close($con);
?>

<p>Zurück zur <a
href="db_einzel_a.php">Auswahl</a></p>
</body>
</html>
```

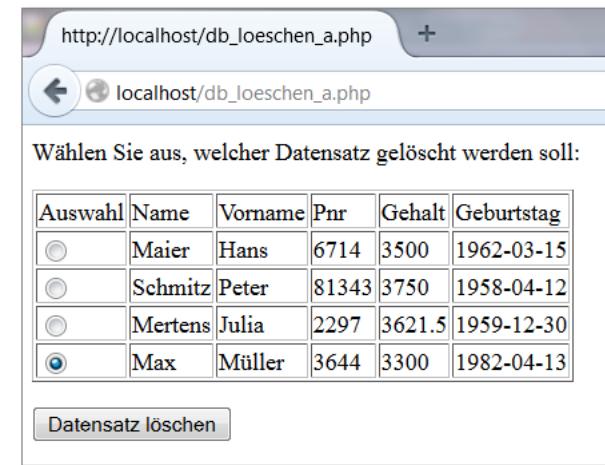


Datensätze löschen

- Zum Löschen eines einzelnen Datensatzes muss (wie beim Ändern) der betreffende Datensatz zuvor über das Feld ermittelt werden, auf dem der Primärschlüssel liegt.
- Es empfiehlt sich die folgende Vorgehensweise:
 - Dem Benutzer werden alle Datensätze angezeigt.
 - Er wählt den Datensatz aus, den er löschen möchte.
 - Er führt die Löschung durch.

Die Vorgehensweise wird in zwei Dateien realisiert:

- *db_loeschen_a.php* zur Anzeige aller Datensätze und zur Auswahl,
- *db_loeschen_b.php* zur Durchführung der Änderungen



Auswahl	Name	Vorname	Pnr	Gehalt	Geburtstag
<input type="radio"/>	Maier	Hans	6714	3500	1962-03-15
<input type="radio"/>	Schmitz	Peter	81343	3750	1958-04-12
<input type="radio"/>	Mertens	Julia	2297	3621.5	1959-12-30
<input checked="" type="radio"/>	Max	Müller	3644	3300	1982-04-13

Datensatz löschen

Die Datei *db_loeschen_a.php* wird hier nicht gesondert aufgeführt, da sie sich von der Datei *db_einzel_a.php* nur darin unterscheidet:

- Im Text oberhalb der Tabelle wird das Wort geändert durch das Wort gelöscht ersetzt.
- Die Aufschrift des Submit-Buttons wird von Datensatz anzeigen auf Datensatz löschen geändert.
- Bei dem aufgerufenen PHP-Programm handelt es sich um die Datei *db_loeschen_b.php* anstelle der Datei *db_einzel_b.php*.

```
//db_loeschen_b.php

<html>
<body>
<?php
if (isset($_POST["auswahl"])) {
    $con = mysqli_connect("", "root");
    mysqli_select_db($con, "firma");
    $sql = "delete from personen where"
        . " personalnummer = " . $_POST["auswahl"];
    mysqli_query($con, $sql);
    $num = mysqli_affected_rows($con);
    if ($num>0)
        echo "<p>Der Datensatz wurde gelöscht</p>";
    else
        echo "<p>Der Datensatz wurde nicht gelöscht</p>";
    mysqli_close($con);
}
else
echo "<p>Es wurde kein Datensatz ausgewählt</p>";
?>
<p>Zurück zur <a href="db_loeschen_a.php">Auswahl</a></p>
</body>
</html>
```



Aufgabe 11.6:

Ermöglichen Sie mit einem PHP-Programm das Hinzufügen von Datensätzen zu der Tabelle fp der Datenbank hardware. Das Formular soll wie folgt aussehen.

Geben Sie bitte einen vollständigen Datensatz ein und senden Sie das Formular ab:

Hersteller

Typ

GB

Preis (Nachkommastellen mit Punkt)

Artikelnummer

Datum der ersten Produktion (in der Form JJJJ-MM-TT)

[Alle Datensätze anzeigen](#)

Aufgabe 11.7:

Ermöglichen Sie mithilfe eines PHP-Programms das Ändern von Datensätzen in der Tabelle fp der Datenbank hardware (Dateien *u_db_einzel_a.php*, *u_db_einzel_b.php* und *u_db_einzel_c.php*).

1 Einleitung: Modulziele, Prüfungsform & Organisatorisches

2 Einführung in Skriptsprachen

3 Einführung in PHP

4 Datentypen

5 Operatoren

6 Kontrollstrukturen

7 Funktionen

8 Objektorientierung

9 Ein- und Ausgabe (Dateien)

10 Web-Formularanbindung

11 Datenbankanbindung

12 Typische Anwendungen

Typische Anwendungen

- Mit PHP können
 - **Blogs,**
 - **Content- Management-Systeme,**
 - **Shop-Systeme, Foren,**
 - **Bildergalerien usw.**programmiert werden.
- Viele bekannte Content-Management-Systeme wie WordPress, Joomla!, Drupal und TYPO3 basieren auf PHP.
- PHP ist auch für kleine Anwendungen geeignet, wie z.B.
 - **Cookies und Sessionverwaltung**
 - **Emailversand**
 - **Netzwerkfunktionen,**
 - **File-Funktionen**
 - **PDF-Erzeugung**
 - **Etc.**

Ausgewählte Anwendungen

1. Cookies und Sessions (Sitzungen)
2. Diskussionsforum als PHP-Anwendung
3. Projektaufgabe: Entwicklung einer kleinen Webanwendung

Wozu Cookies und Sessions?

- Für manche Anwendungen braucht man eine Möglichkeit, Zustände im Dialog zu speichern und einen Anwender wiederzuerkennen, z.B.
 - Für den Warenkorb eines Shop-Systems, in dem die ausgewählten Produkte bleiben, auch wenn der Benutzer die Seite wechselt.
 - Für personalisierte Seiten, die einen Benutzer persönlich begrüßen und die auf ihn zugeschnittene Inhalte zeigen
- Eine Lösung bieten Cookies und Sessions.

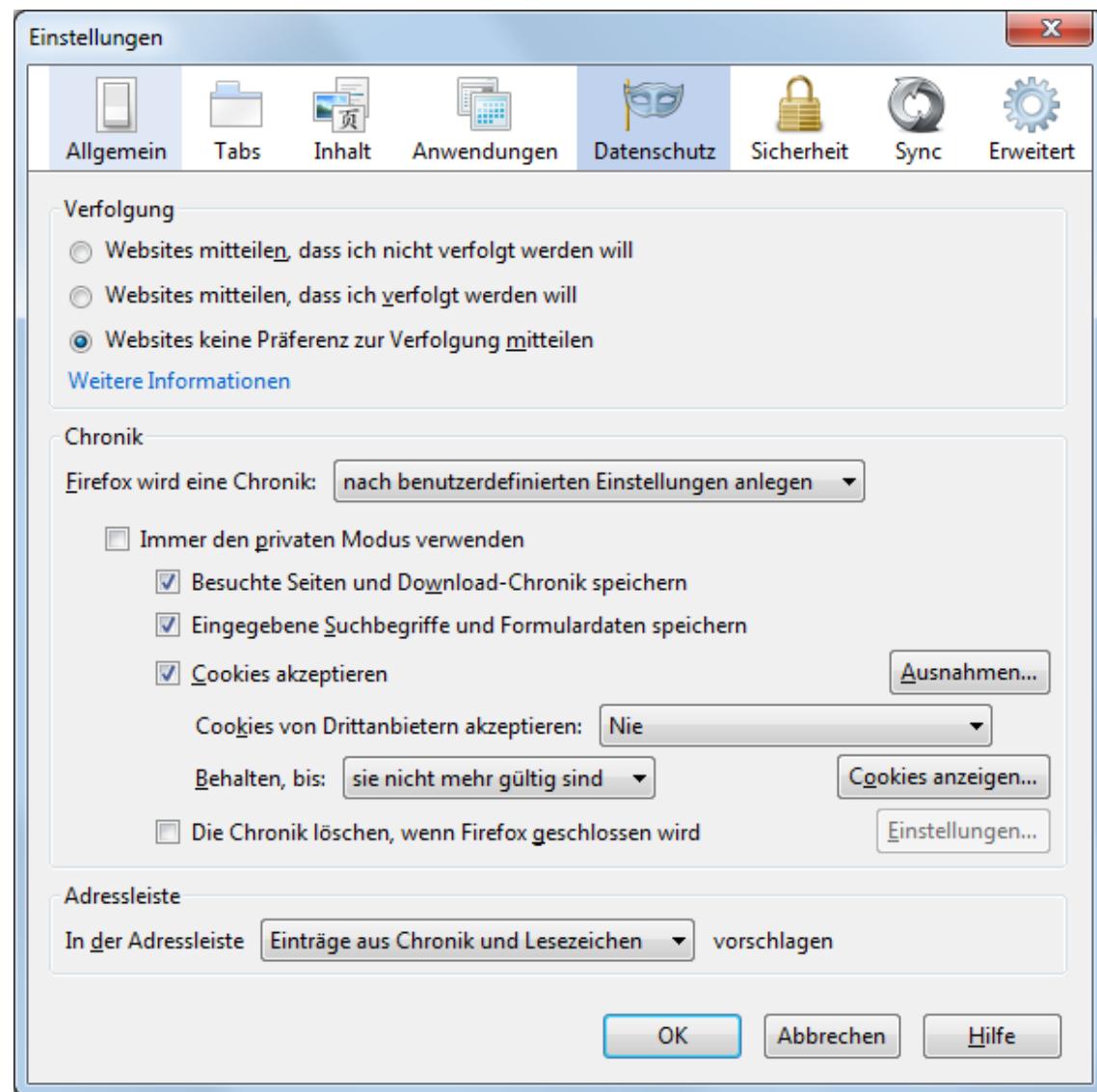
Cookies

- sind kleine Textinformationen, die vom Server an den Browser gesendet werden und die beim Surfer auf der Festplatte gespeichert werden.
- Bei späteren Zugriffen auf denselben Webserver sendet der Browser jeweils das Cookie wieder mit.
- Auf diese Art kann der Webserver einen Anwender »wiedererkennen«.

Allgemeine Eigenschaften

- Cookies sind in den einzelnen Browsern unterschiedlich implementiert.
- Browser speichern mindestens 300 Cookies, die jeweils 4.096 Bytes groß sein dürfen, wobei 20 Cookies pro Domain erlaubt sind.
- Cookies können immer nur von der Domain gelesen werden, die das Cookie auch gesetzt hat.
- Ob und welche Cookies gespeichert werden dürfen, bestimmt der Benutzer in seinen Browsereinstellungen (z.B. im Firefox unter *Einstellungen/Datenschutz*)

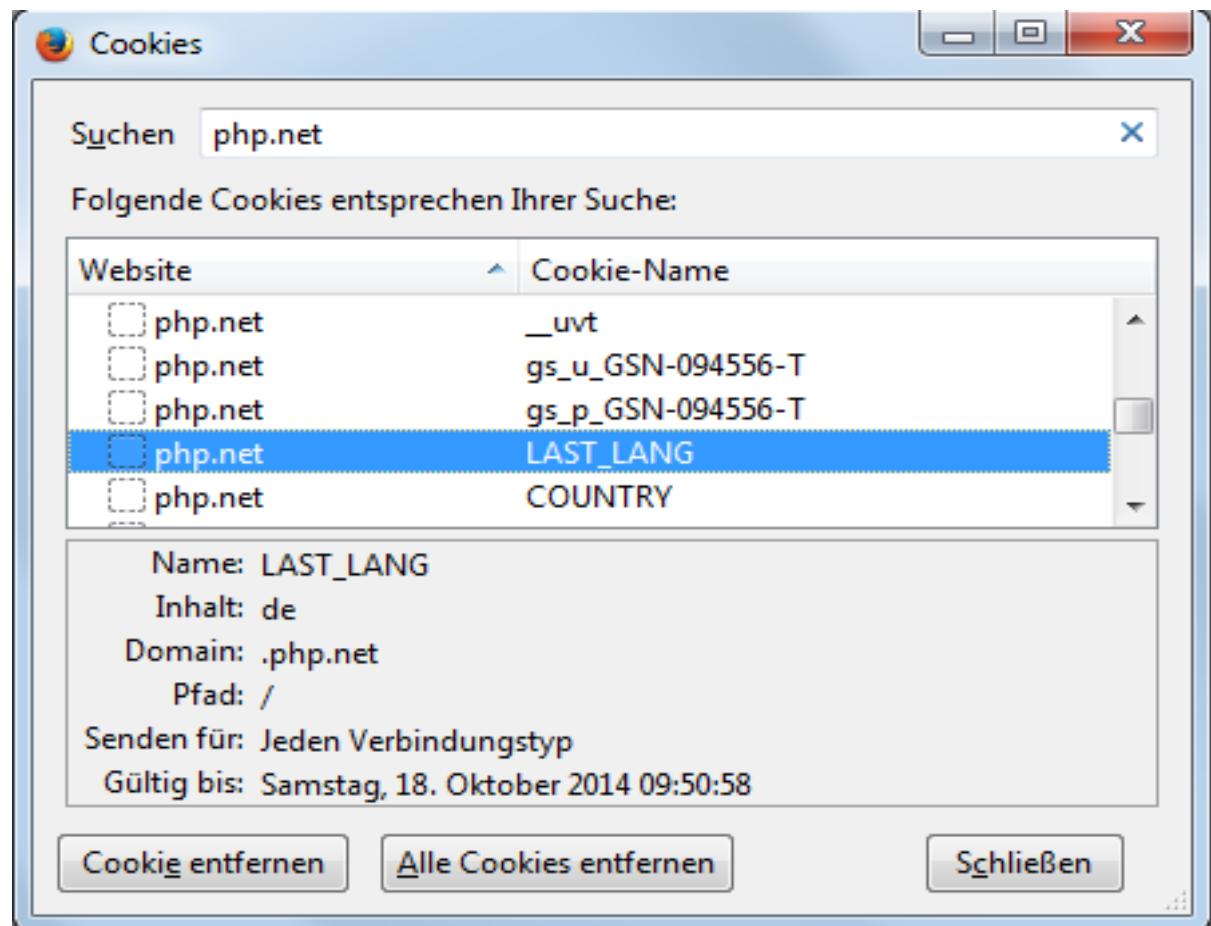
Einstellungen für Cookies im Firefox



Beispiel:

Die von der Website php.net gespeicherten Cookies

- Das Cookie LAST_LANG speichert die zuletzt verwendete Sprache (de).
 - Die Erläuterungen des PHP-Manuals können bei der nächsten Benutzung gleich in der richtigen Sprache angezeigt werden.



Cookies setzen per PHP

Funktion `setcookie()`.

- erlaubt mehrere Parameter

```
setcookie("sprache", "en", time() + 3600, "/", ".example.com", true, true);
```

- Der erste Parameter – der Name – ist obligatorisch, die anderen sind fakultativ

Bedeutung der Parameter:

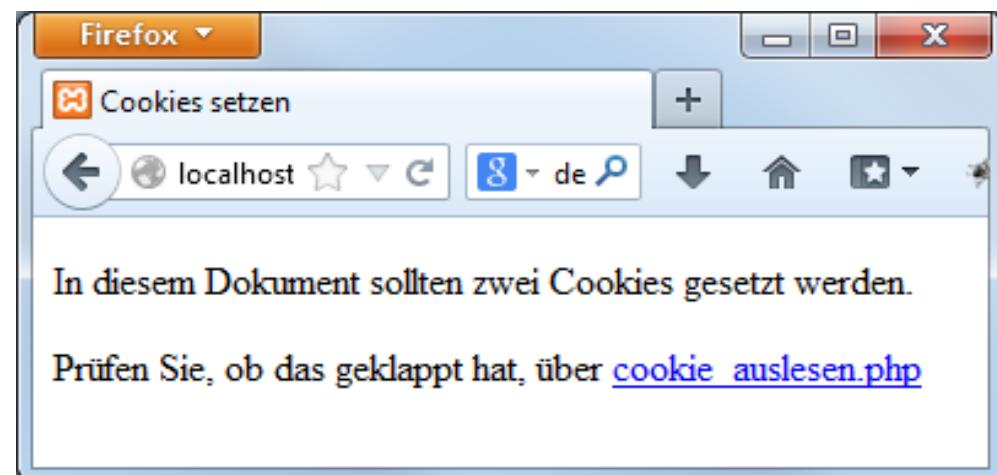
- der *Name* des Cookies (hier *sprache*)
- der *Wert*, hier *en*
- das *Verfallsdatum* in Form eines *Zeitstempels*.
 - Dazu wird noch die gewünschte Anzahl an Sekunden addiert. Wird dieser Parameter nicht gesetzt, gilt das Cookie nur während der aktuellen Browsersitzung

Bedeutung der Parameter (Fortsetzung):

- **der *Pfad*.**
 - Standardmäßig gilt das Cookie nur für den Unterordner, in dem das Skript steht, das das Cookie gesetzt hat.
 - / bedeutet, dass das Cookie für die gesamte Domain gültig ist. Geben Sie hier /manual/ an, ist das Cookie nur für den Ordner *manual* und alle Unterordner gültig.
- **die *Domain*, für die das Cookie gilt.**
 - Das ist nur relevant, wenn Sie mit Subdomains arbeiten. Zwei mögliche Subdomains zu example.com könnten z.B. de.example.com und en.example.com sein. Normalerweise gilt das Cookie nur für die Subdomain, die das Cookie setzt. Soll ein Cookie in allen Subdomains von example.com gelten, schreiben Sie .example.com.
- **Verbindungstyp:**
 - Wenn Sie für diesen Parameter true angeben, darf das Cookie nur über sichere HTTPS-Verbindungen versendet werden.
- **httponly:**
 - Über den letzten Parameter können Sie durch die Angabe true festlegen, dass das Cookie nur über das HTTP-Protokoll ausgelesen wird und nicht beispielsweise per JavaScript. Das kann helfen, einen Identitätsklau zu verhindern.

Beispiel: Setzen von zwei Cookies (*cookie_setzen.php*)

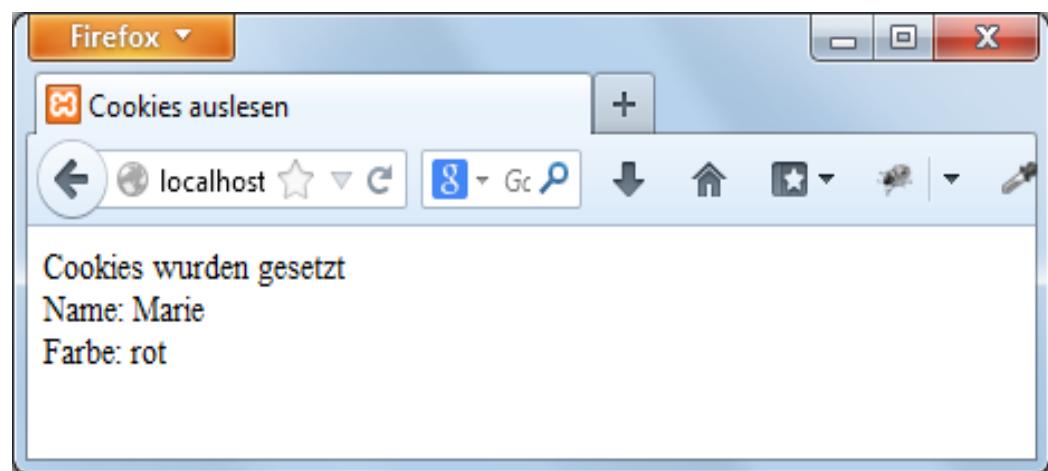
```
<?php  
  
    setcookie("name", "Marie", time() +7200);  
    setcookie("farbe", "rot", time() +7200);  
  
?>  
  
<!DOCTYPE html>  
  
<html>  
<head>  
  
<meta charset="UTF-8" />  
<title>Cookies setzen</title>  
</head>  
  
<body>  
  
<p>In diesem Dokument sollten zwei Cookies gesetzt werden.</p>  
  
<p>Prüfen Sie, ob das geklappt hat, über cookie\_auslesen.php</p>  
  
</body>  
  
</html>
```



Vorsicht: Nach dem Setzen eines Cookies stehen die Cookieeinträge erst nach dem Laden der nächsten Seite im assoziativen Array `$_COOKIES` zur Verfügung

Skript zum Auslesen der Cookie-Werte:

```
// cookie_auslesen.php
if (isset($_COOKIE["name"]) && isset($_COOKIE["farbe"])) {
    echo "Cookies wurden gesetzt<br />\n";
    echo "Name: " . htmlspecialchars($_COOKIE["name"]) . "<br />\n";
    echo "Farbe: " . htmlspecialchars($_COOKIE["farbe"]);
} else {
    echo "keine Cookies gesetzt";
}
```



Zum Löschen von Cookies gibt es keine eigene Funktion.

- Sie löschen ein Cookie, indem Sie ein Cookie mit denselben Parametern setzen, aber mit einer Ablaufzeit, die in der Vergangenheit liegt:

```
setcookie("name", "Marie", mktime(0, 0, 0, 1, 1, 1980));
```

- Je nach Einstellung des Browsers des Benutzers werden Cookies automatisch gelöscht, unabhängig von den gewählten Werten.

Aufgabe 12.1:

- Erstellen Sie ein Formular mit einem Textfeld, in dem der Benutzer seinen Namen eingeben kann. Zusätzlich gibt es außerdem einen Absendebutton.
- Speichern Sie diese Information in einem Cookie.
- Erstellen Sie dann ein weiteres Dokument – das heißt eine weitere Datei –, in der diese Information ausgelesen wird.
- Begrüßen Sie dort den Benutzer bei gesetztem Cookie mit seinem Namen oder ansonsten mit »Hallo Unbekannter«.

Aufgabe 12.2: Adressspeicherung

In dieser Übung soll eine Adresse gespeichert werden, z.B. die Lieferadresse oder die Rechnungsadresse eines Benutzers bei einem Webshop. Bei der nächsten Bestellung kann dem Benutzer somit Arbeit erspart werden.

Beim ersten Besuch erscheint die Adressseite (in Abbildung 1 nur Nachname und Vorname) mit einem leeren Formular, da der Benutzer dem Webshop noch unbekannt ist.

Der Benutzer gibt seine Adressdaten ein, betätigt den Button Bestellen und erhält eine Bestätigung, siehe Abbildung 2.

Gleichzeitig werden seine Adressdaten in Cookies gespeichert.

Beim nächsten Besuch des Webshops erscheint die Adressseite mit einem bereits gefüllten Formular, siehe Abbildung 3.

Die Daten werden aus den gespeicherten Cookies ermittelt. Der Benutzer kann diese Daten direkt verwenden oder geänderte Daten eintragen.

Aufgabe 12.2: (Fortsetzung)

The screenshot shows a web browser window with the URL `http://localhost/sc_adresse_a.php`. The page contains a form with two input fields labeled "Nachname" and "Vorname". Below the fields is a button labeled "Bestellen".

http://localhost/sc_adresse_a.php

localhost/sc_adresse_a.php

Ihr Name

Nachname

Vorname

Bestellen

Abb. 1 Neuer Kunde,
Daten noch nicht gespeichert

The screenshot shows a web browser window with the URL `http://localhost/sc_adresse_b.php`. The page displays a confirmation message: "Ihre Ware wird versandt, Hans Maier".

http://localhost/sc_adresse_b.php

localhost/sc_adresse_b.php

Bestätigung

Ihre Ware wird versandt, Hans Maier

Abb. 2 Bestätigung der Adresse

The screenshot shows a web browser window with the URL `http://localhost/sc_adresse_a.php`. The page displays a form with pre-filled fields: "Nachname" set to "Maier" and "Vorname" set to "Hans". Below the fields is a button labeled "Bestellen".

http://localhost/sc_adresse_a.php

localhost/sc_adresse_a.php

Ihr Name

Maier Nachname

Hans Vorname

Bestellen

Abb. 3 Bekannter Kunde,
Daten bereits gespeichert

Sessions - Sitzungen

Auch Sessions werden dazu verwendet, Benutzer wiederzuerkennen und Daten über mehrere Webseiten hinweg verfügbar zu halten.

Unterschied zu Cookies:

- die Daten werden nicht auf dem Client (Browser), sondern *auf dem Server* gespeichert.
- Auf dem Client wird hingegen nur ein Cookie gespeichert, das den Client beim Server identifiziert.
- Daten werden während einer Browsersitzung (d.h. bis der Browser geschlossen wird) gespeichert.

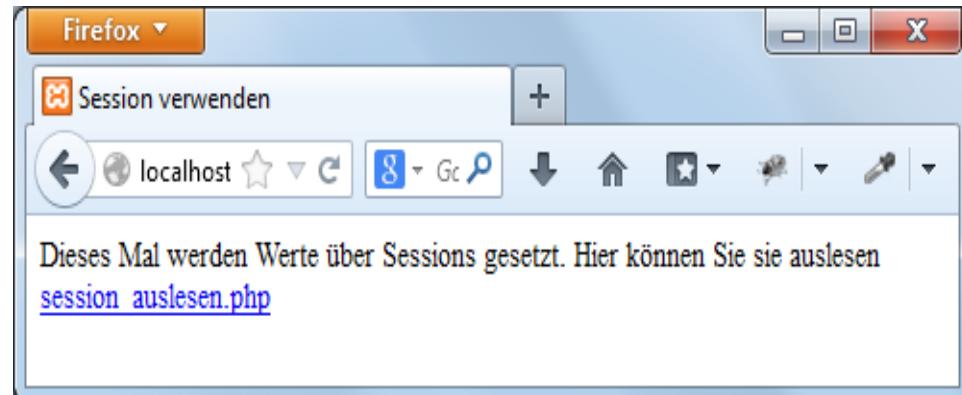
Eine Session starten und Daten speichern

- `session_start()`. *// soll als erstes vor HTML gestartet werden*
- assoziative Array `$_SESSION`, um Schlüssel-Wert-Paare zu speichern.

Eine Session starten und Daten speichern

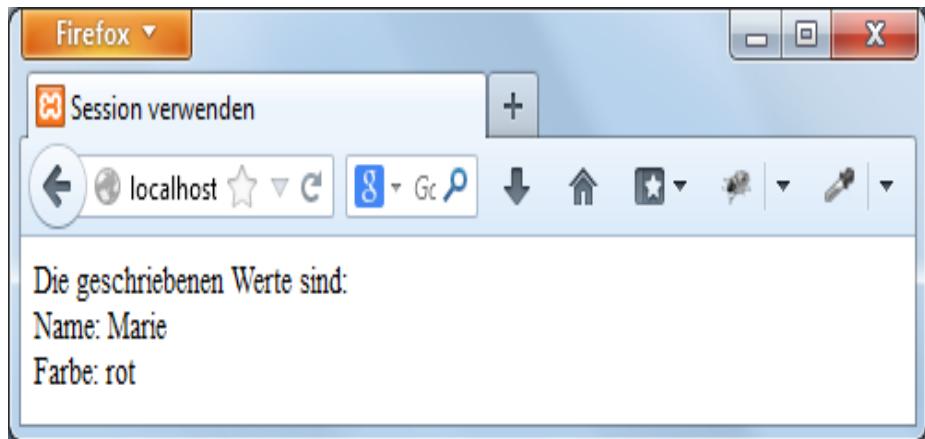
```
// session.php
```

```
<?php
    session_start();
?>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<title>Session verwenden</title>
</head>
<body>
<?php
    $_SESSION["name"] = "Marie";
    $_SESSION["farbe"] = "rot";
    echo "Dieses Mal werden Werte über Sessions gesetzt. ";
    echo "Hier können Sie sie auslesen
        <a href='session_auslesen.php'>session_auslesen.php</a>";
?>
</body>
</html>
```



```
// session_auslesen.php

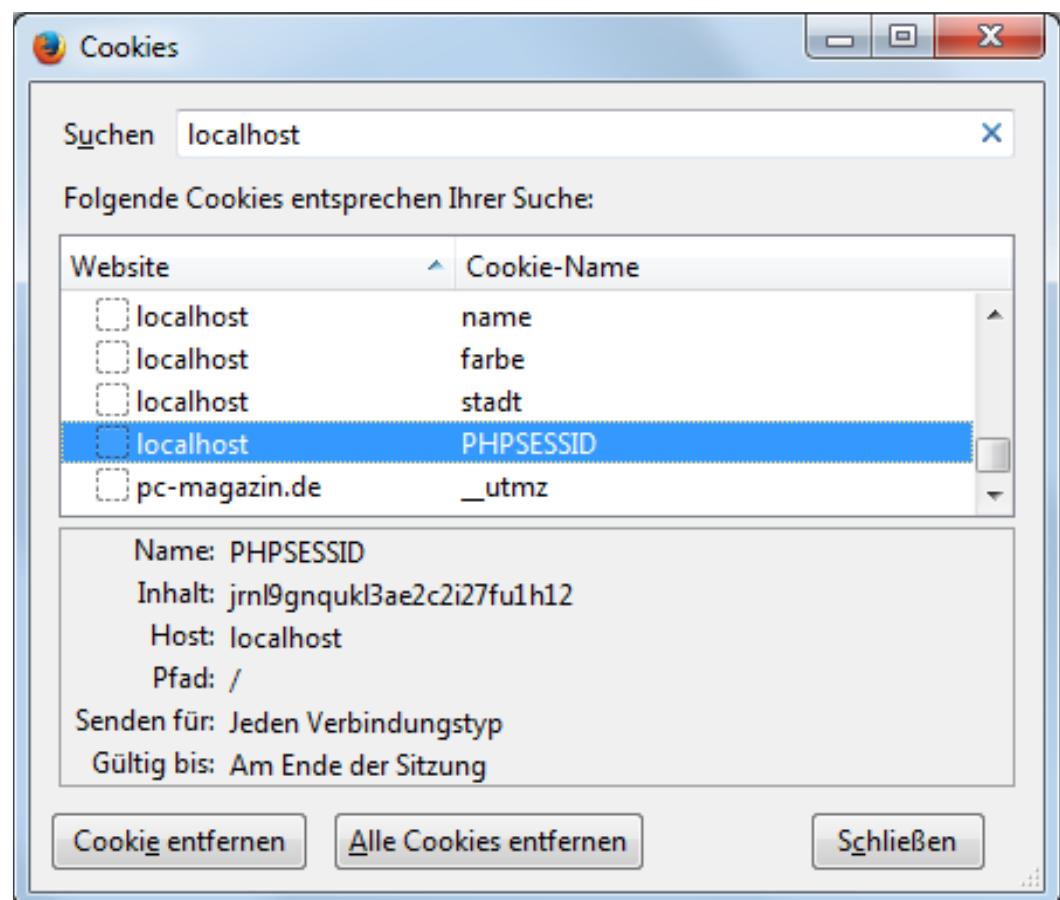
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<title>Session verwenden</title>
</head>
<body>
<?php
    if (isset($_SESSION["name"]) && isset($_SESSION["farbe"])) {
        echo "Die geschriebenen Werte sind: <br />";
        echo "Name: {" . $_SESSION['name'] . "}<br />\n";
        echo "Farbe: {" . $_SESSION['farbe'] . "}<br />\n";
    } else {
        echo "Noch keine Session gesetzt";
    }
?>
</body>
</html>
```



Wo werden die Session-Daten gespeichert?

Im Browser findet sich ein Cookie namens **PHPSESSID** mit einem ziemlich kryptischen Wert, den PHP selbst vergibt.

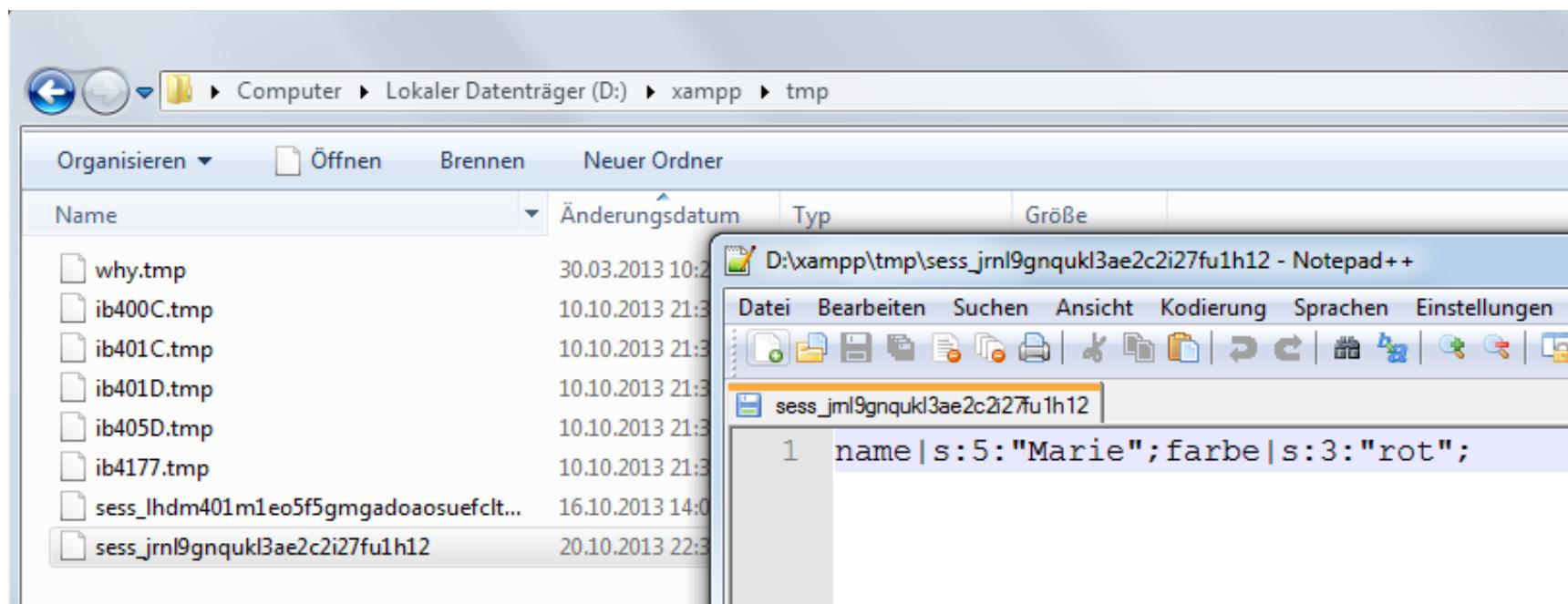
- Man kann sich das Cookie z.B. in Firefox anzeigen lassen (den entsprechenden Dialog wählen und nach den Cookies für *localhost* suchen).



Wo werden die Session-Daten gespeichert?

Die eigentlichen Informationen verbleiben auf dem Server.

- Bei der XAMPP-Installation unter Windows werden die Session-Daten im **Unterverzeichnis *tmp*** innerhalb von *xampp* gespeichert.
- Hier findet man eine Datei, die mit ***sess_*** beginnt, und darauf folgt der als Inhalt des Cookies vergebene Wert. Darin stehen die Daten in serialisierter Form.



Sessions bei deaktivierten Cookies

Da die Identifizierung des Clients über ein Cookie geschieht, funktionieren Sessions bei vollständiger Deaktivierung von Cookies nicht.

- Der Benutzer muss zumindest temporäre Cookies zulassen, d.h. diejenigen, die bis zum Beenden des Browsers erhalten bleiben.

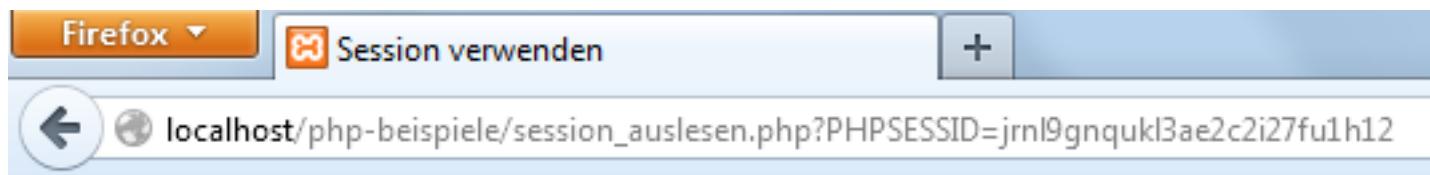
Sollen Sessions auch bei deaktivierten Cookies funktionieren, muss man die Session-ID über die URL übergeben.

- In unserem Beispiel muss man dafür die Datei `session.php` bearbeiten, und zwar an der Stelle, an der der Link steht.
- Diesen muss man um die Session-Information erweitern. Dafür braucht man
 - den von PHP vergebenen Sessionnamen, den `session_name()` liefert, und
 - den Wert, den man über `session_id()` ermitteln kann.

An den Link wird die Session-Information angehängt

```
echo "Hier können Sie sie auslesen <a href='session_auslesen.php?'  
    . session_name() . "=" . session_id()  
    . "'>session_auslesen.php</a>";
```

Hier wird die Session-ID über die URL übertragen.



Allerdings ist die Übertragung der Session-ID über Links aus sicherheitstechnischen Gründen nicht empfehlenswert

Beispiel: Ein Login-System mit Sessions

- Eine klassische Anwendung für Sessions, die nur berechtigten Benutzern Zugriff auf bestimmte Informationen erlaubt.

Das Beispielprogramm besteht aus folgenden Dateien:

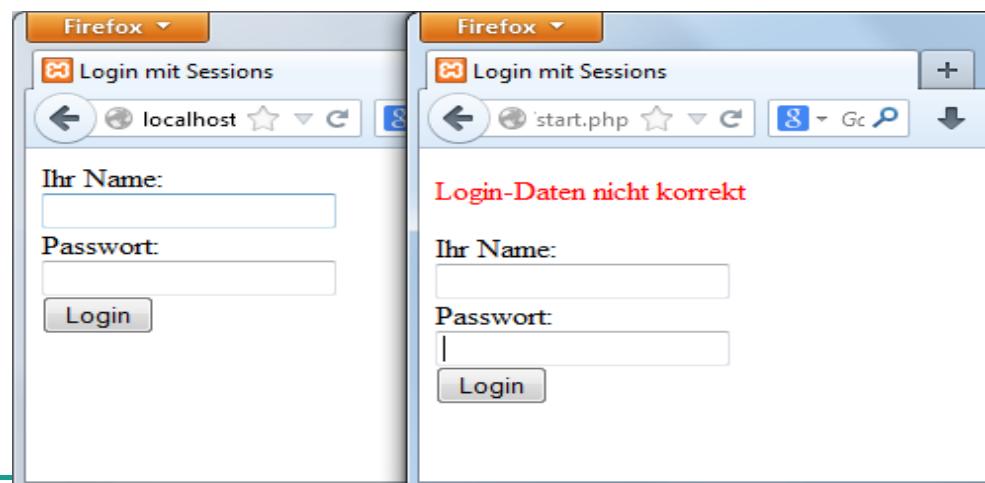
- *start.php* beinhaltet die Startseite mit dem Formular, in das der Benutzer seinen Benutzernamen und sein Passwort eintragen kann.
- In *login.php* werden die Login-Informationen verarbeitet. Wenn die eingegebenen Daten korrekt sind, findet eine Weiterleitung zur Datei *willkommen.php* statt. Sind die Daten hingegen nicht korrekt, wird wieder zur Startseite umgeleitet und es erscheint eine Fehlermeldung.
- Die Seite *willkommen.php* erreicht nur ein dazu berechtigter Benutzer. Er findet hier auch einen Link auf *logout.php*, um sich auszuloggen. Wird die Seite *willkommen.php* ohne Berechtigung aufgerufen, wird der Benutzer wieder auf *start.php* umgeleitet.
- In *logout.php* werden alle Session-Informationen gelöscht, und der Benutzer wird wieder zur Startseite umgeleitet.

Die Startseite mit dem Login-Formular

```
// start.php

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<title>Login mit Sessions</title>
<style>
.fehler { color: red; }
</style>
</head>
<body>
<?php // f wird in login.php gesetzt
if (isset($_GET["f"]) && $_GET["f"] == 1) {
    echo "<p class='fehler'>Login-Daten
            nicht korrekt</p>";
}
?>
```

```
<form action="login.php" method="post" >
    Ihr Name: <br />
    <input type="text" name="name" size="20" />
    <br />
    Passwort: <br />
    <input type="password" name="passwort"
           size="20" /><br />
    <input type="submit" value="Login" />
</form>
</body>
</html>
```



Die Überprüfung der Benutzerdaten (vereinfachte Version)

// login.php

```
<?php
` session_start();           // Session starten
if (isset($_POST["name"]) && $_POST["name"] == "Hans"
    && $_POST["passwort"] == "geheim")           // In einer realen Anwendung
{
    $_SESSION["name"] = "Hans";                  // würden diese Daten aus einer
    $_SESSION["login"] = "ok";                   // Datenbank stammen
    header("Location: willkommen.php");        // Funktion Header: Hier für die Umleitung
    exit;
} else {
    header("Location: start.php?f=1");         // Fehler wird hier gesetzt & in Start.php geprüft
    exit;
}
?>           // Vorsicht: Stellen Sie nach der Umleitung mit Header sicher, dass der
              // nachfolgende Code nicht ausgeführt wird → exit verwenden
              // Siehe: http://www.php.net/manual/de/function.header.php
```

Ein Login-System mit Sessions

Die Willkommenseite nur für berechtigte Benutzer

// willkommen.php

```
<?php
    session_start();
    if (isset($_SESSION["login"]) &&
        $_SESSION["login"] == "ok") {
?>
<!DOCTYPE html>

<html>
<head>
    <meta charset="UTF-8" />
    <title>Willkommen im geschützten
          Bereich</title>
</head>
<body>
<?php
    echo "<h1>Hallo {$_SESSION['name']}</h1>";
?>
    <p>Hier stehen viele weitere interessante
          Informationen</p>
    <p><a href='logout.php'>Ausloggen</a></p>
</body>
</html>
```

```
<?php
} else {
    $host = htmlspecialchars
        ($_SERVER["HTTP_HOST"]);
    $uri = rtrim(dirname(htmlspecialchars
        ($_SERVER["PHP_SELF"])), "/\\");
    $extra = "start.php";
    header("Location: http://$host$uri/
        $extra");
}
?>
```



Das Skript zum Ausloggen

// logout.php

```
<?php
    session_start();
    $_SESSION = array();      //auf ein leeres Array gesetzt→gelöscht
    if (isset($_COOKIE[session_name()])) {           // Cookie gelöscht
        setcookie(session_name(), "", time()-42000, "/");
    }
    session_destroy();

$host = htmlspecialchars($_SERVER["HTTP_HOST"]);
$uri = rtrim(dirname(htmlspecialchars($_SERVER["PHP_SELF"])), "/\\");
$extra = "start.php";
header("Location: http://$host$uri/$extra");
?>
```

Nützliche Links

Ein weiteres Beispiel für ein Loginsystem und Benutzerregistrierung mit PHP und MySQL finden Sie unter:

- http://wiki.selfhtml.org/wiki/Benutzer:Suit/Loginsystem_und_Benutzerregistrierung_mit_PHP_und_MySQL

Aufgabe 12.3: Session für ein Webshop

In dieser Übung handelt es sich um eine Session für ein Webshop. Die Produktdaten für diese Übung stehen in einer include-Datei (sc_shop.inc.php) im Online-Campus zur Verfügung.

Der Benutzer wählt die jeweils gewünschte Anzahl von verschiedenen Artikeln aus insgesamt drei Abteilungen aus und legt sie in den Warenkorb.

Dabei werden diese Daten in das Session-Array eingetragen.

Zu einem beliebigen Zeitpunkt kann der Benutzer sich den Inhalt des Warenkorbs ansehen und zur Kasse gehen. Der Warenkorb wird erst gelöscht, nachdem der Browser wieder geschlossen worden ist.



Abb. 1. Startseite des Webshops

Aufgabe 12.3: (Fortsetzung 1)

Nach Auswahl einer Abteilung erscheint eine Tabelle mit den Artikeln dieser Abteilung, wie in Abb. 2

Der Benutzer kann die jeweils gewünschte Anzahl von verschiedenen Artikeln eintragen und in den Warenkorb legen.

Der Warenkorb erscheint mit der bisher getroffenen Auswahl, wie in Abb. 3.

Abb. 3. Warenkorb

Falls der Benutzer weitere Artikel aus anderen Abteilungen auswählen möchte, kann er diese über die Startseite erreichen.

The screenshot shows a web browser window with the URL http://localhost/...op_b.php?abtnr=1. The page title is "localhost/sc_shop_b.php?abtnr=1". The heading "DVD/Video" is followed by the instruction "Wählen Sie aus:". Below this is a table with columns "Artikel", "Nr.", "Preis", and "Anzahl". The table contains five rows of data:

Artikel	Nr.	Preis	Anzahl
DVD-Recorder	4418	249,00 €	
DVD-Player	4422	49,95 €	1
Fernbedienung	4471	19,95 €	
Portable DVD-Kombi	4475	279,00 €	
DVD-Videokombi	4482	189,00 €	

[In den Warenkorb](#)

[Zur Startseite](#)

Abb. 2. Abteilung DVD/Video

The screenshot shows a web browser window with the URL http://localhost/...op_c.php?abtnr=1. The page title is "localhost/sc_shop_c.php?abtnr=1". The heading "Warenkorb" is followed by the message "Sie haben bisher gewählt:". Below this is a table with columns "Artikel", "Nr.", "Einzelpreis", "Anzahl", and "Gesamtpreis". The table contains two rows of data:

Artikel	Nr.	Einzelpreis	Anzahl	Gesamtpreis
DVD-Player	4422	49,95 €	1	49,95 €
Gesamteinkaufspreis				49,95 €

[Zur Kasse](#)

[Zur Startseite](#)

Aufgabe 12.3: (Fortsetzung 2)

Falls er zu einer Abteilung wechselt, in der er bereits Artikel ausgewählt hat, erscheint die eingegebene Anzahl des jeweiligen Artikels im Eingabefeld. Der Warenkorb füllt sich, siehe Abb. 4

Abschließend geht der Benutzer zur Kasse (die in Abb. 5 nur angedeutet wird).

Sicherlich ist dieses Beispiel noch unkomfortabel und einfach gehalten. Es zeigt aber das Wesentliche: die Übernahme und Aufbewahrung ausgewählter Daten im Session-Array.

Artikel	Nr.	Einzelpreis	Anzahl	Gesamtpreis
DVD-Player	4422	49,95 €	1	49,95 €
PMR-Funkgerätepaar	6213	29,95 €	2	59,90 €
Gesamteinkaufspreis				109,85 €

[Zur Kasse](#)

[Zur Startseite](#)

Abb. 4. Warenkorb nach weiteren Einkäufen

Bitte bezahlen Sie den Gesamteinkaufspreis von 109,85 €.

.....

[Zur Startseite](#)

Abb. 5. Abschluss des Einkaufs

Aufgabe 12.4: Diskussionsforum

Aufgabenstellung

Das Programm des Diskussionsforums, welches als PHP-Anwendung im Online-Campus zur Verfügung steht, soll

- (a) analysiert und
- (b) aktualisiert werden.

Die folgenden Dateien sind im Online-Campus bereitgestellt:

- *Index.php*, um die sämlichen Beiträge aufzulisten.
- *New.php* und *New.html*, um einen neuen Beitrag zu erstellen.
- *Read.php*, um einen konkreten Beitrag aus der Datenbank auszulesen.
- *Replay.php* und *reply_entry.php*, um eine Antwort auf einen Beitrag zu schreiben.

Quelle der Dateien: Seeboerger-Weichselbaum. M. (2004). PHP. Webseiten dynamisch programmieren. Rowohlt Taschenbuchverlag.

Aufgabe 12.4 (Fortsetzung 1)

a) Zur Analyse:

Das Anwendungsprogramm soll analysiert werden, um die Logik und den Aufbau des Programms zu verstehen.

b) Zur Aktualisierung:

Das Programm wurde mithilfe einer älteren PHP-Version erstellt. Ab PHP5 stehen manche Funktionen wie `HTTP_POST_VARS` nicht mehr zur Verfügung.

- Diese sollen durch neuere Variablen/Funktionen ersetzt werden, wie z.B. `$_POST` (für `$HTTP_POST_VARS`), `$_GET` (für `$HTTP_GET_VARS`), `$_COOKIE` (für `$HTTP_COOKIE_VARS`) oder `$_SERVER` (für `$HTTP_SERVER_VARS`).
- Darüber hinaus müssen die Befehle für Datenbankzugriffe, die mit "mysql_" beginnen, durch "mysqli_" ersetzt und angepasst werden.

Aufgabe 12.4 (Fortsetzung 2)

Mögliche Vorgehensweise zum Testen des Programms

1. Erstellen Sie eine Datenbank *manitu* mit dem Tool phpMyAdmin (oder über einen SQL-Befehl).
2. Erstellen Sie eine Tabelle *forum* mit den erforderlichen Datenfeldern, die Sie den bereitgestellten Programcodes entnehmen können.
3. Lassen Sie die alte Forum-Version mit den bereitgestellten Dateien ausführen.
 - *Index.php, New.php, New.html, Read.php, Replay.php, Replay_entry.php,*
4. Korrigieren Sie die angezeigten Fehler schrittweise und testen Sie den aktualisierten Programmcode.

Projektaufgabe: Entwicklung einer Webanwendung

- Um die in der Vorlesung gelernten PHP-Konzepte anzuwenden, soll ein kleines Anwendungssystem mit PHP individuell oder in zweier Gruppen entwickelt werden.
- Die Projektaufgabe wird zum angemessenen Zeitpunkt im Online-Campus zur Verfügung gestellt. Weitere relevante Informationen und Links finden Sie ebenfalls im Online-Campus.
- Für die Bearbeitung der Projektaufgabe werden drei Sitzungen eingeplant. Sie können die Fortschritte des Projektes (die Zwischenergebnisse bzw. Komponenten) in den einzelnen Sitzungen dem Dozenten kurz zeigen, um Rückmeldungen und Anregungen zu erhalten.
- Bitte beachten Sie auch die Abgabe- und Präsentationstermine für die Endversion Ihres Projektes, die in der Projektbeschreibung angegeben sind.