

# Web Server Side Scripting

mit PHP und MySQL



Version 2014 / 2015

## Sources

- Support de cours "Web Server side scripting", Robert Fisch, <http://www.fisch.lu>
- Support de cours "Web Server Side Scripting", Lycée technique d'esch-sur-alzette, <http://www.lte.lu/people/departements/informatique/cours/t2if-wsers1>
- Einstieg in PHP 5.5 und MySQL 5.6, Thomas Theis, Galileo Computing
- W3Schools, <http://www.w3schools.com>
- PHP.net, <http://www.php.net>
- Web Application Development, Gilles Everling, <http://wsers.foxi.lu/WebApplicationDevelopment.pdf>

## Zusammensetzung

- Patrick Eischen, Lycée des Arts et Métiers

## Inhaltsverzeichnis

Sources.....	2
Zusammensetzung.....	2
Inhaltsverzeichnis.....	3
<b>1 Einführung .....</b>	<b>4</b>
1.1 Was ist PHP?.....	4
1.2 Installation einer PHP-Umgebung.....	4
1.3 Mein erstes PHP-Skript.....	5
<b>2 Kurzer PHP-Programmierungskurs .....</b>	<b>6</b>
2.1 Variablen.....	6
2.2 Datentypen.....	6
2.3 Arbeiten mit Zahlen .....	7
2.4 Arbeiten mit Zeichenketten.....	7
2.5 Zufallszahlen .....	7
2.6 Verzweigungen.....	8
2.6.1 IF-Anweisung .....	8
2.6.2 Ternary IF .....	8
2.7 Wiederholstrukturen.....	9
2.7.1 FOR-Schleife.....	9
2.7.2 WHILE-Schleife.....	9
2.8 Felder (Arrays).....	9
2.8.1 Numerisch indizierte Felder.....	10
2.8.2 Assoziative Felder .....	10
2.8.3 Debugging mit der print_r-Funktion.....	10
2.9 Variablen und Elemente aus assoziativen Feldern löschen.....	11
<b>3 Eingabe von Daten .....</b>	<b>12</b>
3.1 Eingabe via HTTP GET Parameter .....	12
3.2 Existenz von Variablen prüfen .....	12

# 1 Einführung

## 1.1 Was ist PHP?

PHP ist eine Skriptsprache, die hauptsächlich zur Erstellung von dynamischen Webseiten oder Webanwendungen verwendet wird. Mit PHP lassen sich beispielsweise E-Commerce Systeme, Chats oder Foren erstellen. PHP unterstützt insbesondere die einfache serverseitige Auswertung von Formularen. Außerdem zeichnet sich PHP durch eine breite Datenbankunterstützung aus. Die meisten PHP-Entwickler setzen das Datenbanksystem MySQL ein. Das ist aber noch nicht alles, denn mit PHP können zudem Grafiken sowie PDF-Dokumente erstellt werden.

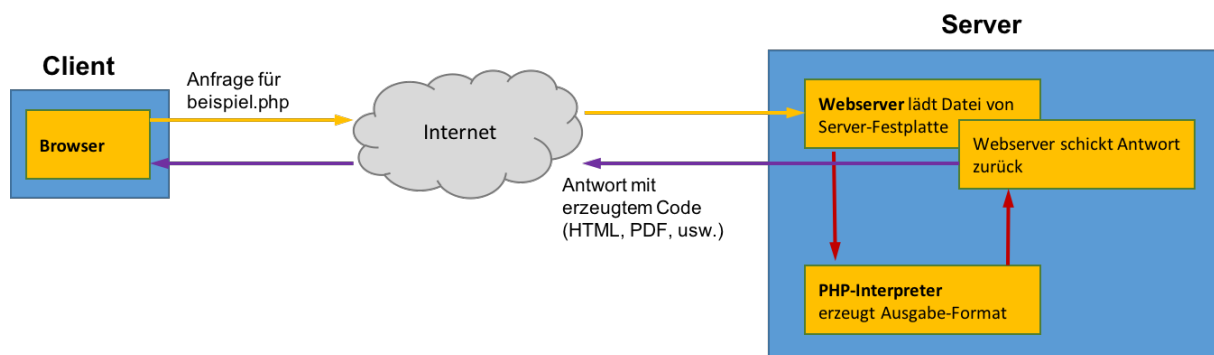
PHP steht heute übrigens für "**PHP** Hypertext **P**rocessor" (ursprünglich hieß es "**P**ersonal **H**ome **P**age Tools").

Hier die wichtigsten Vorzüge, welche PHP bietet:

- Es dient der Entwicklung von dynamischen Internetanwendungen.
- Es unterstützt verschiedene Plattformen.
- Es lässt sich leicht in den beliebten Apache-Webserver integrieren.
- Es ist kostenlos.

## 1.2 Installation einer PHP-Umgebung

Wird ein PHP-Skript angefordert, führt der PHP-Interpreter (PHP-Modul) dieser zuerst aus. Das Resultat, meistens ist dies HTML-Code, wobei aber auch andere Datentypen wie Bilder oder PDF-Dateien ausgegeben werden können, wird vom Webserver auf den Client zurückgesendet.



Der PHP-Quellcode wird also auf dem **Webserver** ausgeführt, während das erzeugte Ausgabeformat in den meisten Fällen weiterhin auf dem Client angezeigt wird. Von vielen Entwicklern wird Apache als Webserver bevorzugt, da dieser kostenlos ist und auf allen Plattformen installiert werden kann. Der **PHP-Interpreter** muss natürlich auch installiert und konfiguriert werden. Da **MySQL** perfekt mit PHP zusammenarbeitet, wird dieses von vielen Entwicklern direkt mitinstalliert.

### Apache + MySQL + PHP = AMP Environment

- Apache ist der Webserver
- MySQL ist der Datenbankserver (läuft aber üblicherweise auf dem gleichen Computer wie der Webserver)
- PHP ist der PHP-Interpreter/Modul und läuft natürlich auf dem gleichen Computer wie der Webserver

Man kann diese drei Komponenten einzeln herunterladen, installieren und konfigurieren, aber es geht auch einfacher mit **vorkonfigurierten Paketen**.

Downloade einfach eines dieser Pakete und befolge den Anweisungen des Installationsprogramms. Ein Apache-Server, PHP-Interpreter und MySQL sollten jetzt erfolgreich installiert sein.

- XAMPP <https://www.apachefriends.org> (Mac, Windows, Linux)
- MAMP <http://www.mamp.info> (nur für Mac)
- WAMPSEVER <http://www.wampserver.com/en> (nur für Windows)

Nach der Installation sollte man sich in der Konfigurationsdatei **php.ini** vergewissern, dass folgende Einstellungen die entsprechenden Werte besitzen:

Einstellung	Wert
register_globals (ab PHP 5.4 entfernt)	Off
short_open_tag	Off
display_errors	On
error_reporting	E_ALL
safe_mode (ab PHP 5.4 entfernt)	Off

### 1.3 Mein erstes PHP-Skript

Speichere folgendes Dokument auf dem Webserver unter **helloWorld.php** ab.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Mein erstes PHP-Skript</title>
  </head>
  <body>

    <?php
      // In PHP werden Variablen immer mit einem $ angegeben
      $name = "Donald Duck";
      echo "Hallo, mein Name ist " . $name . "!";
    ?>

  </body>
</html>
```

Im Browser kann die Webseite über die URL des Webserver aufgerufen werden. Wenn der Browser (auf dem Client) das vom PHP-Interpreter (auf dem Server) erzeugte Dokument empfängt, "sieht" er den PHP-Code dabei nicht mehr, da dieser bereits auf dem Server ausgeführt und durch die Ausgabe ersetzt wurde.

Ein PHP-Skript hat immer einen Dateinamen, welcher auf **.php** endet. Der PHP-Code selber, steht dabei zwischen den Markierungen **<?php** und **?>**.

## 2 Kurzer PHP-Programmierskurs

### 2.1 Variablen

Innerhalb eines Programms können Informationen zur späteren Verwendung in Variablen gespeichert werden. Variablen werden immer über ihren Namen identifiziert.

Für den Namen einer Variablen gelten folgende Regeln:

- **Er muss immer mit einem Dollarzeichen \$ beginnen.**
- Er darf keine Leerzeichen enthalten.
- Er darf nur aus Buchstaben und Ziffern sowie dem Sonderzeichen \_ bestehen, wobei das erste Zeichen ein Buchstabe sein muss.
- Er darf keine Umlaute (ä, ü, é, î, usw.), sowie kein ß enthalten.
- Er darf nicht mit einem reservierten Wort (z.B. for) identisch sein
- Es wird zwischen Groß- und Kleinbuchstaben unterschieden.

### 2.2 Datentypen

Eine Variable hat einen bestimmten Datentypen, welcher sich nach ihrem Inhalt richtet. Ist der Inhalt einer Variable z.B. 7, so hat sie den Datentypen Integer. Ist ihr Inhalt "Bob", so hat sie den Datentyp String. PHP unterstützt unter anderem Datentypen für:

Datentyp	Beispiel eines Wertes für diesen Datentyp
Integer (ganze Zahl)	7
Float/Double (Dezimalzahl)	7.125
String (Zeichenkette)	"Bob" "7"
Boolean (Wahrheitswert)	true false
Array (Felder)	{ "BMW", "Volvo", "Audi", "Opel" }
Objekte	<i>sind nicht Teil dieses Kurses</i>
Null	null

In PHP brauchen Variablen de facto nicht mit einem bestimmten Datentyp deklariert zu werden. Der Datentyp einer Variable richtet sich immer nach dem aktuellen Wert. So kann eine Variable zu einem Zeitpunkt im Programm eine Zeichenkette beinhalten, zu einem anderen Zeitpunkt aber eine Integer-Zahl. Dies wird auch noch als **"Type Juggling"** gezeichnet.

**Beispiel:**

```
<?php
$foo = "0";           // $foo is string
$foo = 2;             // $foo is now an integer
$foo = $foo + 1.3;    // $foo is now a float/double
?>
```

## 2.3 Arbeiten mit Zahlen

Bei Zahlen (Integer oder Float/Double) können folgende Rechenoperationen verwendet werden.

Operator	Bedeutung
+	Addition
-	Substruktion
*	Multiplikation
/	Division
%	Modulo-Operator: der Rest bei einer ganzzahligen Division. Z.B. ergibt 7 % 3 den Wert 1, denn 7 dividiert durch 3 ergibt 2, Rest 1

```
<?php
$a = 12;
$r = $a + 10;
$r++;           // Wert von Variable $r wird um 1 inkrementiert
$r += 0.5;      // Wert von Variable $r wird um 0.5 inkrementiert
$r /= 2;        // Wert von Variable $r wird durch 2 dividiert
?>
```

## 2.4 Arbeiten mit Zeichenketten

Will man 2 Zeichenketten verbinden (oder eine Zeichenkette mit einer Variablen), so wird in PHP der Punkt-Operator `.` benutzt.

```
<?php
$a = "Max";
$b = "Mustermann";
$c = $a . " " . $b;
$c .= ", old boy!";
echo "Hello " . $c;
?>
```

Benutzt man die doppelten Anführungszeichen (*quotes*) um eine Zeichenkette anzugeben, so können in PHP Variablenwerte auch innerhalb dieser ausgegeben werden.

```
<?php
$a = "Max";
$b = "Mustermann";
echo "Hello $a $b";
```

## 2.5 Zufallszahlen

Zufallszahlen können in PHP mit der Methode `rand()` generiert werden. Besser ist es allerdings die etwas neuere Methode `mt_rand()` zu benutzen, welche auch Zufallszahlen in einem Bereich [min, max] generieren kann.

```
$r = mt_rand(1, 100);
echo "Eine Zufallszahl zwischen 1 (inbegriffen) und 100 (inbegriffen): " . $r;
```

## 2.6 Verzweigungen

### 2.6.1 IF-Anweisung

```
<?php
$a = 2;
$b = 3;
if ($a > $b)
{
    echo "$a ist größer als $b";
}
else
{
    echo "$a ist kleiner als $b";
}
?>
```

Folgende Vergleichsoperatoren stehen uns zur Verfügung:

Operator	Bedeutung
<	kleiner als
>	größer als
<=	kleiner als oder gleich
>=	größer als oder gleich
==	gleich
!=	ungleich

Wenn mehrere Bedingungen in einer If-Verzweigung kombiniert werden sollen, werden die logischen Operatoren **&&** (UND), **||** (ODER) verwendet.

### 2.6.2 Ternary IF

Betrachten wir folgendes Beispiel:

```
<?php
$age = 19;
if ($age >= 18)    echo "Du bist volljährig";
else              echo "Du bist noch minderjährig";
?>
```

Dieses Skript kann mit einem Ternary IF wie folgt verkürzt werden:

```
<?php
$age = 19;
echo ($age >= 18 ? "Du bist volljährig" : "Du bist noch minderjährig");
?>
```

Der Ternary IF (auch noch "inline IF" oder "shorthand IF" genannt) hat folgende Syntax:

**(Bedingung ? Ausgabe\_wenn\_Bedingung\_erfüllt : Ausgabe\_wenn\_Bedingung\_nicht\_erfüllt)**

Oder umgangssprachlich ausgedrückt:

**Wenn ? Dann : Ansonsten**

## 2.7 Wiederholstrukturen

Falls sich innerhalb eines Programms einzelne Anweisungen oder Blöcke von Anweisungen wiederholen, werden Schleifen verwendet. In PHP gibt es unter anderem die for-Schleife und die while-Schleife.

### 2.7.1 FOR-Schleife

Die FOR-Schleife wird verwendet, um eine feste Anzahl an Wiederholungen zu erzeugen. Entweder ist diese Anzahl vorher bekannt, oder Start und Ende der Wiederholungen sind bekannt bzw. können errechnet werden.

```
<?php
    for ($i=1; $i<=5; $i++)
    {
        echo "<p>Zeile $i</p>";
    }
?>
```

### 2.7.2 WHILE-Schleife

Die WHILE-Schleife wird dazu genutzt, eine unbestimmte Anzahl an Wiederholungen zu erzeugen. Im nachfolgenden Beispiel wird gewürfelt. Es wird solange gewürfelt bis eine 6 gewürfelt wurde.

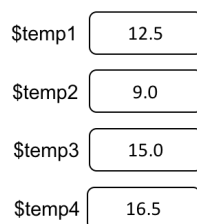
```
<?php
    $dice = mt_rand(1, 6);
    echo "<p>Würfel: $dice</p>";
    while ($dice != 6)
    {
        $dice = mt_rand(1, 6);
        echo "<p>Würfel: $dice</p>";
    }
    echo "<p><strong>Du hast eine 6 gewürfelt!</strong></p>";
?>
```

## 2.8 Felder (Arrays)

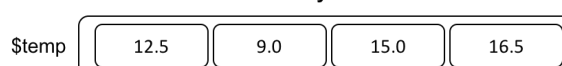
Nehmen wir an, es sei eine Woche lang jeden Tag an einem bestimmten Ort eine Temperatur gemessen worden. Es stehen somit 7 Temperaturwerte zur weiteren Betrachtung zur Verfügung. Diese Werte können jetzt in 7 einzelnen Variablen, jeweils mit einem eigenen Namen, gespeichert werden. Das ist aber relativ umständlich, besonders wenn anstatt an nur 7 Tagen, an 365 Tagen eine Temperaturmessung durchgeführt werden soll.

In diesem Fall bevorzugt man ein Feld (Array). Ein Feld ist einfach nur eine Liste von Variablen mit einem einheitlichen Namen.

#### 4 unterschiedliche Variablen



#### Array



PHP unterscheidet Arrays mit **numerischen Schlüsseln** und **Text-Schlüsseln** (assoziative Felder).

### 2.8.1 Numerisch indizierte Felder

In numerisch indizierten Feldern werden für den Zugriff fortlaufende Nummern verwendet.

```
// Erstellen und Füllen eines numerisch indizierten Arrays
$fruits = array("Apfel", "Banane", "Birne", "Ananas");

// Einzelne Elemente des Arrays anzeigen
echo "<p>" . $fruits[0] . "</p>";
echo "<p>" . $fruits[3] . "</p>";

// Überschreiben eines bestehenden Elementes
$fruits[2] = "Mango";

// Anfügen eines neuen Elementes ans Ende des Arrays
$fruits[] = "Erdbeere";

// Alle Elemente des Arrays mit FOR durchlaufen
for ($i=0; $i<count($fruits); $i++)
{
    echo "<p>" . $fruits[$i] . "</p>";
}
```

**\$fruits**

"Apfel"	"Banane"	"Birne"	"Ananas"
Index: 0	Index: 1	Index: 2	Index: 3

### 2.8.2 Assoziative Felder

Bei assoziativen Feldern werden für den Zugriff keine fortlaufenden Nummern benutzt, sondern String-Werte.

```
// Erstellen und Füllen eines assoziativen Arrays
$capitals = array(
    "Germany" => "Berlin",
    "France"  => "Paris",
    "Italy"   => "Rome",
    "Belgium" => "Brussels"
);

// Einzelne Elemente des Arrays anzeigen
echo "<p>" . $capitals["Germany"] . "</p>"; // schreibt "Berlin" auf den Bildschirm
echo "<p>" . $capitals["Belgium"] . "</p>"; // schreibt "Brussels" auf den Bildschirm

// Einfügen eines neuen Elementes
$capitals["Netherlands"] = "Amsterdam";
```

**\$capitals**

"Berlin"	"Paris"	"Rome"	"Brussels"
Key: "Germany"	Key: "France"	Key: "Italy"	Key: "Belgium"

Um assoziative Felder mit einer Schleife durchlaufen zu können, kann eine **FOREACH-Schleife** benutzt. Die Namen der Hilfsvariablen **\$key** und **\$value** können dabei frei gewählt werden.

```
foreach ($capitals as $key => $value)
{
    echo "<p>The capital of $key is $value</p>";
}
```

### 2.8.3 Debugging mit der print\_r-Funktion

Zu Testzwecken können alle Elemente eines Feldes am Bildschirm angezeigt werden.

```
// Das pre-Element wird hier benutzt, damit die Ausgabe im Browser formatiert wird
echo "<pre>".print_r($myArray, true)."</pre>";
```

## 2.9 Variablen und Elemente aus assoziativen Feldern löschen

In PHP ist es möglich, nicht mehr benutzte Variablen zu löschen. Dies geschieht mit der Funktion `unset()`. Dabei wird der Wert der Variable aus dem Arbeitsspeicher gelöscht und der Name wird von PHP "vergessen". Eine so gelöschte Variable darf danach nicht mehr benutzt werden. Es ist aber weiterhin möglich, eine neue Variable mit dem gleichen Namen wie die eben gelöschte Variable zu definieren.

Auf die gleiche Weise können auch einzelne Elemente eines assoziativen Felder gelöscht werden.

```
$capitals = array(
    "Germany" => "Berlin",
    "France"  => "Paris",
    "Italy"   => "Rome",
    "Belgium" => "Brussels"
);

unset($capitals["Belgium"]); // Element Belgium:Brussels wurde gelöscht
unset($capitals);           // Ganzes Feld wurde entfernt
```

## 3 Eingabe von Daten

In allen bisherigen Beispielen und Aufgaben fand keine Benutzereingabe statt. Alle Werte mit denen wir unsere Skripte gefüttert haben, waren vorher in Form von einer Variable definiert worden. Das wird sich jetzt ändern.

In PHP kann man kein Popup-Fenster für die Eingabe aufrufen, so wie wir es in JavaScript mit der prompt-Anweisung tun würden. Der Grund hierfür liegt darin, dass der Benutzer die Daten auf dem Client eingibt, PHP aber auf dem Server läuft.

### 3.1 Eingabe via HTTP GET Parameter

Mit Hilfe der HTTP GET Parameter können wir auf sehr einfache Weise Informationen an unser PHP-Skript übermitteln. Dazu werden die zu übergebenen Daten einfach an die URL angehängt.

**Beispiele:**

- `http://www.beispiel.com/mySkript.php?name1=wert1&name2=wert2&name3=wert3`
- `http://www.beispiel.com/sayHello.php?nachname=Mustermann&vorname=Max`
- `http://www.beispiel.com/volume.php?length=10&width=4&depth=2`

Um diese Informationen in unserem PHP-Skript zu verwenden, müssen wir die vordefinierte Variable `$_GET` benutzen. Die Variable `$_GET` enthält automatisch alle über HTTP GET Parameter übermittelte Werte in Form eines assoziativen Feldes.

**Beispiel:** Aufruf der Seite mit z.B. `webseite.php?nachname=Mustermann&vorname=Max`

```
<?php
echo $_GET["vorname"] . " " . $_GET["nachname"];
?>
```

### 3.2 Existenz von Variablen prüfen

Oft schreibt man PHP-Skripte, die auf Werte aus GET-Parametern angewiesen sind. Diese Skripte können keine sinnvolle Aufgabe erledigen, wenn einer der GET-Parameter fehlt. Natürlich ist es nicht ratsam, sich blind darauf zu verlassen, dass immer alle GET-Parameter vorhanden sind.

Mit der Funktion `isset()` gibt uns PHP die Möglichkeit zu testen, ob eine Variable existiert. Dies funktioniert sowohl für einfache Variablen wie auch für Felder oder spezifische Elemente eines Feldes.

```
<?php
if (isset($_GET["vorname"]) && isset($_GET["nachname"]))
{
    echo $_GET["vorname"] . " " . $_GET["nachname"];
}
else
{
    echo "Fehler: GET-Parameter nicht angegeben!";
}
?>
```