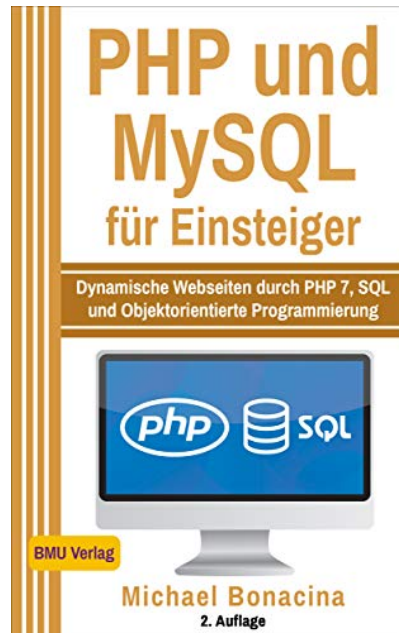


Webtechnologien

Prof. Dr.-Ing. Georg J. Schneider

Multimedia und Medieninformatik
Fachbereich Informatik
Hochschule Trier

Literatur



PHP 7 und MySQL: Von den Grundlagen bis zur professionellen Programmierung
Christian Wenz, Tobias Hauser
Rheinwerk Computing



PHP: und MySQL für Einsteiger:
Dynamische Webseiten durch PHP 7,
SQL und Objektorientierte Programmierung
Michael Bonacina, BMU Verlag

PHP

Was ist PHP?

- PHP: Hypertext Preprocessor
- Serverseitige Skriptsprache
- Besitzt Ähnlichkeiten zu C und Perl
- Kann in (X)HTML-Dokumente integriert werden
- PHP-Programme benötigen eine PHP-Engine und einen Webserver

PHP

Historie von PHP?

- PHP wurde 1995 von Rasmus Lerdorf entwickelt.
- PHP ist Open Source.
- Die PHP Gruppe kümmert sich um die Implementierungen von PHP:
<http://www.php.net>
- PHP 7.4.12 wurde am 29 Oktober 2020 veröffentlicht.

PHP

Warum PHP?

- Für einen Neueinsteiger einfach, für professionelle Programmierer viele fortgeschrittene Funktionen
- PHP gibt es für viele Betriebssysteme wie Linux, Windows und Mac OS X.
- PHP kann Formulardaten sammeln
- PHP kann Cookies senden und empfangen
- PHP kann Dateien auf dem Server erstellen, öffnen, lesen, schreiben, löschen und schließen.
- PHP unterstützt viele Datenbanken wie MySQL, MS SQL, Oracle, Sybase, PostgreSQL und viele andere.

PHP

Warum PHP?

- PHP kann dynamisch HTML, PDF, Flash, Text, CSV, XML und viele andere Ausgaben generieren.
- PHP kann zur Kontrolle des Benutzerzugriffs verwendet werden
- PHP kann Daten verschlüsseln
- Viele Webhosting-Optionen sind für PHP zu einem fairen Preis erhältlich.
- Viele Web Content Management Systeme sind in PHP geschrieben oder bieten Programmierschnittstellen an.

PHP

Was Sie lernen

- Was ist PHP?
- Erstes PHP-Skript
- PHP-Sprachelemente
- Formulare mit PHP verarbeiten
- Formulare mit PHP validieren
- Dateiverarbeitung mit PHP

PHP

Ausführen von PHP-Code

- Programme auf dem lokalen Rechner:
Web-Server (z.B. Apache), PHP (Interpreter), MySQL-Datenbank (optional)
- Entweder separat installieren oder alles in einem Setup-Paket namens WAMP, LAMP, MAMP und XAMPP herunterladen und installieren: WAMP (Windows, Apache, MySQL, PHP), LAMP (Linux, Apache, MySQL, PHP), MAMP (MAC, Apache, MySQL, PHP), XAMPP (Windows/Linux/MAC, Apache, MySQL, PHP): <http://www.apachefriends.org/>
- Nach der Installation den Apache-Server (und evtl. PHP) ausführen und starten
- Dann <http://localhost> in Ihren Web-Browser eingeben und schauen, ob der Webserver läuft
- Im Apache-Installationsordner ist der www-Ordner (meist „htdocs“), in dem PHP-Dateien gespeichert und über den Browser aufgerufen werden können

Erstes PHP-Skript

PHP-Skript

- Datei-Endung: standardmäßig php
- Die Dateierweiterung php führt dazu, dass das php-Skript ausgeführt wird. Wird z.B. die Dateierweiterung html verwendet, wird der Code **NICHT** ausgeführt.
- Codeblöcke enthalten PHP-Code:
 - `<?php ... ?>`: Empfohlene Standard-Markierung
 - `<? ... ?>`, `<% ... %>`: Kurzformen, evtl. Webserver Konfiguration anpassen

Erstes PHP-Skript

Beispiel

```
<?php  
    print "mein erstes PHP Programm";  
?>
```

```
<?php  
    print "<h1> mein zweites PHP Programm </h1>";  
?>
```

- HTML Elemente können in den PHP Code eingebettet werden.
- Ohne Anführungszeichen entstehen Fehler.

Erstes PHP-Skript

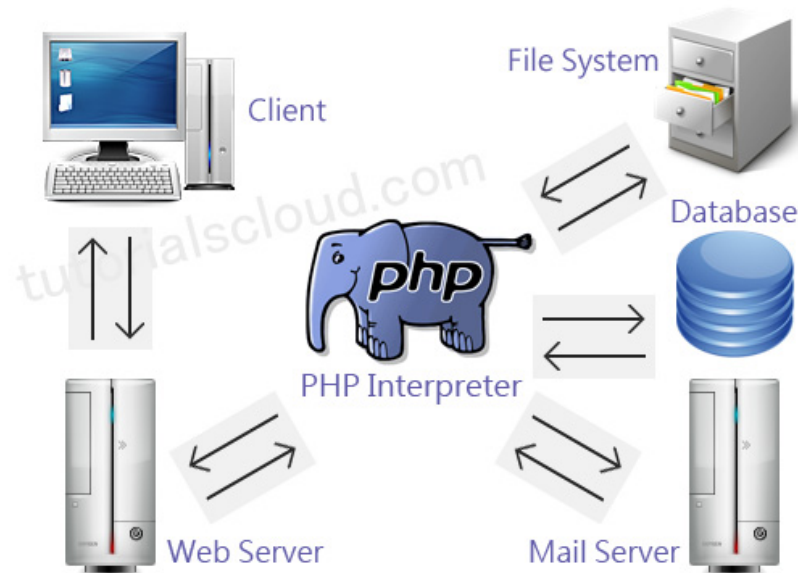
Verarbeitung

- Benutzer navigiert mit dem Browser zu einer Seite mit einer .php-Erweiterung, z.B.: index.php
- Der Webserver sendet die Datei an den PHP-Interpreter.
- Der Webserver sendet nur Dateien mit der Dateiendung .php an den Interpreter, alle anderen Dateien wie .html, .htm nicht!
- Der PHP-Interpreter findet alle öffnenden und schließenden PHP-Tags zu finden und verarbeitet den PHP-Code.
- PHP-Skripte werden auf dem Webserver interpretiert, und das Ergebnis (HTML) wird an den Client-Rechner zurückgeschickt.

Erstes PHP-Skript

Verarbeitung

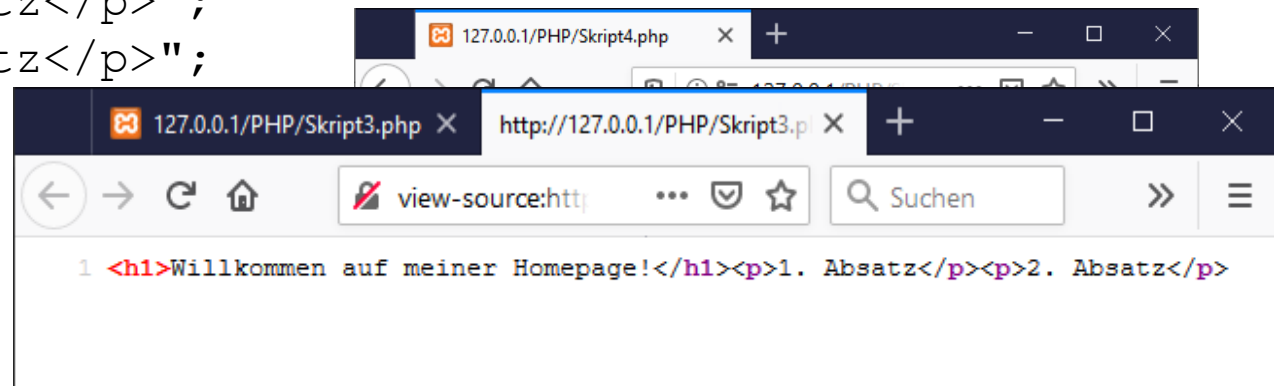
- PHP interpretiert die Seite und kommuniziert in erster Linie mit dem Dateisystem, der Datenbank und dem E-Mail-Server und liefert dann eine Webseite über den Webserver zum Browser zurück.



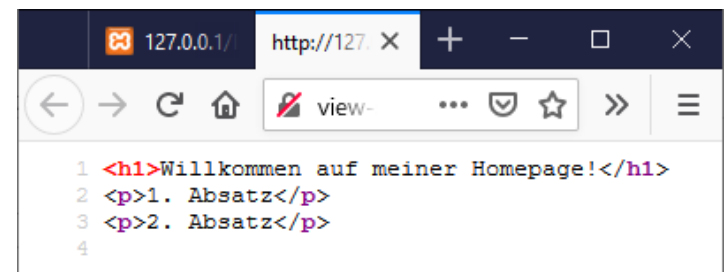
Zweites PHP-Skript

Beispiel

```
<?php
print "<h1>Willkommen auf meiner Homepage!</h1>";
print "<p>1. Absatz</p>";
print "<p>2. Absatz</p>";
?>
```



```
<?php
print "<h1>Willkommen auf meiner Homepage!</h1>\n";
print "<p>1. Absatz</p>\n";
print "<p>2. Absatz</p>\n";
?>
```



PHP

PHP in HTML

Beispiel

```
<h1>Hauptüberschrift für die Website</h1>
<?php
    print "<h2>Artikelüberschrift</h2>\n";
    print "<p>Textblock</p>\n";
?>
<p>Fußzeile</p>
```

- HTML und PHP Skripte können gemischt werden.
- Die Dateiendung muss weiterhin .php heißen.
- Methode zum flexiblen Anpassen von dynamischen Inhalten in eine statische Rumpfseite

PHP

PHP in HTML

Beispiel

```
<h1>Hauptüberschrift für die Website</h1>
<?php
    print "<h2>Artikelüberschrift</h2>";
    print "<p>Textblock</p>";
?>
<p>Fußzeile</p>
```

- HTML und PHP Skripte können in einer Datei kombiniert werden
- Die Dateierweiterung muss weiter auf .php stehen
- Methode zum flexiblen Anpassen von dynamischen Inhalten in eine statische Rumpfseite



PHP-Sprachelemente

- Einfache PHP-Elemente
- Operatoren in PHP
- Kontrollstrukturen in PHP
- Felder in PHP
- Funktionen in PHP
- Klassen in PHP

PHP-Elemente

Kommentare

```
//ein einzeiliger Kommentar
```

```
# noch ein einzeiliger Kommentar
```

```
/* ein mehrzeiliger  
   Kommentar */
```

PHP-Elemente

Variablen

- Müssen nicht deklariert werden
- Variablentyp ergibt sich aus dem zugewiesenen Wert
- beginnen immer mit einem \$-Zeichen
- Restlicher Name darf Buchstaben, Ziffern und Unterstriche () enthalten und muss mit einem Buchstaben oder Unterstrich beginnen.
- Groß- Kleinschreibung wird unterschieden.

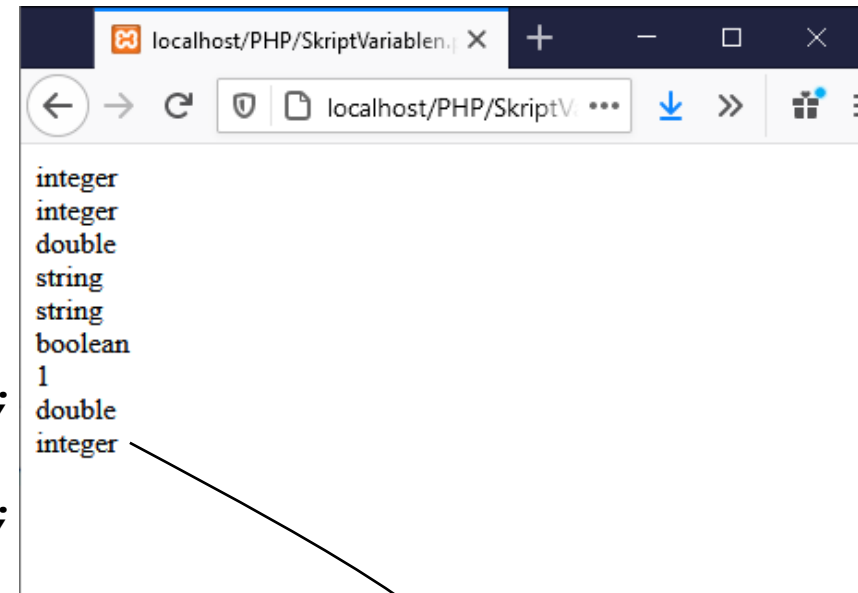
PHP-Elemente

Implizite Datentypen

- **integer-Variable:** `$weight = 61;` oder binär `$weight = 0b111101;`
- **double-Variable:** `$height = 1.7;`
- **string-Variable:** `$name = "Marie";`
- **Auch eine string-Variable:** `$surname = 'Risser';`
- **boolean-Variable:** `$istGesund = true;`
- Typdeklarationen für Funktionen möglich, erweitert in PHP7, auch für Rückgabetypen von Funktionen
- Typkonversion durch cast auf den gewünschten Typ
- Mit der Anweisung im Script: `declare(strict_types=1);` müssen Typdeklarationen für Variablen angegeben werden

PHP-Elemente

```
<?php
$weight = 61;
$width = 0b111101;
$height = 1.7;
$name = "Marie";
$surname = 'Risser';
$isGesund = true;
print gettype($weight) . "<br>";
print gettype($width) . "<br>";
print gettype($height) . "<br>";
print gettype($name) . "<br>";
print gettype($surname) . "<br>";
print gettype($isGesund) . "<br>";
print (integer)$height . "<br>";
print gettype($height) . "<br>";
settype($height, "integer");
print gettype($height) . "<br>";
?>
```



Ändert Typ und Wert
der Variablen

PHP-Elemente

Parameterübergabe

- Parameterübergabe bei der Zuweisung von Variablen: **Call by Value**

```
$a = 1.7;  
$b = $a;  
$b = 3.8; // $a hat den Wert 1.7
```

- Parameterübergabe bei der Zuweisung von Variablen: **Call by Reference**

```
$a = 5;  
$b = &$a;  
$b = 7; // $a ist jetzt auch 7
```

- Parameterübergabe bei Objekten immer: **Call by Reference**

PHP-Elemente

Beispiel

```
<?php  
$textbaustein = "Meine erste Variable";  
print $textbaustein;  
?>
```

- Definition von Textbausteinen
- Zuweisung an Variablen
- Integration an beliebigen Stellen in der Webseite und Ausdruck mit `print`

PHP-Elemente

Konstanten

- Kann man mit der Funktion `define()` definieren.
- Der Wert muss eine Zahl oder ein String sein.
- Konstanten werden ohne Angabe des `$`-Zeichen verwendet.

```
define ("KONSTANTE", 100);
```

```
$var = KONSTANTE * $var2;
```

PHP-Elemente

Ausdrücke Beispiele

- `$a = 5;`
- `$y = $a = $c = 3; // weist alles Variablen den Wert 3 zu`
- `$fahrenheit = $celsius * (9/5) + 32;`
- `$bmi = $weight / ($height * $height);`
- `$eins ? $zwei : $drei`
- `$f = verdoppeln($d); //verdoppeln ist eine Funktion`

PHP-Elemente

Strings

- Sowohl mit doppelten als auch mit einfachen Anführungszeichen
- Strings mit doppelten Anführungszeichen
 - ersetzen Variablen durch Werte
 - erkennen Sonderzeichen (Escape-Zeichen)
- `$result = "Ihr Gewicht ist $weight kg";` wird für `$weight = 61` interpretiert als: `Ihr Gewicht ist 61 kg`
- `$result = 'Ihr Gewicht ist $weight kg';` wird interpretiert als: `Ihr Gewicht ist $weight kg`
- `$result = "<table width=\"600\">";` wird interpretiert als: `<table width="600">`

Einfache PHP-Elemente

Anführungszeichen in Strings

- Maskieren mit \ Escape-Zeichen
 - \n für eine neue Zeile,
 - \" für das Anführungszeichen "
 - \\ für den Backslash \
 - \\$ für das \$-Zeichen
-
- `print "<table width=\"600\">";`
 - `print "<table width='600'>";`
 - `print '<table width="600">';`

Einfache PHP-Elemente

Verarbeitung von Strings (Auswahl)

- Anzahl der Zeichen

```
strlen("Hello world!"); // Ausgabe 12
```

- Anzahl der Worte

```
str_word_count("Hello world!"); // Ausgabe 2
```

- Rückwärts

```
strrev("Hello world!"); // Ausgabe !dlrow olleH
```

- Position des 2. Strings im ersten

```
strpos("Hello world!", "world"); // Ausgabe 6
```

- Ersetzung

```
str_replace("Sam", "Pete", "I am Sam"); // Ausgabe I am Pete
```

- Vergleich

```
strcmp("Hello", "Hello"); //Ausgabe 0 (da identisch)
```

PHP-Elemente

Ausgaben

```
print ("Ihr Gewicht ist $weight kg");  
print "Ihr Gewicht ist $weight kg";
```

Oder `echo`, `echo()`

Oder Heredoc bzw. Nowdoc Schreibweise für Strings:

```
<<<EOT  
    Text und mehr Text  
EOT;
```

Oder für formatierte Ausgabe `printf()`

PHP

Ausgaben

```
<?php
$str = <<<EOD
Beispiel für eine
Zeichenkette
über mehrere Zeilen
hinweg
mit heredoc-Syntax.
EOD;
```

```
/* Komplexeres Beispiel mit
Variablen. */
$foo = "Hallo";
$bar = "PHP";
echo <<<EOT
<p>Meine Begrüßung ist "$foo".</p>
<p>Ich programmiere "$bar".</p>
EOT;
?>
```

PHP

Ausgaben

```
<?php
```

```
$str = <<<EOD
```

```
Beispiel für eine  
Zeichenkette
```

```
über mehrere Zeilen  
hinweg
```

```
mit heredoc-Syntax.
```

```
EOD;
```



el mit

```
st "$foo".</p>
```

```
$bar".</p>
```

```
EOT;
```

```
?>
```

Operatoren in PHP

arithmetische Operatoren

- `+`, `-`, `*`, `/`, `%`, `**` (`**`: Exponential Operator)

Bit Operatoren

- `&`, `|`, `^`, `~`, `<<`, `>>`

Bedingungsoperator

- `?` `:`

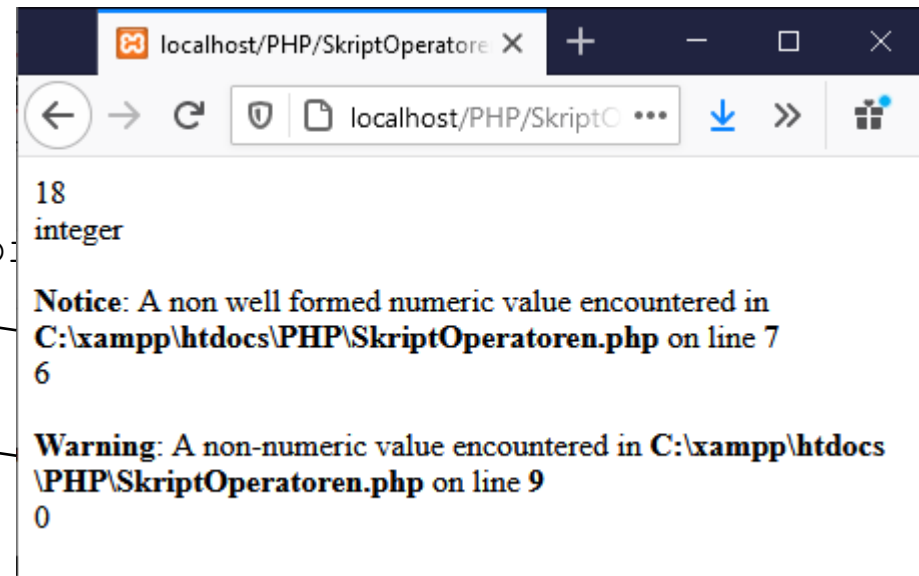
Verkettungsoperator

- Zeichen `"."`
- `"Fitness"."-". "Tipps"` ergibt die Zeichenkette `"Fitness-Tipps"`

Operatoren in PHP

Anpassung des Typs

```
<?php
$weight = 6;
$width = "3";
$height = $width * $weight;
print $height . "<br>";
print gettype($height) . "<br>";
$height = 2 * "3x";
print $height . "<br>";
$height = 2 * "x3";
print $height . "<br>";
?>
```



Operatoren in PHP

Zuweisungsoperatoren

- Zuweisung: `$a = $b;`
- `$a += $b;` entspricht `$a = $a + $b;`
- `$a .= "beta";` entspricht `$a = $a."beta";`
- Analog sind für die anderen arithmetischen Operatoren zusammengesetzte Zuweisungen definiert.

Auch übliche Kurzform zum Inkrementieren und Dekrementieren:

```
$i++; , $i--; ++$i; , --$i;
```

Operatoren in PHP

Vergleichsoperator

- `==`, `!=`, `<>`, `<`, `<=`, `>`, `>=`
- `!=` und `<>` sind identisch
- Zusätzlich Operator `===`, der prüft, ob Wert und Typ gleich sind
- `"7"===7` ergibt `false`, weil links eine Zeichenkette und rechts eine Zahl steht.
- `"7"==7` ergibt `true`, weil sowohl die Zeichenkette als auch die Zahl den Wert 7 besitzen.
- Bei Objekten prüft der Operator `===`, ob beide Referenzen auf dasselbe Objekt weisen.
- Ab PHP 7: `<=>` gibt -1, 0 oder 1 zurück, wenn der erste Operand kleiner, gleich oder größer als der zweite Operand ist

Operatoren in PHP

logische Operatoren

- `if ($a || $b)` bzw. `if ($a or $b)`: `$a` oder `$b` müssen `true` sein (auch beide)
- `if ($a xor $b)`: entweder `$a` oder `$b` müssen `true` sein (nicht beide)
- `if ($a && $b)` bzw. `if ($a and $b)`: `$a` und `$b` müssen `true` sein
- `if (! $a)`: `$a` darf nicht `true` sein

Operatoren in PHP

Fehler Operatoren

- @ Operator zur Fehlerkontrolle.
Stellt man das @ in PHP vor einen Ausdruck werden alle Fehlermeldungen, die von diesem Ausdruck erzeugt werden könnten, ignoriert.
- Vorsichtig mit dem Operator umgehen, weil Fehler nicht mehr angezeigt werden und die Fehlersuche damit schwierig wird.
- Anmerkung: ab PHP7 Fehlerbehandlung, ähnlich wie in Java mit `try catch`

Kontrollstrukturen in PHP

Auswahl

if-Anweisung

```
if (($zchn >= "0") && ($zchn <= "9"))  
{  
    print "Ziffer <br>";  
}  
else  
{  
    print "keine Ziffer";  
}
```

Kontrollstrukturen in PHP

Auswahl

elseif oder else if

```
if (($zchn >= "1") && ($zchn <= "9"))  
{  
    print "Ziffer 1..9";  
}  
elseif ($zchn == "0")  
{  
    print "Ziffer 0";  
}  
else  
{  
    print "keine Ziffer";  
}
```

Kontrollstrukturen in PHP

Auswahl

switch-Anweisung

```
switch ($wochentag)
{
    case "samstag": print "Endlich Samstag"; break;
    case "freitag": print "Bald ist Wochenende"; break;
    default: print "Frohes Schaffen";
}
```

Ausdrücke nach case sind auch möglich: `case $alter > 20 && $alter < 30:`
Anmerkung: In Java geht das nicht!

Kontrollstrukturen in PHP

Schleifen

while-Schleife

```
$celsius = 0;
while ($celsius <= 40)
{
    $fahrenheit = (( $celsius * 9 ) / 5 ) + 32;
    print "$celsius :  $fahrenheit <br/>";
    $celsius = $celsius + 10;
}
```


Kontrollstrukturen in PHP

Schleifen

do-while-Schleife

```
$celsius = 0;  
do  
{  
    $fahrenheit = (( $celsius * 9 ) / 5 ) + 32;  
    print "$celsius :  $fahrenheit <br>";  
    $celsius = $celsius + 10;  
} while ( $celsius <= 40 );
```

Kontrollstrukturen in PHP

Schleifen

for-Schleife

```
for ($celsius = 37; $celsius <= 40; $celsius++)  
{  
    $fahrenheit = (( $celsius * 9 ) / 5 ) + 32;  
    print "$celsius :  $fahrenheit <br>";  
}
```

break und continue ähnlich wie in Java

Felder in PHP

Numerisch indiziertes Feld

Erzeugung entweder

- `$produkt = array();`
- `$produkt[0] = 1;`
- `$produkt[1] = "Bohrmaschine";`

Oder

- `$names = array ("Stefan", "Andreas");`
- Feld mit Indexpositionen 0 und 1

Feldausgabe mit `print_r()` oder `var_dump()`

Felder in PHP

Numerisch indiziertes Feld

Zugriff über Index

- Element an vorhandenes Feld anhängen: `$names[] = "Markus";`
- `$names[2] = "Ina";`
- Dynamisch erweiterbare Felder
- Wert an beliebiger Indexposition zuweisen
- Dazwischenliegende Indexpositionen erhalten KEINE Werte

Feldlänge

- `$length = count ($names);`

Felder in PHP

Numerisch indiziertes Feld

Feld durchlaufen

- foreach-Schleife
- Iteriert von 0 bis n
- Für durchgängig besetzte Felder

```
foreach ($names as $name)
{
    print "$name <br>";
};
```

Felder in PHP

Assoziatives Feld

Assoziatives Feld

- Indizierung nicht über Zahlen, sondern über Zeichenketten

```
$telefonbuch = array("Stefan" => "01234/56789",  
                    "Andreas" => "01111/2323",  
                    "Ina" => "01234/98765" );  
  
$telefonbuch["Peter"] => "32168";
```

Felder in PHP

Assoziatives Feld

Zugriff

- `print $telefonbuch["Stefan"];`
- `print $telefonbuch['Stefan'];`

Assoziatives Feld durchlaufen

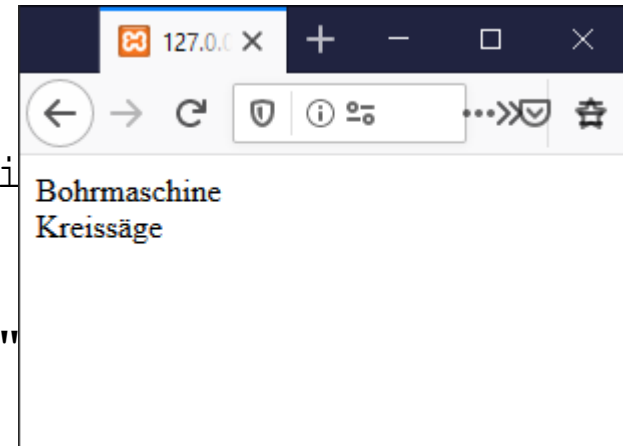
```
foreach ($telefonbuch as $key=>$value)
{
    print "$key: $value <br>";
};
```

Felder in PHP

zusammengesetzte Felder

Zugriff

```
<?php $sortiment = array();  
$sortiment[0]['Artikelnummer'] = 1;  
$sortiment[0]['Produktname'] = "Bohrmaschine";  
$sortiment[0]['Preis'] = 45;  
$sortiment[1]['Artikelnummer'] = 2;  
$sortiment[1]['Produktname'] = "Kreissäge";  
$sortiment[1]['Preis'] = 79;  
$sortiment[2]['Artikelnummer'] = 3;  
$sortiment[2]['Produktname'] = "Bandschleifer";  
$sortiment[2]['Preis'] = 89;  
  
print $sortiment[0]['Produktname']."<br>\n";  
print $sortiment[1]['Produktname']."<br>\n";  
?>
```



Felder in PHP

zusammengesetzte Felder

Zugriff

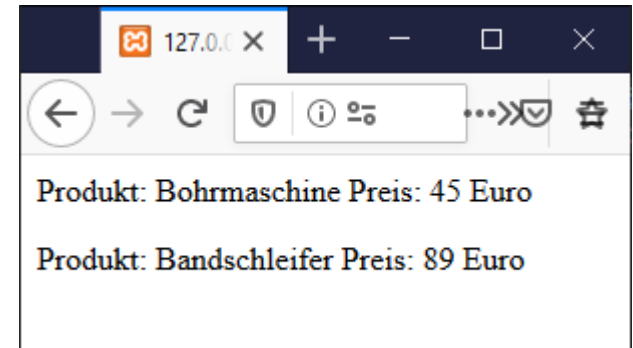
```
<?php
$sortiment = array();
$produkt[0]['Produktname'] = "Bohrmaschine";
$produkt[0]['Preis'] = 45;
$produkt[0]['Anzahl'] = 6;
$produkt[1]['Produktname'] = "Kreissäge";
$produkt[1]['Preis'] = 79;
$produkt[1]['Anzahl'] = 0;
$produkt[2]['Produktname'] = "Bandschleifer";
$produkt[2]['Preis'] = 89;
$produkt[2]['Anzahl'] = 11;
//-->
```

Felder in PHP

zusammengesetzte Felder

Zugriff

```
$i = 0;
while ($i < 3) {
    if ($produkt[$i]['Anzahl'] > 0) {
        print "<p>Produkt: " . $produkt[$i]['Produktname'] .
            " Preis: " . $produkt[$i]['Preis'] . " Euro</p>\n";
    }
    $i++;
}
?>
```



Funktionen in PHP

Funktionen programmieren

- Keine Unterscheidung bei Groß- Kleinschreibung im Namen
- Funktion ohne Parameter
- Fehlerunterdrückung mit `@funktionenname()`
- Variable `$bmi` ist eine lokale Variable

```
function bmi1()  
{  
    $bmi = 61.5 / (1.68 * 1.68);  
    $message = "1: Der BMI für 61.5 Kg und 1.68 m ist $bmi";  
    print $message;  
}
```

- Aufruf: `bmi1()` ;

Funktionen in PHP

Funktionen programmieren

- Eingabe- und Ausgabeparameter
- Übergabe der Eingabeparameter erfolgt mittels „Call By Value“
- Rückgabe über Ergebniswert (Variable oder Ausdruck)

```
function bmi3($weight, $height)
{
    $bmi = $weight / ($height * $height);
    return $bmi;
}
```

- Aufruf: `$result = bmi3 (61.5, 1.68);`

Funktionen in PHP

Funktionen programmieren

- Vorgabeparameter und variable Anzahl

```
function ausgabe($par1, $par2 = "Standard2") {  
    echo "Parameterwert 1: $par1<br>";  
    echo "Parameterwert 2: $par2";  
}  
ausgabe("Exklusiv1");
```

```
function funktion($a, ...$params) {  
    $elemente = count($params);           // ähnlich: func_get_args();  
    echo $elemente . '<br>';  
    echo $params[0] + $params[2];  
}  
funktion(0, 1, 2, 3);
```

Funktionen in PHP

Funktionen programmieren

- Parameter als Referenzen
- Eingabeparameter einer Funktion können in PHP auch durch Call by Reference übergeben werden
- Kennzeichnung durch &-Zeichen, z.B. `&$number1` statt `$number1`

```
function swap (&$number1, &$number2)
{
    $help = $number1;
    $number1 = $number2;
    $number2 = $help;
}
```

- Aufruf: `swap ($num1, $num2);`

Funktionen in PHP

Funktionen programmieren

- Funktionsname als Variable

```
function ausgabe($par) {  
    echo "Hallo $par";  
}  
$funktionsname = "ausgabe";  
$funktionsname("PHP 7");
```

Funktionen in PHP

Funktionen programmieren

- Gültigkeitsbereich von Variablen:
Lokale Variablen gelten in der Funktion, in der sie definiert wurden

```
<?php
function scope1(){
    $weight = 170;
    $height = 60;
}
scope1();
print $weight; // Keine Ausgabe, Variable existiert nicht mehr.
?>
```


Funktionen in PHP

Funktionen programmieren

- Gültigkeitsbereich von Variablen:
Außerhalb von Funktionen deklarierte Variablen gelten global

```
<?php
$weight = 170; //globale Variable
function scope2(){
    $height = 60;
}
scope2();
print $weight; // Ausgabe 170
print $GLOBALS['weight'] //alternativer Zugriff über $GLOBALS
?>
```

Funktionen in PHP

Funktionen programmieren

- Zugriff auf globale Variablen bei Namensgleichheit:

```
$weight = 60; $height = 1.85;  
function scope3() {  
    global $weight; //global1  
    global $height; //global2  
    $bmi = $weight / ($height * $height);  
}
```

- Fehlen die Zeilen
global1 und global2,
dann erfolgt **KEIN** Zugriff auf globale
Variablen, sondern es werden lokale -
undefinierte - Variablen angelegt.

Alternativ kann der Zugriff über \$GLOBALS erfolgen

Funktionen in PHP

Vorhandene Funktionen benutzen

Datum und Uhrzeit ermitteln

- `string date (string format, int timestamp)`
- Fehlt `timestamp`, dann wird Systemzeit verwendet

Parameter `format`

- `j`: Tag des Monats ohne führende Null
- `n`: Monat als Zahl ohne führende Null
- `Y`: Jahr als vierstellige Zahl
- `G`: Stunden im 24-Stunden-Format ohne führende Null
- `i`: Minuten (zweistellig mit führender Null)
- `s`: Sekunden (zweistellig mit führender Null)

Funktionen in PHP

Vorhandene Funktionen benutzen

- `$timestamp = time();`
- `$todayDate = date ("j.n.Y", $timestamp);`
- `$todayTime = date ("G:i:s", $timestamp);`
- `print "Heute ist der $todayDate und es ist $todayTime Uhr";`

Funktionen in PHP

Wdh: Beispiele String Verarbeitung

- `$brief = str_replace ("muß", "muss", $brief);`
Ersetzt „muß“ durch „muss“
- `$file_contents = str_replace ("</html>", "", $file_contents);`
löscht "</html>"
- `strlen ($name)`: Ermittelt die Länge von \$name.
- `strcmp ($name1, $name2)`: Vergleicht \$name1 und \$name2

Funktionen in PHP

Eingaben prüfen

- `bool isset ($var)`
 - Liefert `true`, wenn die Variable existiert, sonst `false`.
- `bool empty ($var)`
 - `if (!empty($var))` liefert `true`, wenn die Variable einen Inhalt enthält, sonst `false`.
- `is_numeric ($var)`
 - Prüft, ob `$var` eine Zahl oder ein numerischen String ist. Falls ja, dann wird `true` zurückgegeben, sonst `false`.

PHP

Funktionen auslagern

Funktionen in eigener Datei

- Funktion oder Sammlung von Funktionen in eigener Datei speichern.
- Funktionen dann in das Skript einbinden über:
 - `require` //bricht ab, wenn die Datei nicht vorhanden ist
 - `include` //fährt trotzdem mit der Abarbeitung fort
- Wird zum Modularisieren von Webseiten verwandt, z.B.:

```
<?php include ("header.inc.php"); ?>  
<article> <h2>Überschrift</h2> ... </article>  
<aside> <h3>Weiterführende Links</h3> ... </aside>  
<?php include ("footer.inc.php"); ?>
```

PHP

Funktionen auslagern

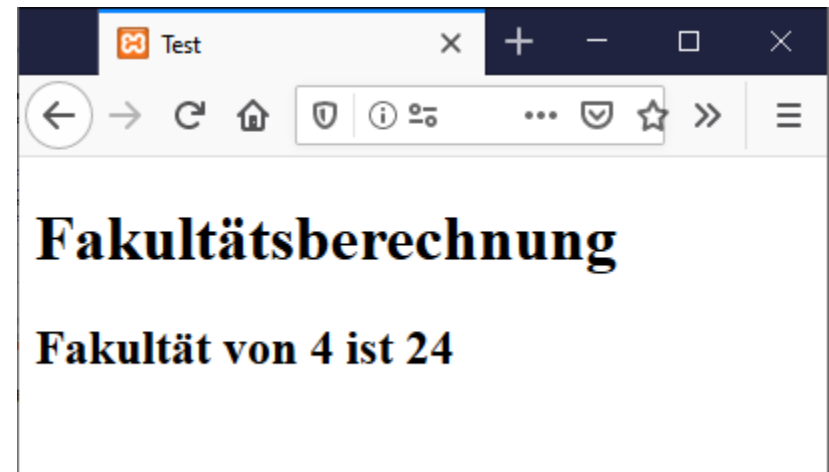
fakultaet.php

```
<?php
function fakultaet($arg){
    if ($arg == 0){
        return 1;
    }
    else{
        return $arg * fakultaet($arg-1);
    }
}
```


PHP

Funktionen auslagern

```
<!DOCTYPE html>
<html lang="de">
<head>
<meta charset="utf-8">
<title>Test</title>
</head>
<body>
<h1>Fakultätsberechnung</h1>
<p>
<?php
include ("fakultaet.php");
print "<h2>Fakultät von 4 ist ". fakultaet(4);
?>
</p>
</body>
</html>
```



PHP

Funktionen auslagern

Funktionen in eigener Datei

- Gültigkeitsbereich von Variablen
- Die Variable `$a` steht auch in den Funktionen des Skripts als globale Variable zur Verfügung:

```
<?php
```

```
$a = 1;
```

```
include 'b.inc';
```

```
?>
```

Klassen in PHP

Erweitertes Klassenkonzept ab PHP 5

- Attribute werden als Objekteigenschaften, Operationen als Objektmethoden bezeichnet.
- Sichtbarkeit: `private`, `protected` und `public` (default)
- Die Pseudovariablen `$this`: Referenz auf das jeweilige Objekt und steht innerhalb der Klasse zur Verfügung.
- Klassenmethoden und –attribute `static`. Zugriff: `Klassenname::`
- `final` ähnlich wie Java für Methoden und Klassen; `const` für (Klassen-) Attribute
- `parent::` Zugriff auf Superklasse (vgl. `super.` in Java)
- Überschreiben von Methoden und Attributen
- Abstrakte Klassen

Klassen in PHP

Klasse deklarieren

```
class Mitarbeiter
{
    // -- Attribute --
    private $personalnr;
    private $name;
    private $gehalt;
    //-- Konstruktor --
    function __construct ($nummer, $name, $bruttogehalt)
    {
        $this->personalnr = $nummer;
        $this->name = $name;
        $this->gehalt = $bruttogehalt;
    }
}
```

Klassen in PHP

Klasse deklarieren

```
// -- Methoden --
    public function erhoeheGehalt ($erhoehung)
    {
        $this->gehalt = $this->gehalt + $erhoehung;
    }

    public function ausgabe ()
    {
        return ("Personalnr: $this->personalnr,
                Name: $this->name,
                aktuelles Gehalt: $this->gehalt <br>");
    }
}
```

Klassen in PHP

Klasse deklarieren

Konstruktoren

```
function __construct ($nummer, $name, $bruttogehalt) { ... }
```

Konstruktoren werden nicht verkettet. Expliziter Aufruf des Konstruktors der Superklasse `parent::__construct()`;

Objekte erzeugen

- `$mitarbeiter = new Mitarbeiter (1234, "Meier", 2000.00);`

Auf Attribute zugreifen

- `$mitarbeiter->personalnr = "1234");`

Klassen in PHP

Operationen anwenden

- `$mitarbeiter->erhoeheGehalt(100.00);`
- `$ergebnis = $mitarbeiter->ausgabe();`

Objekte löschen

- `function __destruct()`
- Wird aufgerufen z.B. bei `unset($Mitarbeiter)` oder beim Verlassen des Skripts

Klassen in PHP

Weitere Konzepte der Objektorientierung

Klassenattribute und -methoden

- `private static $stundenlohn = 12;`
- `public static function aktuellerLohn() {...}`

Schnittstelle

- Schnittstellen können Attribute und Operationsköpfe enthalten

```
interface Schnittstellename {  
    function schnittstellenMethode();  
}
```

Schnittstellen werden von Klassen implementiert

- `class Mitarbeiter implements Ausgabe{}`

Klassen in PHP

Weitere Konzepte der Objektorientierung

Vererbung

- `class Teilzeitmitarbeiter extends Mitarbeiter`
 - Überschreiben von Methoden; Aufruf von Methoden der Superklasse
`parent::methodeName()`

Abstrakte Klasse

- `abstract class OneMoreClass { ... }`

Abstrakte Methoden

- `abstract public function oneMethod();`

Formulare mit PHP verarbeiten

GET und POST

- `method="GET"`: Eingabedaten des Formulars werden als Parameter an die Adresse angehängt (Voreinstellung)
- `method="POST"`: Formulardaten als Teil der Anfrage an den Webserver geschickt
- `POST` ist zu bevorzugen. Die Daten sind nicht in der URL zu sehen und es können größere Datenmengen versendet werden.

Formulare mit PHP verarbeiten

Superglobale Arrays

- Vordefinierte PHP-Arrays
- Werden automatisch mit Werten gefüllt
- Stehen in jedem Gültigkeitsbereich zur Verfügung
- `$_GET`
- `$_POST`
- `$_REQUEST` (für beide Übertragungsarten)

Formulare mit PHP verarbeiten

Eingabefelder an PHP-Skript übergeben

- Zuweisung der übergebenen Werte durch Zugriff auf die superglobalen Felder an Variablen
- `$height = $_GET['height'];`
- `$height = $_POST['height'];`
- `$length = $_REQUEST['height'];`

Im HTML Dokument wird das `name`-Attribut mit dem entsprechenden Wert gesetzt:

`<input name="height" ... > usw.`

Formulare mit PHP verarbeiten

Beispiel

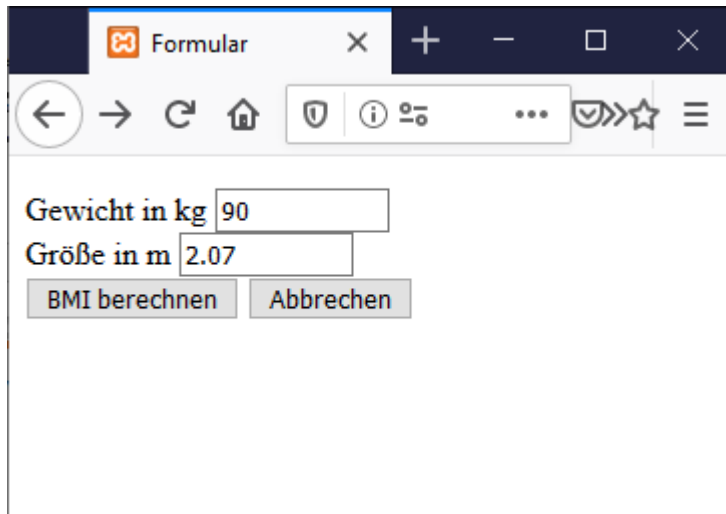
```
<html>
...
<form action="SkriptFormular1.php" method="POST">
  <p>
    Gewicht in kg <input name="weight" size="10" type="text"> <br>
    Größe in m <input name="height" size="10" type="text"> <br>
    <input name="Abschicken" type="submit" value="BMI berechnen">
    <input name="Abbrechen" type="reset" value="Abbrechen">
  </p>
</form>
...
</html>
```

Formulare mit PHP verarbeiten

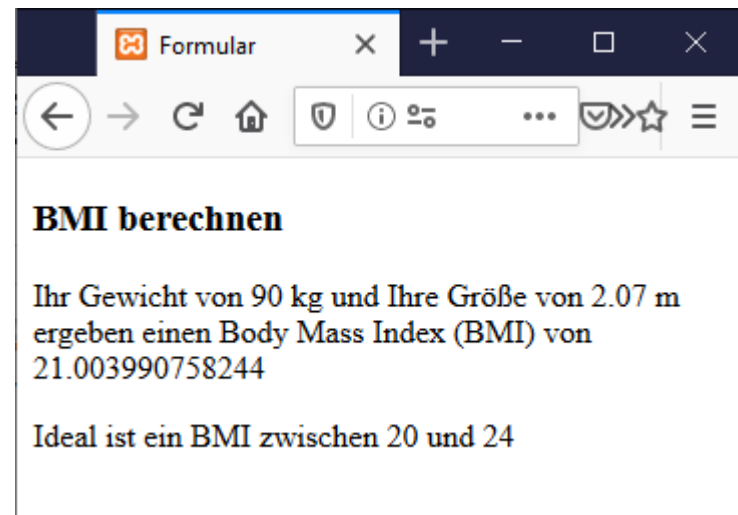
Beispiel

```
<?php
print "<h3>BMI berechnen</h3>";
$height = $_POST['height'];
$weight = $_POST['weight'];
$bmi = $weight / ($height * $height);
print "<p> Ihr Gewicht von $weight kg und Ihre Größe von
      $height m ergeben einen Body Mass Index (BMI) von
      $bmi </p>";
print ("<p>Ideal ist ein BMI zwischen 20 und 24</p>");
?>
```

Formulare mit PHP verarbeiten



A screenshot of a web browser window showing a form titled "Formular". The form contains two input fields: "Gewicht in kg" with the value "90" and "Größe in m" with the value "2.07". Below these fields are two buttons: "BMI berechnen" and "Abbrechen". The browser's address bar shows a lock icon, an information icon, and a percentage icon.



A screenshot of the same web browser window after the BMI calculation. The page title is "Formular". The main content area displays the text: "BMI berechnen", "Ihr Gewicht von 90 kg und Ihre Größe von 2.07 m ergeben einen Body Mass Index (BMI) von 21.003990758244", and "Ideal ist ein BMI zwischen 20 und 24". The browser's address bar shows the same icons as the previous screenshot.

Formulare mit PHP verarbeiten

Eingabefelder an PHP-Skript übergeben

- Nicht angeklickte Checkboxes werden nicht übertragen.
- Bei `checkbox` hinter den Namen eckige Klammern einfügen, damit alle Werte in einem Feld übertragen werden:
`<input type="checkbox" name="zutaten[]" ... >`
- Bei `option` Menüs mit Mehrfachauswahl (`multiple`) hinter den Namen eckige Klammern einfügen, damit alle Werte in einem Feld übertragen werden:
`<select id="payment" name="payment[]" multiple>`
`<option ...>`

Formulare mit PHP verarbeiten

Formular

← → ⓘ localhost/... ☆ Keine Synchronisierung

Ware

- Schokoriegel 1

Größe

Größe ☐ klein ☐ mittel ☒ groß

Sonderzutaten

- ☒ Kirschen
- ☒ Heidelbeeren
- ☐ Himbeeren

Bezahlung

Art der Bezahlung

☒ Quittung per E-Mail an folgende Adresse

Versand

Lieferanschrift

- Name

Zusatz

Ihre Nachricht an uns

absenden

```
var_dump($_REQUEST);  
var_dump($_GET);  
var_dump($_POST);
```

Formular

← → ⓘ localhost/PH... ☆ Keine Synchronisierung

```
array(9) { ["article-1"]=> string(1) "1" ["groesse"]=> string(1) "L"  
["zutaten"]=> array(2) { [0]=> string(8) "kirschen" [1]=> string(12)  
"heidelbeeren" } ["payment"]=> string(6) "master" ["email-receipt"]=> string(2)  
"on" ["email"]=> string(11) "me@home.com" ["recipient-name"]=> string(15)  
"Georg Schneider" ["message"]=> string(13) "Alles lecker!" ["Abschicken"]=>  
string(8) "absenden" }  
  
array(9) { ["article-1"]=> string(1) "1" ["groesse"]=> string(1) "L"  
["zutaten"]=> array(2) { [0]=> string(8) "kirschen" [1]=> string(12)  
"heidelbeeren" } ["payment"]=> string(6) "master" ["email-receipt"]=> string(2)  
"on" ["email"]=> string(11) "me@home.com" ["recipient-name"]=> string(15)  
"Georg Schneider" ["message"]=> string(13) "Alles lecker!" ["Abschicken"]=>  
string(8) "absenden" }  
  
array(0) { }
```

Formulare mit PHP validieren

Serverseitige Prüfungen mit PHP

Prüfung, ob Eingabe vorhanden sind

- `if (isset ($_POST['key']))`: Prüft, ob Parameter übermittelt wurde
- `if (empty($_POST['key']))`: Liefert `true`, wenn Feld leer ist oder nicht übermittelt wurde
- `if (array_key_exists('search', $_GET))`: Liefert `true`, wenn ein entsprechender Schlüssel im assoziativen Feld existiert.

Formulare mit PHP validieren

Prüfung, ob korrekte Zahl vorliegt

Alle Eingaben liegen als Zeichenketten vor

- `if (is_numeric($_POST[$form]))`: Liefert `true`, wenn numerischer String vorliegt
- `if (isset($_POST[$form]) && is_numeric($_POST[$form]))`: Liefert `true`, wenn Daten übermittelt werden und numerischer String eingegeben wurde.

Implizite Konvertierung

- `$var = 3 + "3";`: `$var` erhält den Wert 6

Formulare mit PHP verarbeiten

Formular

← → ⓘ localhost/... ☆ Keine Synchronisierung

Ware

- Schokoriegel 1

Größe

Größe ☐ klein ☐ mittel ☒ groß

Sonderzutaten

- ☒ Kirschen
- ☒ Heidelbeeren
- ☐ Himbeeren

Bezahlung

Art der Bezahlung

☒ Quittung per E-Mail an folgende Adresse

Versand

Lieferanschrift

- Name

Zusatz

Ihre Nachricht an uns

absenden

Formular

← → ⓘ localhos... ☆ Keine Synchronisierung

1 Schokoriegel

Der Größe

groß

Extras

Kirschen
Heidelbeeren

Bezahlt mit Kreditkarte

master

Als Quittung per Email

me@home.com

Zustellung an

Georg Schneider

Bemerkung

Alles lecker!

Formulare mit PHP verarbeiten

```
<form action="SkriptFormular3.php">
<fieldset>
  <legend>Ware</legend>
  <ul>
    <li>
      <label for="article-1">Schokoriegel</label>
      <input id="article-1" name="article-1" type="number"
        value="0">
    </li>
  </ul>
</fieldset>
```

Formulare mit PHP verarbeiten

```
<fieldset>
  <legend>Größe</legend>
  <p>
    Größe
    <label><input name="groesse" type="radio" value="S">
      klein</label>
    <label><input name="groesse" type="radio" value="M">
      mittel</label>
    <label><input name="groesse" type="radio" value="L"
      checked> groß</label>
  </p>
</fieldset>
```

Formulare mit PHP verarbeiten

```
<fieldset>
<legend>Sonderzutaten</legend>
<ul>
  <li><label><input type="checkbox" name="zutaten[]"
    value="kirschen"> Kirschen</label></li>
  <li><label><input type="checkbox" name="zutaten[]"
    value="heidelbeeren"> Heidelbeeren</label></li>
  <li><label><input type="checkbox" name="zutaten[]"
    value="himbeeren"> Himbeeren</label></li>
</ul>
</fieldset>
```

Formulare mit PHP verarbeiten

```
<fieldset>
<legend>Bezahlung</legend>
<p>
  <label for="payment">Art der Bezahlung</label>
  <select id="payment" name="payment">
    <option value="master">MasterCard</option>
    <option value="visa">VISA</option>
  </select>
</p>
<p>
  <input id="email-receipt" name="email-receipt" type="checkbox">
  <label for="email-receipt">Quittung per E-Mail</label>
  <label for="email">an folgende Adresse</label>
  <input id="email" name="email" type="email">
</p>
</fieldset>
```


Formulare mit PHP verarbeiten

```
<fieldset>
<legend>Versand</legend>
  <dl>
    <dt>Lieferanschrift</dt>
    <dd>
      <ul>
        <li>
          <label for="recipient-name">Name</label>
          <input id="recipient-name" name="recipient-name">
        </li>
      </ul>
    </dd>
  </dl>
```

Formulare mit PHP verarbeiten

```
<fieldset>
  <legend>Zusatz</legend>
  <p>
    <label for="message">Ihre Nachricht an uns</label>
    <textarea id="message" name="message"></textarea>
  </p>
</fieldset>
  <p>
    <input name="Abschicken" type="submit" value="absenden">
  </p>
</form>
```

Formulare mit PHP verarbeiten

```
<?php
print "<h1>".$_REQUEST['article-1']." Schokoriegel". "<h1>";
print "<h2>". "Der Größe". "</h2>";
switch($_REQUEST['groesse']) {
    case "S": print "klein";
    break;
    case "M": print "mittel";
    break;
    case "L": print "groß";
    break;
}
if (array_key_exists('zutaten', $_REQUEST)) {
    print "<h2>". "Extras". "</h2>";
    foreach ($_REQUEST['zutaten'] as $zutat) {
        print ucfirst($zutat). "<br>"; } //Großbuchstabe
}
```

Formulare mit PHP verarbeiten

```
print "<h2>". "Bezahlt mit Kreditkarte". "</h2>";
print $_REQUEST['payment']; //Besser switch, wie oben

if (array_key_exists('email-receipt', $_REQUEST)){
    print "<h2>". "Als Quittung per Email". "</h2>";
    print $_REQUEST['email'];
} else{
    print "<h2>". "Ohne Quittung". "</h2>";
}
print "<h2>". "Zustellung an". "</h2>";
@print $_REQUEST['recipient-name'];
//@: Fehlerunterdrückung bei leerem Feld
print "<h2>". "Bemerkung". "</h2>";
@print $_REQUEST['message'];
//@: Fehlerunterdrückung bei leerem Feld
?>
```

Formulare mit PHP - Sicherheit

Validierung mit Sicherheitschecks

- Zuerst die Daten an die PHP-Funktion `htmlspecialchars()` übergeben, um Code Injection zu verhindern.

Im Textfeld steht:

```
<Skript>Ortsangabe.href('http://www.hacked.com')</Skript>
```

Daraus wird:

```
&lt;Skript&gt;location.href('http://www.hacked.com')&lt;/Skript&gt;
```

und der Code wird nicht ausgeführt.

- Unnötige Zeichen (zusätzliches Leerzeichen, Tabulator, Zeilenumbruch) entfernen
PHP-Funktion `trim()`
- Backslashes (\) entfernen
PHP-Funktion `stripslashes()`

Formulare mit PHP - Sicherheit

```
<?php
// Alle Variablen definieren und leere Werte zuweisen
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name = test_input($_POST["name"]);
    $email = test_input($_POST["email"]);
    $website = test_input($_POST["website"]);
    $comment = test_input($_POST["comment"]);
    $gender = test_input($_POST["gender"]);
}

function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}

?>
```

Dateiverarbeitung mit PHP

Datei öffnen

- **Syntax:** `fopen (string filename, string mode, ...);`
 - Liefert: `resource`

mode kann sein:

- `r`: nur lesen, beginnt am Dateianfang. Datei muss existieren
- `w`: nur schreiben, beginnt am Dateianfang. Existiert die Datei nicht, wird versucht, sie zu erzeugen.
- `a`: nur schreiben, beginn am Dateiende. Existiert die Datei nicht, wird versucht, sie zu erzeugen.

Dateiverarbeitung mit PHP

- Rückgabewert der Funktion `fopen()` liefert Referenz auf Datei (*file handle*)
- Nach dem Öffnen wird die Datei nur noch über diesen *file handle* referenziert
- Beispiel: `$file = fopen ($filename, "r");` oder
`$file = fopen ($filename, 'r');`

Öffnen erfolgreich?

- Wenn eine Datei nicht geöffnet werden kann, liefert die Funktion `fopen()` den Wert `false` zurück.
- Testen, ob Öffnen erfolgreich war, mit `if` oder dem logischen Operator `die()`. Die Verwendung von `if` wird empfohlen.
- `$file = fopen ($filename, "r");`
`if ($file){...}`

Dateiverarbeitung mit PHP

Datei lesen

- N Bytes: `$file_contents = fread ($file, $length);`
- Komplette Datei: `$file_contents = fread ($file, filesize($filename));`
- Ein Zeile der Datei: `$file_contents = fgets($file);`

Datei beschreiben

- Variableninhalt schreiben: `fwrite ($file, $file_contents);`

Datei schließen

- `fclose ($file);`

PHP


Fallstudie: Gästebuch mit PHP programmieren

- Einfaches Gästebuch, um Fitness-Tipps zu sammeln
- Fitness-Tipps direkt als HTML-Dokument `alletipps.html` speichern
- Datei `fitnesstipps.php` bildet Rahmen und bindet `alletipps.html` ein.

Fit werden und fit bleiben

localhost/PHP/tipps.html

Keine Synchronisierung



[Startseite](#)
[Vitamine](#)
[Sport](#)
Deine Fitness Tipps

Fitness Tipps

Hier findest Du [Fitness Tipps](#) von Deiner Web Seite.

Sende uns Deine persönlichen Fitness Tipps. Wir freuen uns auf den Eintrag.

Tipps

Vor- und Zuname

Email

Ihre Nachricht an uns

Send

Abort



PHP

Einstiegsseite: **tipps.html**

```
<!DOCTYPE html>
...
<div class="content">
  <h1>Fitness Tipps</h1>
  <p>Hier findest Du <a href="allfitnesstipps.php">Fitness
    Tipps</a> von Deiner Web Seite.</p>
  <p>Sende uns Deine pers&ouml;nlichen Fitness Tipps. Wir freuen
    uns auf den Eintrag.</p>
  <form action="make_guestbook.php" method="POST">
    <fieldset>
      <legend>Tipps</legend>
      <p>
        <label for="name">Vor- und Zuname</label>
        <input id="name" name="name" size="40">
      </p>
    </fieldset>
  </form>

```

...

PHP

Erweitern der Datei und Anzeigen der Webseite:
make_guestbook.php

```
<?php
//Datenübernahme aus dem Formular von tipps.html
//Keine Prüfungen
$tipp = $_POST['tipp'];
$name = $_POST['name'];
$email = $_POST['email'];
addToGuestbook($tipp, $name, $email);
```

PHP

```
function addToGuestbook ($tip, $name, $email)
{
    $filename = "alltipps.html"; //Datei im gleichen Ordner

    //bisherigen Inhalt der Datei lesen
    $file_contents=file_get_contents($filename);

    //aktuelles Datum und aktuelle Zeit formatieren
    $timestamp = time();
    $todayDate = date ("j.n.Y", $timestamp);
    $todayTime = date ("G:i:s", $timestamp);

    $neuereintrag = "<article class='eintrag'>". "<h2
class='tipp'>". "$tip". "</h2>".
    "<div class='date'>".
    "Heute $todayDate um $todayTime wurde der Tipp von $name
gesendet".
    "</div>";
```

PHP

```
if ($email != "") {
    $neuereintrag = $neuereintrag."<div class='email'>".
    "email: "."<a href='mailto:$email'>$email</a>".
    "</div> </article>";
}
$neuereintrag = $neuereintrag." \n";

//Tipp anhängen
file_put_contents($filename, $neuereintrag, FILE_APPEND);

//Seiten Header schreiben
print"<!DOCTYPE html>
<html lang='de'>
<head>
<meta http-equiv='Content-Type' content='text/html; charset=utf-8'>
<title>Fit werden und fit bleiben</title>
<link href='stylesheet.css' rel='stylesheet' type='text/css'>
</head>
<body>";
```


PHP

```
print "Danke f&uuml;r den Tipp.<br>
    Er wurde zu den Fitness Tipps zugef&uuml;gt";
//Link zur&uuml;ck zur Tipps-Formular-Seite
print "<p><a href='tipps.html'>Zur&uuml;ck zum Formular</a></p>";

//Anzeigen der Tipps
include ("fitnesstipps.php");

//Ende der Seite
print"</body>";
print"</html>";
}
?>
```

PHP

fitnesstipps.php

```
<h1>Liste aller Fitnesstipps</h1>
<?php
    include ("alltipps.html");
?>
```

PHP

alltipps.html

```
<article class='eintrag'><h2 class='tipp'>Laufen</h2><div  
  class='date'>Heute 18.11.2020 um 20:50:13 wurde der Tipp von Georg  
  Schneider gesendet</div><div class='email'>email: <a  
  href='mailto:me@home.com'>me@home.com</a></div> </article>  
<article class='eintrag'><h2 class='tipp'>Schwimmen</h2><div  
  class='date'>Heute 18.11.2020 um 20:50:24 wurde der Tipp von Georg  
  Schneider gesendet</div><div class='email'>email: <a  
  href='mailto:me@home.com'>me@home.com</a></div> </article>  
<article class='eintrag'><h2 class='tipp'>Radfahren</h2><div  
  class='date'>Heute 18.11.2020 um 20:51:42 wurde der Tipp von Georg  
  Schneider gesendet</div><div class='email'>email: <a  
  href='mailto:me@home.com'>me@home.com</a></div> </article>
```

Zustände speichern mit PHP

Cookies

Cookies schreiben

- **Syntax:** `setcookie ("Bezeichnung", "Inhalt", Verfallsdatum);`
Verfallsdatum in Sekunden.
- **Beispiel:**
`<?php setcookie("Kundennummer", "12345", time()+2592000); ?>`

`setcookie` muss immer ganz am Anfang des Dokuments stehen – vor allen HTML-Tags, PHP-print-Befehlen oder anderen Ausgaben. Selbst Leerzeichen dürfen ihm nicht vorangestellt werden.

Cookies lesen

- `$_COOKIE['Kundennummer']`

Zustände speichern mit PHP

Sessions

- Sessions dienen dazu, mehrere Werte und Felder zu speichern.
- Sessions werden gelöscht, wenn der Benutzer den Browser schließt.
- Eine Session wird gestartet mit `session_start()`;
- Der Befehl muss auch zu Beginn stehen (vgl. Cookies)
- Die Inhalte sind in der Variable `$_SESSION['schluessel']` gespeichert
- Es kann mehrere Schlüssel geben und es können als Werte auch mehrdimensionale Felder gespeichert werden.

Beispiel: `$_SESSION['Warenkorb'] = [[54321, 1], [54326, 2]];`