Programmiersprache

PHP mit MySQL

(Zusammenfassung der Kurse „PHP und MySQL für Kids“ und „PHP Praxisbuch“ von J.C.Hanke)
(Stand 14.6.2013 / Korr. 11.2.2015 / Erweiterung 6.9.2017)

1. Einrichtung des lokalen Webservers:

- Installation mit Programm XAMPP — siehe Anhang 2
- Üblicher Stamm-Ordner für Html-Seiten wäre — C:\xammp\htdocs\html
- Apache-Server ist unter — C:\xammp\apache
- PHP-Dateien sind unter — C:\xammp\php
- MySQL-Dateien sind unter — C:\xammp\mysql
- Zugang zu Webserver ab Browser — http://localhost
- Zugang zu html-Dateien ab Browser — http://localhost/*.html
- Zugang zu php-Dateien ab Browser — http://localhost/*.php
- Document Root in apache/conf/**httpd.conf** an zwei Orten (in Document Root und <directory „D:\webseiten“>) auf D:/Webseiten/ geändert.
- Buchbeispiele sind zu finden unter http://www.phpkid.de
- **Passwort des Lokalen Host** mit phpMyAdmin einstellen bzw. ändern unter localhost > Rechte > neuen Benutzer hinzufügen > Benutzername (zawad_walo), Host (localhost) und Passwort einfüllen. Der Datenbank für Benutzer alle Rechte gewähren, Globale Rechte (alle) vergeben und mit OK abschliessen.
- **Passwörter einzelner DB** werden im übergeordneten Verzeichnis eingestellt

2. Werkzeuge:

- PHP-Editor Weaverslave / HTML-Editor GoLive
- PHP und MySQL für Kids / J.C. Hanke (Seitenangabe ->Ba123)
- PHP Praxisbuch / J.C. Hanke (Seitenangabe ->Pr123)

3. HTML:

- Grundgerüst mit DOCTYPE HTML PUBLIC (HTML4) — Ba98
- Grundgerüst mit DOCTYPE HTML (HTML5)
- Ueberschriften, Textabsatz, html-Tags — Ba31
- Bild einfügen / einmitten <p style="text-align: center;"> — Ba35
- Aufzählung (punkte, Zahlen, Radioknöpfe) — Ba37
- Hyperlinks zurück — Ba39
- Tabellen — Ba42
- Style Sheets (Verweis in Head, Inline-CSS, Farbgestaltung) — Ba44 - Ba51
- Seitenbreite (mit und oder mit css) — Ba52
- Formular — Ba54, Ba 163

4. PHP:

4.1 Allgemeine Regeln:

- PHP-Dokumente: sind html-Dokumente mit php-Umrandung (<?php …..?>)
- Es sind mehrere PHP-Abschnitte in einem PHP-Dokument möglich!
- Ausblenden von Programmteilen oder Bemerkungen mittels // xxxxx am Ende einer Zeile
- Ausblenden von Programmteilen oder Bemerkungen mittels /* xxxxxxx */ geht auch über mehrere Zeilen!
- Textausgabefunktion **echo “xxx“;** zeigt den Zeichenstring “ xxx“ auf dem Monitor an.

4.2 Regeln fürPHP-Programmierung:

Text-Strings immer in Gänsefüsschen oder Apostroph einhüllen / Zahlen aber nie!

Zeilenumbruch
im Browserabbild / \n im Quelltext. Kombinierte Verwendung
\n ist auch möglich.

Funktionen benötigen keine Gänsefüsschen und sind sehr oft mit einer Verkettungen (. FUNKTION .) eingefügt.

Strings-Verkettungen sind in php meist nicht nötig. **Verkettungsoperator** ist ein Punkt (“ . “). Zwischen und geht’s ohne!

Verkettungs-Kurzform: Abkürzung $content .=$name bedeutet eff. $content = $content$name; und $content .= $vorname. ”Zach”; bedeutet eff. $content = $content$vorname. ”Zach”;

Logische Operatoren: AND (&&) bzw. OR (||) Es gehen beide Varianten!

Vergleichsoperatoren (gleich == / grösser > / kleiner < / grösser gleich >= / kleiner gleich <= / ungleich != / false !== /)

Rechenfunkionen: Addition: + / Subtraktion: - / Multiplikation: * / Division: /

Balkendiagramme werden mit Tabellenzeilen erstellt

@ vor nicht immer unterstützten Funktionen verhindert Fehlermeldung des Servers

Variablen sind Platzhalter beginnend mit einem $-Zeichen und anschliessendem Variablenname (beginnend mit einem Buchstaben / keine Umlaute! / keine Leerzeichen!). Variablen benötigen keine Gänsefüsschen und sind sehr oft mit einer Verkettungen (. $xy .) eingefügt.

$abc=. . . .

Einfügen von PHP-Variablen in HTML-Dokumente (ausserhalb PHP-Abschnitt liegend!) mittels <?php echo $xy ; ?>
Einfügen von PHP-Variablen in HEREDOC-Abschnitte (in PHP-Abschnitt liegend!) mittels {$xy}
Konstanten: Wirkung wie Variablen / besitzen kein vorangestelltes $-Zeichen / lassen sich nicht umdefinieren oder zurücksetzen / Setzen mittels Funktion **define(‘Name‘,‘Wert‘)** / Zugriff von überall her mit Funktion **if defined(‘Name‘)**

Array ist eine Werteliste bestehend aus Feldvariablen (key-Werte).

$abc[´key0’]=value0 -> Kurzform: $abc = value0
$abc[´key1’]=value1 -> Kurzform: $abc = value1
etc.

Es wird Lang- und Kurzform unterschieden: Langform -> vollständige Aufzählung der Werteliste inkl. key untereinander. Kurzform -> serielle Aufzählung der Werte ohne key-Angabe
Arrays [key] sind Feldvariablen und machen aus der vorangestellten Variablen eine Werteliste. Sie können aus einer Zahl (0 … x / indizierte Arrays) oder aus einer Zeichengruppe (z.B. SUI / assoziative Arrays) bestehen.

Server-Variablen: $_SERVER[‚QUERY_STRING‘] (Pr293) für URL-Parameter nach ? / $_SERVER[‚PHP_SELF‘] (Pr54) für eigene Seite.

Superglobale (GPC)-Variable: $_GET, $_POST, $_COOKIE und $_REQUEST haben im ganzen Script (auch innerhalb von Funktionen) Gültigkeit. Verhinderung dass „normale“ Variable direkt (d.h. ohne if-Abfrage) durch „superglobale“ gesetzt werden können, erfolgt mittels Setzten der “register_globals“ in PHP.ini auf „off“.

Globale Variable: Variable die ausserhalb Funktionen deklariert wird und auch in Funktionen gültig ist.

Normale -Variable: Im Gegensatz zu superglobalen Variablen sind die normalen Variablen $abc, welche auch aus mit included eingebundenen Bereichen stammen können, nur ausserhalb von Funktionsscripts gültig.

Normale Variable innerhalb Funktions-Scrips: Variable, die innerhalb eines Funktions-Scripts definiert sind, gelten nie ausserhalb der Funktion. Beim Uebergang auf das Script und zurück werden die Werte in Form eines Array auf die neuen, im anderen Teil geltenden Variablen übergeben.

Reguläre Ausdrücke (Basis 212 – 214 / Praxis 184 - 191, 213)**.** Sind rafinierte Suchmuster- bzw. Suchmusterschablonen für gewisse Anwendungen (z.B. E-Mail, Hyperlinks, etc).

Ternärer Operator: Dieser Operator erlaubt die Zuweisung eines Wertes zu einer Variablen unter gewissen Bedingungen.

$Ausgabe = (Resultat der Bedingung) ? (Ausdruck_2) : (Ausdruck_3);

Wenn Resultat der Bedingung "true" ist, gibt der Operator Ausdruck 2 zurück. Ist Resultat der Bedingung dagegen "false", wird Ausdruck 3 zurückgegeben

Beispiel 1: $FROM = (!empty($_POST['From'])) ? trim($_POST['From']) : " ";

Beispiel 2: $ausgabe = (empty($PLZ)) ? "PLZ fehlt" : "Deine PLZ: $PLZ";

Formularabfragen: Pro PHP-Seite können mehrere Formulare mit verschiedenen Bezeichnungen und unterschiedlichen Zielen verwendet werden.

Herauslesen von Werten von Formular-Feldern: Bei GET-Uebermittlung mit URL-Übertragung (?start=xxx) -> mittels **GET[„name"]** und bei POST-Uebermittlung mit versteckter Übertragung -> mittels **POST[„name"]** **Bemerkung:** Die Daten werden jeweils nur einmal bei der Ansteuerung einer PHP-Seite übermittelt! Sie werden aber zu „included"-Dateien weitergegeben. Eine GET-Übermittlung kann maximal 10 Megabyte übertragen!

Heredoc-Operator: Wird mittels **<<<FREIERNAME** aktiviert und mittels **FREIERNAME;** wieder abgeschlossen. Erlaubt das Schreiben von ganz normaler HTML-Codierung innerhalb eines PHP-Dokumentes. Zu Beachten ist, dass beide Operatoren auf einer separaten Zeile sind, nie eingerückt und keine Leerzeichen vor bzw. nachher besitzen!

```
$html = <<<FREIERNAME
beliebiger HTML-Code
FREIERNAME;
```

In einem Heredoc-Bereich einzubringende PHP-Variablen (z.B. $xy) können nur mit {$xy} eingefügt werden! Die Methode mit <?php echo £xy; ?> geht nicht, da sich der Heredoc-Teil bereits innerhalb eines PHP-Abschnittes befindet.

4.3 Sicherheits- Regeln:

Fehlermeldungen ein / aus – schalten: Zum Programmieren immer einschalten mittels error_reporting = E_ALL in PHP.ini oder in jedem Script nach PHP-Eröffnung mittels error_reporting(E_ALL). Bei Betrieb sollten Fehlermeldungen in jedem Script nach PHP-Eröffnung mittels error_reporting(0) unterdrückt werden .

Setzen von Variablen bei Beginn eines Scripts: Variablen sollten immer am Anfang eines Scrips vordefiniert werden, damit sie nicht manipuliert werden können bzw. keine unerwartete Abläufe entstehen können.

Bei if-Abfragen mit Variablenvergleich: Immer mit = = vergleichen. Die Verwendung eines einzigen = wird als Setzen der entsprechenden Variablen interpretiert!

Code auslagern: Oft benötigter oder Code mit vertraulichen Daten immer in xxxx.inc.php Seiten auslagern und im Hauptscript mittels include_once "xxxx.inc.php" einbinden erhöht die Sicherheit und Ueberblickbarkeit. Diese Seiten können wegen der Endung "inc.php" nicht von Aussen eingesehen werden.

Rechtevergabe für Dateien kann normalerweise im WS_FT mittels CHMOD gesetzt werden. Üblich sind chmod 666 (lesen & schreiben exkl. Execute) und chmod 777 (alles erlaubt). Beim Webland-Host muss dies über die Konfiguration des Host (Konfiguration > Zugriffsrechte > Ordner wählen > Zugriffsrechte setzen) gemacht werden.

Datenbank Zugangsdaten als Zugriff.inc.php auslagern

4.4 Schleifen:

If-else Entscheidungsstruktur: Eine Bedingung wird auf Wahrheit geprüft. Wenn die Bedingung erfüllt ist tritt Fall A in Kraft; sonst führt das Programm Fall B oder C aus

```
if (Bedingung) {
Fall A;   // wird gemacht, wenn if-Bedingung erfüllt
}
elseif {
Fall B;     // wird gemacht, wenn if-Bed'g nicht erfüllt und elsif-Bed'g erfüllt
}
else {
Fall C;     // wird gemacht, wenn Bedingungen nicht erfüllt
}
```

Es können mehrere elseif Zweige eingeschoben sein. Der nächste elseif-Zweig wird nur dann ausgeführt, wenn der vordere nicht erfüllt werden konnte. If-else-Entscheidungen können verschachtelt oder in Serie gesetzt werden; Reaktion ist dabei unterschiedlich!

Der else-Teil kann auch weggelassen werden. In diesem Fall wird einfach nach der if-Endklammer mit Abarbeiten weitergefahren

Die **while-Schleife** wird ausgeführt solange Bedingung klar ist. Die Zählerprüfung findet im Schleifenkopf statt d.h. die Zählung erfolgt vor dem Abarbeiten des Befehls.

```
$i=1;
while ($1<=$_POST[`anzahl`]) {
echo "Happy Birthday";
$i++
}
```

```
$result = mysql_query_($sql)
while ($row=@mysql_fetch_assoc(result) {
$adr .= "$row[email]], ";
}
```

Bei der **dowhile-Schleife** wird der Befehl ausgeführt bevor die Bedingung geprüft wird. Die Zählerprüfung findet am Schleifenschluss statt d.h. der Befehl wird auch ausgeführt wenn die Zählbedingung von Anfang an falsch ist.

Die **for-Schleife** (normale Zählschleife) initialisiert die Zählvariable, den Test der Bedingung und legt das Zählmuster direkt am Schleifenkopf fest. Reagiert also wie eine komfortable while-Schleife

```
for ($i=1;$i<=$_POST[`anzahl`];$i++) {
echo "Happy Birthday";
}
```

Die **foreach-Schleife** durchläuft ein Array und gibt die dort gespeicherten Werte (key) und (value) zurück. Diese Schleife liest alle Elemente aus dem Array aus bis alle Keys durchgegangen sind!

```
foreach ($tag[0]="Sonntag";
echo "$value<br>\n";
}
```

```
                    Array      Element
foreach ($ausgabe as $value) {
echo "value";      -> Ausgabe solange etwas vorhanden
}
```

switch() Besser als verschachtelte if-else Entscheidungsstrukturen ist die Verwendung der switch-Funktion. Geht aber nur für ein und dieselbe Variable mit verschiedenen Werten!

```
switch($abs)  {       // legt zu untersuchende Variable fest
case „x“:             // setzt Wert „x“ für Variable ein
Befehl A;             // legt Aktion bei Uebereinstimmung mit x fest
break;                // springt zu default, wenn Befehl ausgeführt ist
case „y“:             // setzt Wert „y“ für Variable ein
Befehl B;             // legt Aktion bei Uebereinstimmung mit y fest
break;
default:              // bei keiner Übereinstimmung
Befehl D;             // legt Aktion bei keiner Uebereinstimmung fest
}
```

Fallunterscheidung mit switch() untersucht ob Variable mit den aufgelisteten Werten übereinstimmt. Geht nur für eine Variable!

4.5 PHP-Funktionen und Befehle (alphabetisch geordnet):

addstripslashes() setzt alle nötigen „escaping Backslashes (\)“ in einem String

count() Diese Funktion kann Elemente eines Arrays zählen

date(“Y“) kann Datum, Tag, Monat ermitteln. Y heisst z.B. „ganze Jahrzahl“

die(“Meldung“) kann laufendes Script mit Fehlermeldung abbrechen. Siehe auch Pt. 4.6.4

empty() Diese Funktion prüft ob eine Variable leer ist, nicht gesetzt ist bzw. den Wert 0 hat

empty($_POST[’x’]) prüft ob POSTx leer ist und gibt dann “true“. Bei nicht gesetzt oder Wert 0 erscheint “false“. Kann auch in invertierter Form (!empty()) verwendet werden; in diesem Fall wird bei nicht leerem Feld immer „true“ abgegeben

error_reporting(0) direkt nach PHP-Eröffnung (<?php) unterdrückt alle Fehlermeldungen & Notices des PHP-Teils. Sollte auf hochgeladene PHP-Seiten gesetzt werden.

error_reporting(“E_ALL“) direkt nach PHP-Eröffnung (<?php) zeigt alle Fehlermeldungen & Notices des PHP-Teils. Sollte zur Kontrolle der PHP-Programmierung eingesetzt werden.

exit() beendet Ausführung eines Scripts ohne Fehlermeldung. Siehe auch Pt. 4.6.3

explode(„Trennzeichen“,“Zeichenfolge“) (Basis 254 / Praxis 27, 36, 85 126). Wandelt eine Zeichenfolge mithilfe eines Trennzeichens in einen Array um welcher u.U. anschliessend mit foreach() aufgelöst werden kann.

fclose() schliesst Datei und gibt als Argument die Dateizeigerposition zurück

fgets(“Dateizeiger“,Länge_in_Byte) liest Daten zeilenweise aus einer Datei. Lesevorgang endet bei Zeilenende, Dateiende oder bei max Bytelänge. Dateizeiger geht ans Ende der Zeile.

filesize(“Dateiname“) ermittelt Grösse einer Datei

fopen(“Dateinamen“,“Modus“) öffnet Datei zum Lesen bzw. Schreiben. Modus r => lesen / r+ => lesen und w = schreiben ab Anfang. Positioniert Dateizeiger automatisch auf Dateibeginn.

foreach($ausgabe as $value) (Basis 85 / Praxis 27). Geht durch alle Zeilen des Arrays $ausgabe und legt jede Zeile als $value ab.

fputs(“Dateizeiger“,“Daten“) schreibt ab Position des Dateizeigers zeilenweise Daten

fread(fp,filesize) liest Binärdaten aus Datei fp entsprechend der Filegrösse

fseek(fp,Bit,??) verschiebt ab Datenzeiger um x Bits

getimagesize (file) erlaubt das Ermitteln von Bildinformationen in Array-Form

get_magic_goutes_gpc() zeigt die Einstellung (On oder OFF) der PHP-Konfigurations Option magic_quotes_gpc.

header(URL:...) wechselt auf Seite mit angegebenem URL. Geht nur, wenn vorher auf gleicher Seite kein anderes URL (z.B. css- oder form- oder echo „ xxxx"; Link) verwendet wird.

htmlspecialchars($Variable) wandelt < > & und " in html-code um und verhindert so das Einschmuggeln von html-Tags.

if_numeric() Diese Funktion prüft ob Klammerinhalt einen numerischen Inhalt hat

include() Diese Funktion bindet externe Dateien bzw. Funktionsmodule ein. Include() schaltet PHP-Mode ab! Die angesteuerte Datei wird dabei unabhängig von der Endung als html- bzw. Textdatei eingelesen. PHP-Teile müssen mit <?PHP und ?> eingebunden sein!

include_once() Diese Funktion ist wie include(); hier wird aber der Code nur einmal integriert.

isset() Diese Funktion prüft ob eine Variable gesetzt ist. Bei gesetzt = true sonst = false

isset($_POST['x']) prüft ob POSTx gesetzt ist und gibt dann "true". Bei leer oder Wert 0 wird „false"abgegeben. Kann auch in invertierter Form (!isset()) verwendet werden; in diesem Fall wird ausser bei gesetztem Feld immer „true" abgegeben.

mail("Empfängeradresse", "Betrefftext", "Botschaft","From: Absender") erlaubt Mail ab PHP-Server zu versenden

move_uploaded_file(filename, destination) ladet eine hochgeladene Date an ein vorgegebenes Ziel

mysql_fetch_assoc(result) erstellt aus einer zweidimensionalen Ergebnisliste ein zeilenorientiertes Array ($row). Leere Zeile am Schluss wird als „false" herausgegeben und führt zum Anhalten der Schleife.

mysql_num_rows(result) zählt Anzahl Zeilen der zweidimensionalen Ergebnisliste.

mysql_query(sql) führt SQL-Abfrage durch und gibt als Resultat ($result) eine zweidimensionale Ergebnisliste heraus.

mysql_affected_rows() gibt die erkannten Zeilen aus.

nl2br($comment) zeigt sichtbar den Zeilenumbruch im abgespeicherten Gästebuch (new line to break)

preg_match_all(Suchmuster,Zeichenkette,Treffer-Array) (Ba213 / Pr209, Pr213). Durchsucht Zeichenkette nach Übereinstimmung mit Suchmuster und gibt Übereinstimmungen als Treffer-Array ab. Sucht nach allen Übereinstimmungen und gibt als Rückgabewert eine Zahl >0 ab. Bei keiner Übereinstimmung wird eine 0 (false) abgegeben. Das Suchmuster muss mit Begrenzer eingerahmt sein, der weder Backslash noch Alphanumerisches Zeichen sein darf. Will man auf den Treffer-Array zugreifen, so muss als dritter Parameter ein Variablenname (z.B. $ausgabe) eingegeben werden, über den der Array geöffnet werden kann. Beispiele von Suchmustern:

für Zahlen:	`$muster = "/^[0-9]+$/"`
für Mailadressen:	`$muster = "/^[a-zA-Zäöüéèô0-9-_.]@[a-zA-Z0-9-_.]+$/"`
für Webseitenlinks:	`$muster = "/^http://[a-zA-Z0-9-_.]+$/"`
für Text mit Zahlen:	`$muster = "/^[a-zA-Zäöüéèô0-9-_.]+$/"`

preg_match(Suchmuster,Zeichenkette,Treffer-Array) (Ba213 / Pr209, Pr213). Durchsucht Zeichenkette nach Übereinstimmung mit Suchmuster, bricht bei erster Übereinstim-mung die Suche ab und gibt einen Rückgabewert von 1 (true) ab. Bei keiner Übereinstimmung wird ein

Rückgabewert von 0 (false) abgegeben. Das Suchmuster muss mit Begrenzer eingerahmt sein, der weder Backslash noch Alphanumerisches Zeichen sein darf. Will man auf den Treffer-Array zugreifen, so muss als dritter Parameter ein Variablenname (z.B. $ausgabe) eingegeben werden, über den der Array geöffnet werden kann. Beispiel eines Suchmusters:

Suche nach: {minigal,29,images3}
Suchmuster: `$muster = "|{ minigal,(.*?))}|si";`

preg_replace(Suchmuster,Ersatzstring,Zeichenkette) (Pr184, Pr196). Duchsucht Zeichenkette nach Übereinstimmung mit Suchmuster und setzt dort anstelle der Übereinstimmung den Ersatzstring ein. Das Suchmuster muss mit Begrenzer eingerahmt sein, der weder Backslash noch Alphanumerisches Zeichen sein darf. Der Ersatzstring muss ein regulärer Ausdruck beginnend mit einer Begrenzung sein. Sonderzeichen mit vorangestelltem Backslash (z.B. \[) maskieren. Der Ersatzstring darf nicht mit einem Sonderzeichen wie ? etc. beginnen.

rewind($fp) setzt filepointer (Dateizeiger) an den Anfang der Datei zurück.

round () macht eine Zahlenrundung entsprechend den Angaben in Klammer

set_magic_qoutes_gpc() ergänzt ´ und " und \ und 0 mit einem Backslash wenn sie in der PHP-Konfiguration eingeschaltet (d.h. = 1) ist. Normalerweise ist sie eingeschaltet!

sprintf() Diese Funktion formatiert eine Zahl entsprechend den Angaben in Klammer

strip-tags() entfernt alle vorkommenden HTML und PHP Tags in einem String

stripslashes() entfernt alle möglichen „escaping Backslashes (\)" in einem String

strlen() kontrolliert die Länge eines "String".

strtr() wechselt bzw. ersetzt Zeichen und Teilstücke in einem String.

str_replace("ä","ae",dateiname) wechselt bzw. ersetzt Zeichen in einer Datei.

setcookie("Cookiename", "Cookiewert", Ablaufdatum) setzt (Ablaufdatum: +86400*x) und löscht (Ablaufdatum: -3600) cookies. Ist ganz am Anfang der Html-Seite (vor Doctype!) zu verwenden. Auslesen mit $_COOKIE[„visit"]. Funktioniert bei Off-Line Tests auf dem eigenen Server in der Regel nicht!

readfile("Dateiname") liest Inhalt einer Datei aus

readdir("Verzeichnisname") liest Inhalt eines Verzeichnisses aus

substring("String",Startposition,Länge) schneidet String-Stücke aus einem Basisstring

substr_count("string $haystack, string $needle) ermittelt einen Teilstring ($needle) in einer Zeichenkette ($haystack)

trim() entfernt alle Leerzeichen, Zeilenumbrüche oder Tab-Sprünge am Anfang und Ende einer Zeichenkette

unlink (link, filename) erlaubt das Löschen von gespeicherten Dateien

4.6 Spezielle Funktionen:

4.6.1 Unterdrückung bzw. Umwandlung störender Zeichen:

```
$name = ‚xyz‘;
$array = (‘ä‘=>‘ae‘,‘ö‘=>‘oe‘,‘ü‘=>‘ue‘);
$neuername = strtr($name,$array);
```

4.6.2 Formular und Auswertung auf derselben Seite:

```
<form action = “<?php echo $_SERVER[‘PHP_SELF‘]; ?>“
```

4.6.3 exit()

Mit der Funktion **exit()** kann man die Ausführung eines Skripts beenden. Es ist zu beachten, dass das Skript nicht mehr fortgesetzt wird. Alles, was vor der Funktion aufgerufen wurde, wird an den Browser ausgegeben. Danach wird das Skript beendet. Es wird auch keine Fehlermeldung abgesetzt, ausser man erzeugt mittels echo unmittelbar vor der Funktion exit() selber eine.

```
exit();
```

4.6.4 die()

Mit der Funktion **die()** kann man ein laufendes Skript abbrechen und eine eigene Fehlermeldung (message) an den Browser senden. Diese Funktion wird oftmals genutzt, um zu überprüfen, ob eine Datei geöffnet werden konnte oder eine Verbindung zu einer Datenbank aufgenommen wurde. Eine Rückkehr zur Skriptausführung ist nach dem Aufruf von die() nicht mehr möglich.

```
die(“konnte Datei nicht öffnen“);
```

4.6.5 Herauslesen von übertragenen Formular-Werten:

Bei GET-Uebermittlung mit URL-Übertragung (?start=xxx) -> mittels **$_GET[„x“]**
und bei POST-Uebermittlung mit versteckter Übertragung -> mittels **$_POST[„x“]**
Mit **$_REQUEST[„x“]** können die Werte von POST, GET und COOKIE herausgelesen werden.
Bemerkung: Die Daten werden jeweils nur einmal bei der Ansteuerung einer PHP-Seite übermittelt! Sie werden aber zu „Included“-Dateien weitergegeben, da es superglobale Variablen sind.
Funktion **isset($_POST[’x’])** prüft ob POSTx gesetzt ist und gibt dann “true“. Bei leer oder Wert 0 wird „false“abgegeben. Kann auch in invertierter Form (!isset()) verwendet werden; in diesem Fall wird ausser bei gesetztem Feld immer „true“ abgegeben.
Funktion **empty($_POST[’x’])** prüft ob POSTx leer ist und gibt dann “true“. Bei nicht gesetzt oder Wert 0 erscheint “false“. Kann auch in invertierter Form (!empty()) verwendet werden; in diesem Fall wird bei nicht leerem Feld immer „true“ abgegeben

4.6.6 Zuweisung von Array-Werten an Variable:

Die Funktion **list()** weist den als Argumente übergebenen Variablen die Werte aus einem Array zu. Bitte beachten Sie, das es sich bei dem Array um ein numerisches Array handeln muss und dass dieser bei 0 beginnt.

```
$array = array ( 'PHP', 'ASP', 'Perl' );
list ( $php, $asp, $perl ) = $array;
print $asp "\n";
```

4.6.7 Weiterleitung mit der Funktion Header():

Die Funktion **Header()**

void **header** (string ***$string*** [, bool $replace= true [, int $http_response_code]])

wird zum Senden von HTTP-Anfangsinformationen (Headern) im Rohformat benutzt. Beachten Sie, dass Sie die Funktion **header()** aufrufen müssen, bevor Sie irgendeine andere Art von Ausgabe (seien es normale HTML-Tags, Leerzeilen in einer Datei oder von PHP) zum Client schicken. Es handelt sich hier um einen typischen Fehler, der zum Beispiel auftritt, wenn Sie Code mittels include() oder require() oder einer anderen Dateizugriffs-Funktion einlesen, die Leerzeichen oder Leerzeilen enthalten, die ausgegeben werden, bevor header() aufgerufen wird. Das gleiche Problem kann auch auftreten, wenn Sie eine Datei verwenden, in der HTML und PHP vermischt wurden

```
<html>
<?php
/* Dies wird einen Fehler provozieren. Beachten Sie die vorangehende Ausgabe, die vor dem Aufruf von header() erzeugt wird */
header('Location: http://www.example.com/');
?>

<?php
header("Location: http://www.example.com/"); // Browser umleiten
/* Stellen Sie sicher, dass der nachfolgende Code nicht ausgefuehrt wird, wenn
eine Umleitung stattfindet. /*
exit;
?>
```

4.7 Ausgelagerte eigene Funktionen:

```
Funktion ‘name’($a,$b,$c) {
 z.B. mail($a,$b,$c)
 …..
return “…”;        (mögliche Aussage über Resultat der Funktion / nicht nötig,
                    wenn keine Rückmeldung nötig ist)
```

Ausgelagerte Funktionen können von irgendwo im Programm aufgerufen werden, wenn sie „eingebunden sind“. Variable innerhalb einer Funktion gelten nicht ausserhalb der Funktion. Übergabe der Werte für die Funktion erfolgt in der Klammer nach dem Funktionsname. So wird z.B. name(100,A,X) in der Funktion zu $a=100, $b=A, $c=X.

5. Datenbank MySQL

5.1 Allgemeine Informationen:

- SQL = Structured Query Language
- MySQL wird normalerweise in Kommandozeilen-Ebene bedient
- phpMyAdmin = grafische Oberfläche für einfachere Bedienung des MySQL
- Datenbank ist übergeordnete Hülle und setzt sich aus einzelnen Datenbanktabellen zusammen
- Datenbankhost (Server) ist bei Webland: mysql7 und bei XAMP: localhost.
- DB-Benutzer (User) ist bei Webland: zawad_walo und bei XAMP: zawad_walo
- DB-Name (Name) ist bei Webland: zawad_walo und bei XAMP: zawad_walo
- DB_Passwort (Kennwort) ist bei Webland und lokal meist: halowalo
- Datentabelle für PHP-Kurs ist z.B. `adresse`,
- Feldnamen sind die Namen der Kolonnen (keine Umlaute und keine Trennstriche!)
- Schlüsselfeld (id) wird normalerweise mit AUTO_INCREMENT durchnummeriert
- Datensatz besteht aus Datenbankfenstern welche die Info enthalten
- Passwörter einzelner DB werden im übergeordneten DB-Verzeichnis eingestellt

5.2 SQL-Befehle:

Datentabelle erstellen:

```
CREATE TABLE adressen (
id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
Vorname VARCHAR(20),
Name VARCHAR(20),
Str VARCHAR(20),
PLZ VARCHAR(5),
Ort VARCHAR(30),
Tel VARCHAR(25),
Email VARCHAR(30),
WWW VARCHAR(20),
Notizen TEXT
)
```

Daten eingeben mit SQL:

```
INSERT INTO `adressen`
(id,Vorname,Name,Str,PLZ,Ort,Tel,Email,WWW,Notizen)
VALUES ('','Peter','Geppert','Hafenstrasse 7','76543','Küstenstadt','','geppert@lexi.de','','')
```

Alle Daten ausgeben mit SQL:

```
SELECT * FROM `adressen`
```

gezielt Daten ausgeben:

```
SELECT Name, Vorname FROM `adressen`
```

Daten ausgeben nach Suchkriterium:

```
SELECT *  FROM `adressen` WHERE Ort = 'Ostermundigen' OR Name=’Meier’
```

Daten Sortieren:

```
SELECT * FROM `adressen` ORDER BY `Ort, Name` DESC
```

Daten Updaten:

```
UPDATE `adressen` SET Str=’xy’,PLZ=’vvvv’ WHERE `id`=’s’
```

Daten beschränkt auslesen / ansteuern:

```
SELECT * FROM `adressen` DESC LIMIT ‘Start`, `Anzahl`
```

Daten Löschen:

DELETE FROM `adressen` WHERE `id`='s'

Daten exportieren:

1. phpMyAdmin öffnen & Datenblatt markieren
   Export öffnen
   Optionen > Struktur > füge DROP TABLE hinzu **)
   OK klicken
2. Angezeigter Exporttext markieren, kopieren und in die gewünschte Exportdatei einfügen

**) enthält am Beginn des Exporttextes die CREATE-Angaben des Datenbankblattes

Daten importieren:

1. phpMyAdmin öffnen & Datenbank markieren
   Import öffnen
   Datei eingeben
   OK klicken
2. Datenbank wir automatisch erstellt bzw. überschrieben und bereit gemeldet

Doppeleinträge mit **Unique Index** *verhinden:*

Schliesst aus, dass Datenbankeintragungen von (zwei) Spalten gleich sind. Unter Datenbanktabelle > Struktur > „Index über 1 Spalten anlegen" auf 2 vergrössern und unter Indextyp „Unique" und anschliessend die gewünschten Indexe wählen. Eingabe mit SQL:

```
ALTER TABLE  xxxxxx  ADD UNIQUE(
yy  (z.B. aid)
Datum
)
```

5.3 Befehle ab PHP-Scripts:

Verbinden mit Datenbank:

```
mysql_connect      ("server","Datenbankname","Kennwort")
                   ('mysql7','zawad_walo','halowalo')
                   ('localhost','zawad_walo','halowalo')
```

Zugriff auf Datenbank mittels ausgelagerter Datei:

Einbinden der Funktion auf php-Seite mittels Befehl "**include**(`zugriff.inc.php`)"

Aufbau der Zugriffs-Datei `zugriff.inc.php`:

```
<?php
@mysql_connect("localhost","Datenbank","Kennwort") or die („Verbindung gescheitert!");
@mysql_select_db(`Datenblatt`) or die („DB-Zugriff gescheitert!");
?>
```

Auswahl der Datentabelle:

```
mysql_select_db      (`adressen`)
```

Einfügen des Inhalts der Datentabelle 'adressen':

```
mysql_query('INSERT INTO `adressen` (id, A, B, …)')
```

Abfragen des Inhalts der Datentabelle 'adressen':

```
mysql_query('SELECT * FROM `adressen`')
```

Abfragen einer Datenzeile in Array-Form:

```
mysql_affected_rows(SELECT 'yxy' FROM 'abc' WHERE  'id'=z)
```

Löschen von Daten:

```
mysql_query('DELETE FROM `adressen` WHERE `id`='x'
```

Schliessen der Datenbank:

```
mysql_close(`$db')          =>.$db ist Datenbankname welcher weiter oben definiert ist!
```

6. Oft verwendete Script-Funktionen / Funktionsbibliothek

6.1 Umwandlung des PHP-Datumformats ins Format (TT.MM.JJJJ):

```php
<?php
function datemaker2($datum)    // Wandelt JJJ.-MM-TT in TT.MM.JJJJ um
{
  $arr_datum1 = explode(" ", $datum);
  $arr_datum = explode("-", $arr_datum1[0]);
  $datum = "$arr_datum[2].$arr_datum[1].$arr_datum[0], $arr_datum1[1] Uhr";
  return $datum;
}
?>
```

6.2 Auftrennung überlanger Strings (Seite 28 / Kapitel 1):

```php
<?php
function longkicker($arg)
{
  $ausgabe = explode(" ", $arg);
  $arg = "";
  foreach ($ausgabe as $value) {
    $arg .= wordwrap($value, 40, " ", 1) . " ";
  }
  return trim($arg);
}
echo longkicker($string);
?>
```

6.3 Suchscript (Seite 25 / Kapitel 1):

```php
<?php
$nadel = "Heu";              // Hier Suchwort eintragen
$heuhaufen = "Das ist ein großer Haufen Heu";
if (strpos($heuhaufen,$nadel) > 0) {
  echo "Wort <b>$nadel</b> enthalten in:";
  echo "<div>$heuhaufen</div>";
} else {
  echo "Wort <b>$nadel</b> nicht enthalten!";
  }
?>
```

7. Datenbank Funktionen:

7.1a Konfigurations-Funktion für DB-Zugang (z.B. config.inc.php für Webland):

```php
<?php
$dbname = "zawad_walo"; // Name der Datenbank
$dbhost = "mySQL7"; // Datenbankhost, meist localhost
$dbuser = "zawad_walo"; // Benutzername für MySQL
$dbpassword = "halowalo"; // Passwort für MySQL
?>
```

7.1b Einschalt-Funktion für Datenbank (z.B. zugriff.inc.php):

```php
<?php
@mysql_connect("$dbhost", "$dbuser", "$dbpassword")
or die("Verbindung zum Datenbankserver gescheitert!");
@mysql_select_db("$dbname") or die("Datenbankzugriff gescheitert!");
?>
```

7.2 Abfrage von Datenbankinhalten

```php
<?php
include("config.inc.php");                  // Konfigurationsdaten laden
include("zugriff.inc.php");                 // DB aktivieren
$sql1 = "SELECT * FROM f24_fotoblog";       // Def. Abfrage 1
$sql2 = "SELECT * FROM f24_fotoblog ORDER BY id DESC";  // Def. Abfrage 2
// Abfragen machen
   $result1 = @mysql_query($sql1);
   $zeilen = @mysql_num_rows($result1);
   $result2 = @mysql_query($sql2);
// Einträge anzeigen
echo "<p>Anzahl der Einträge: $zeilen</p>\n";
// Beginn while-Schleife zur Anzeige dieser Einträge
while ($row = @mysql_fetch_assoc($result2)) {
  $Eintrag = nl2br($row['Eintrag']);
  $groesse = @GetImageSize("bilder/" . $row['Bild']);
  $output .= "<h3>$row[Headline]</h3>\n
<p><img src='bilder/$row[Bild]' $groesse[3] alt='$row[Headline]'></p>
<div>$Eintrag</div>" . "<p>eingetragen von <b>$row[Name]</b>";
 $output .= "<p class='linie'> </p>\n";
 }                                                  // while-Schleifen Ende
 echo $output;
 mysql_close();                                     // Datenbank schliessen
?>
```

7.3 Anzeige von Daten mit Bildern:

```php
<?php
if (isset($_GET['id']) && is_numeric($_GET['id'])) { // Variablenkontrolle
   $sql1 = "SELECT * FROM f24_fotoblog";
   $sql2 = "SELECT * FROM f24_fotoblog ORDER BY id DESC";
   $result1 = @mysql_query($sql1);
   $zeilen = @mysql_num_rows($result1);
   $result2 = @mysql_query($sql2);
echo "<p>Anzahl der Einträge: $zeilen</p>\n";       // Anzeige der Einträge
while ($row = @mysql_fetch_assoc($result2)) {      // while-Schleife Anfang
  $Eintrag = nl2br($row['Eintrag']);
  $groesse = @GetImageSize("bilder/" . $row['Bild']);
  $output .= "<h3>$row[Headline]</h3>\n     // zusammensetzen der Anzeige
<p><img src='bilder/$row[Bild]' $groesse[3] alt='$row[Headline]'></p>
<div>$Eintrag</div>" . "<p>eingetragen von <b>$row[Name]</b>";
 $output .= "<p class='linie'> </p>\n";
 }                                                  // while Ende
 echo $output;                                      // Daten anzeigen
 mysql_close();                                     // Datenbank schliessen
}
?>
```

7.4 Löschen von Datenbank-Einträgen bzw. -Zeilen

```php
<?php
if (isset($_GET['loeschen']) && isset($_GET['id']) // Variablenkontrolle
   && is_numeric($_GET['id'])) {
    include("zugriff.inc.php");                          // DB aktivieren
  $sql = "DELETE FROM f24_fotoblog WHERE id='$_GET[id]'";  // Abfrage
                                                          // vorbereiten
  if (mysql_query($sql) && mysql_affected_rows() > 0) {    // Löschen
    echo "<h3>Löschupdate <strong>erfolgreich</strong>,     // Erfolg-Text
    Datensatz $_GET[id] gelöscht!</h3>";
  } else {
    echo "<h3>Löschupdate <b>nicht</b> erfolgreich!</h3>"; // Warntext
  }
}
?>
```

7.5 Ändern (Update) von Datenbankeinträgen

```php
<?php
// Update der ganzen Zeile nach Formulareingabe
if (!empty($_POST[idn]) && !empty($_POST[Vorname]) &&!empty($_POST[Name])
    && !empty($_POST[Str]) && !empty($_POST[Plz]) && !empty($_POST[Ort])
    && !empty($_POST[EMail]) && !empty($_POST[WWW])
    && !empty($_POST[Geburtstag])) {
include("zugriff.inc.php");                                  // DB aktivieren
$sql = "UPDATE adressen SET" .
"Vorname='$_POST[Vorname]',Name='$_POST[Name]',".
" Str='$_POST[Str]', PLZ='$_POST[Plz]', Ort='$_POST[Ort]',
WWW='$_POST[WWW]', ".
" Tel='$_POST[Tel]', EMail='$_POST[EMail]', " .
" Geburtstag='$_POST[Geburtstag]' WHERE id='$_POST[idn]' ";
mysql_query($sql);                                           // DB-Zeile updaten
mysql_close();                                               // DB trennen
    echo "<p><h4>Update erfolgreich durchgeführt! </h4></p>";
} else {
 echo " <h3><b>Bitte Felder zuerst vollständig einfüllen!</b></h3>";
}
?>
```

7.6 Auslesen von Datenbankeinträgen in Tabelle

```php
<?php
include("zugriff.inc.php");                                  // DB aktivieren
$sql = "SELECT id, Vorname, Name, Str, PLZ, Ort, Tel, EMail, ".
"WWW, Geburtstag FROM adressen WHERE id='$_POST[idn]'";  // Abfrage def.
$result=mysql_query($sql);                                   // DB Abfage
echo "<table border='1' cellspacing='0'>\n";        // Tabelle definieren
echo "<tr><th>id</th><th>Vorname</th>".          // Kopfzeile der Tabelle
"<th>Name</th><th>Str</th><th>PLZ</th>".
"<th>Ort</th><th>Tel</th><th>Email</th>".
"<th>WWW</th><th>Geburtstag</th></tr>\n";
while ($row=@mysql_fetch_assoc($result)) { // liest Zeilenweise aus solange
                                           // Zeilen vorhanden sind
    echo "<tr>";                             // Zeile erzeugen
    foreach ($row as $key => $value) {       // foreach anfang
    echo "<td>$value </td>\n";
    }                                        // foreach ende
echo "</tr>";                                // Zeile schliessen
}                                            // while Ende
echo "</table>\n";                           // Tabellenende
mysql_close();                               // DB trennen
?>
```

7.7 Zeilenweises Auslesen von bestimmten Daten aus Datentabellen

```php
<?php
function connect_db(){
  global $connect_db;
  if ($connect_db == false){
     include('../../db/connect.inc.php');
     hellodb();
     $connect_db=true;
  }
}
connect_db();
$sql = "SELECT l_update, n_text, n_index FROM sha09_news ".
"ORDER BY n_index DESC";                  // definiert Abfrage
$result = mysql_query($sql); // eruiert vollst. Datentabelleninhalt
while($row = mysql_fetch_array($result)) {
    $datum = $row['l_update'];
    $message = $row['n_text'];
```

```
        echo "<tr><td align='left' valign='top' width='10'></td>".
                  "<td align='left' valign='top'><span ".
                  "class ='text_klein12b'>$datum ".
                  "</span></td><td align='left' valign='top' ".
                  "width='10'></td>".
                  "<td align='left' valign='top'><span ".
                  "class='text_klein12b'>$message ".
                  "</span></td><td align='left' valign='top' ".
                  "width='10'></td></tr>";
    }
        echo "</tr></table>\n";              // schliesst Tabelle
        mysql_close();                       // schliesst Datenbank
    ?>
```

7.8 Auslesen von CSV-Dateien (Seite 35 / Kapitel 1):

```
    <?php
    $fp = fopen("inhalt.csv", "r");
    while ($line = fgetcsv($fp, 10000, ",", "\"")) {
      echo "<h1>$line[0]</h1>
    <p><strong>$line[1]</strong></p>
    <div>$line[2]</div>";
    }
      fclose($fp); // Datei wieder schließen
?>
```

8. Bild Funktionen:

8.1 Imagecreatefromjpeg()

Mit der Funktion **imagecreatefromjpeg()** erstellt man ein neues Bild, das aus einer Datei oder URL im JPEG-Format gelesen wird. Der Inhalt der gelesenen Datei wird in das neue Bild geschrieben. Der von dieser Funktion zurückgegebene Zeiger muss bei allen folgenden Grafikbefehlen genutzt werden, damit man etwas in das Bild hineinzeichnen kann.

imagecreatefromjpeg (string $filename)

$image = imagecreatefromjpeg ($tempname); // Zeiger für Bild

8.2 imagecreatetruecolor()

Mit der Funktion **imagecreatetruecolor()** erzeugt Platzhalterbild. Die Funktion gibt einen Bild-Identifier zurück , der den schwarzen Bereich der gewünschten Grösse repräsentiert in den später das Bild hineingezeichnet wird.

imagecreatetruecolor (int. width, int. heigth)

$image_p = imagecreatetruecolor ($width, $heigth); // Zeiger für Platzhalterbild

Bem: Funktioniert nicht für GIF-Dateien!

8.3 imagecopyresampled()

Mit der Funktion **imagecopyresampled()** wird das Ursprungsbild bzw. veränderte Bild in das Platzhalterbild hineinkopiert. Mit dst_X und dst_Y werden die X- und Y-Koordinaten des Platzhalterbildes angegeben und mit src_X und src_Y werden die X- und Y-Koordinaten des zu kopierenden Teils angegeben.

imagecopyresampled (dst-im, src-im, dst-X, dst-Y, src-X, src-Y, dst-width, dst-heigth, src-with, src-heigth)

imagecopyresampled ($image-p, $image, 0, 0, 0, 0, $width, $heigth, $width_o, $heigth_o)

Bem: Funktioniert nicht für GIF-Dateien!

8.4 imagecopyresized()

Mit der Funktion **imagecopyresized()** kopiert man einen Teil eines Bildes (src_im) in ein anderes Bild (dst_im). Mit src_X und src_Y werden die X- und Y-Koordinaten des zu kopierenden Teils angegeben. Die Breite wird dabei mit src_W und src_H bestimmt.
Der Ausschnitt wird im Bild dst_im an die X- und Y-Koordinaten dst_X bzw. dst_Y kopiert und dabei auf die Breite dst_W und die Höhe dst_H gebracht. Unterscheiden sich die Breite dst_W oder die Höhe dst_H von den Abmessungen des ursprünglichen Ausschnitts, so wird der Ausschnitt gedehnt oder geschrumpft.

imagecopyresized (dst-im, src-im, dst-X, dst-Y, src-X, src-Y, dst-W, dst-H, src-W, src-H)

imagecopyresised ($image-p, $image, 0, 0, 0, 0, $width, $heigth, $width_n, $heigth_n)

Bem: Funktioniert nicht für GIF-Dateien!

8.5 imagejpeg()

Mit der Funktion **imagejpeg()** wird ein zuvor mit imagecreate() erzeugtes Bild (im) im JPEG-Format an den Browser gesendet. Der zweite Parameter (filename) ist optional; wenn Sie ihn angeben, wird das zuvor erstellte Bild in eine Datei geschrieben: imagejpeg($image,"image.jpg"). Möchten Sie einen Wert für den dritten Parameter (quality) übergeben, obwohl Sie das Bild nicht in eine Datei schreiben wollen, so können Sie als zweites Argument einen Leerstring übergeben.
Der Parameter quality legt den Komprimierungsgrad und damit die Qualität des Bildes fest. Je höher dieser Wert, desto besser wird das Bild (10 = starke Komprimierung, 100 = geringe Komprimierung).

imagejpeg(Quellbild, Filename, Qualiät **)**;

imagejpeg($image_p, $filepath, $quality); // Ausgabe als Datei

Der Wert von $image_p (Platzhalter) kann durch Aufruf der Funktion ImageCreate($tempname) eruiert werden.

8.6 imagedestroy()

Mit der Funktion **imagedestroy()** löscht man den Speicher, welcher durch das Bild $image_p belegt wurde.

imagedestroy(Quellbildresource)

imagedestroy($image_p)

8.7 getimagesize()

Mit der Funktion **getimagesize()** kann man verschiedene Informationen über ein Bild (filename) ermitteln. Das Ergebnis wird in einem Array zurückgegeben, das folgende Informationen enthält:

- Breite des Bildes ->[0]
- Höhe des Bildes ->[1]
- Grafik-Typ - 1 = GIF, 2 = JPG, 3 = PNG, 4 = SWF ->[2]
- HTML-Zeichenkette - "height=xx width=xx" ->[3]

Wenn Sie im optionalen Parameter imageinfo ein Array übergeben, trägt die Funktion bei einigen Grafiktypen zusätzliche Daten in dieses Array ein, z.B. Dateiinformationen bei JPEG-Bildern.

getimagesize(String Filename [, array imageinfo])

```
$info = getimagesize( $tempname]);
echo “Bildbreite: “.” $info[0]<br>”;
echo “Grafik-Typ: “.” $info[2]<br>”;
```

9. File Funktionen (siehe dazu auch Anhang 5):

9.1 move_uploaded_file()

Mit der Funktion **move_uploaded_file()** kann man eine mittels HTTP-Post hochgeladene Datei (filename) an ein Ziel (destination) verschieben. Zuerst prüft die Funktion, ob die Datei filename hochgeladen wurde und somit eine gültige Datei ist. Ist dies der Fall, so wird die Datei an das Ziel destination verschoben. Es ist darauf zu achten, dass Sie für filename den temporären Namen der Datei angeben (im Beispiel file) und nicht den tatsächlichen Namen (im Beispiel file_name), da sonst false zurückgegeben wird.

Im Erfolgsfall gibt diese Funktion true zurück. Sollte es sich bei der hochgeladenen Datei um keine gültige Datei handeln oder konnte sie nicht verschoben werden, so wird false zurückgegeben.

move_uploaded_file (string $filename, string $destination)

```
if ( move_uploaded_file ( $_FILES['file']['tmp_name'], 'tmp/test.txt' ) ) {
   echo '<b>Upload beendet!</b>';
   }
```

9.2 filesize (string $filename)

Mit **filesize()** kann man sich die Größe einer Datei (filename) in Byte zurückgeben lassen. Als Rückgabewert dieser Funktion erhalten Sie die Größe der Datei, im Fehlerfall wird false zurückgeliefert.

Unbedingt beachten dass das Ergebnis zwischengespeichert (siehe clearstatcache) wird.

```
$datei='index.php';
$id=filesize($datei);
echo 'Größe der Datei '.$datei.':'.$id;
```

9.3 strpos(string,gesuchtesWort)

```
If (strpos(string,gesuchtesWort)===false {
echo “Gesuchtes Wort ist nicht vorhanden“;
}
```

9.4 substr_replace(string,‘Ersatzzeichen‘,Beginnposition,Endposition)

```
$newstring = substr_replace(123456789,“ABC“,0,2);
echo „umgerechneter String lautet $newstring“;
 => Ausgabe in diesem Fall: ABC456789
```

10. Mail Funktionen:

Die Mail-Basisfunktion sieht wie folgt aus:

> **mail(**string to, string subject, string message[, string additional_headers [, string additional parameters]]**)**

10.1 Einfaches Mail

Einfache Mails können wie folgt versandt werden:

> **mail(**string to, string subject, string message) oder im Klartext
>
> **mail(**$empfaenger, $betreff, $mailbody**)**

10.2 Mail an mehrere Empfänger

Die erste Mailadresse ist im erste Mailargument ($empfaenger), weitere werden im vierten Mailargument in den additional headers ($headers) eingegeben:

> **mail(**string to, string subject, string message, string additional_headers) oder im Klartext
>
> **mail(**$empfaenger, $betreff, $mailbody, $headers**)**

Der optionale String für Headers kann wie folgt aussehen:

```
$headers = “From: abc@wx.yz\r\n“;
$headers .= “To: abc@wx.yz, abc@wx.yz, abc@wx.yz\r\n“;
$headers .= “Cc: abc@wx.yz, abc@wx.yz, abc@wx.yz\r\n“;
$headers .= “Bcc: abc@wx.yz, abc@wx.yz, abc@wx.yz\r\n“;
```

Mehrere Header-Zeilen werden durch \r\n voneinander getrennt. Die letzte darf aber diese Trennzeichen nicht mehr enthalten, da sie hier automatisch beigefügt werden.

10.3 Mail mit Anhang

Mails mit Anhang werden im additional header string nach den Adresszusätzen versandt.

mail(string to, string subject, string message, string additional_headers) oder im Klartext

mail($empfaenger, $betreff, $mailbody, $headers**)**

Es werden folgende optionale Strings für Header verwendet:

$headers .= “MIME-Version: 1.0\r\n“;
$headers .= “Content-type: text/plain; charset=iso-8859-1\r\n”;
$headers .= “Content-type: text/html; charset=iso-8859-1\r\n”;

Mehrere Header-Zeilen werden durch \r\n voneinander getrennt. Die letzte darf aber diese Trennzeichen nicht mehr enthalten, da sie hier automatisch beigefügt werden.

11. Formular-Funktionen:

Wenn Ziel des Forms in der selben Datei liegt wird action=“ “ eingegeben. Dadurch kann vermieden werden, dass Dateien mit inc.php – Abschluss gesperrt werden.

11.1 Formularabfragen mit Passwort:

```
<form action="XXX.php"  name="abfrage_0" method="POST">
<input type="text" name="doc" size="10" value ="" maxlength="4">
<input type="password" name="pw" value="" maxlength="25">
<input type="submit" name="abfrage_0" value="senden"></td>
</form>
```

Bem: Länge des Eingabefeldes ist in size=““ zu finden und Länge der Eingabe in maxlength=““!

11.2 Formularabfragen mit versteckter Wertweitergabe ($doc = Wert):

```
<form action="XXX.php"  name="abfrage_1" method="POST">
<input type="hidden" name="doc" value ="<?php echo $doc;?>" maxlength="25">
<input type="submit" name="abfrage_1" value="senden"></td>
</form>
```

Bem: Wert wird in value=““ weitergegeben!

11.3 Formularfelder mit bleibender Anzeige der Eingabe:

```
$doc=isset($_POST['doc'])? htmlspecialchars(stripslashes($_POST['doc'])):"";

<form action="XXX.php"  name="abfrage_2" method="POST">
<input type="text" name="doc" Value ="{$doc}" maxlength="4">
<input type="submit"  name=" abfrage_2" value="löschen"></td>
</form>
```

Bem: Am Anfang wird der Leerwert festgelegt bzw. werden zur Sicherheit mögliche HTML-Tags eliminiert. Anschliessend wird der Wert unter value=““ hineingeschrieben

11.4 Formularauswertungen & Eingabekontrollen:

11.4.1 Eingabekontrolle (leer wenn nicht gesetzt bzw. ohne Leerschläge und Verhinderung von html-tags):

```
$doc=isset($_POST['doc'])? htmlspecialchars(stripslashes($_POST['doc'])):"";
```

11.4.2 Zeichenauswechslung (ersetzt im String alle im Aray festgelgten Werte):

```
$string = ' . . . . . . '
$array = array( 'ä' => 'ae', 'ü' => 'ue', 'ß' => 'ss')
$neuerstring = strtr ( $string, $array );
```

11.4.3 Umwandlung Grossbuchstaben in Kleinbuchstaben:

```
$name = 'xyz';
$array=('ä'=>'ae','ö'=>'oe');
$neuername = strtr($name,$array)
```

11.4.4 Überprüfen auf Mindesteingabe:

```
if ((empty($_POST['FeldA'])) OR (empty($_POST['FeldB'])) OR
(empty($_POST['FeldC'])) OR (empty($_POST['FeldD']))) {
echo "<p>Eingabe nicht vollständig!Bitte mit Retourknopf zurück (<=) und
Eingabe vervollständigen</p>";
}
```

11.4.5 Überprüfen der Formulareingaben mit Schablonen:

```
$inh_p = $_POST['inh'];                        // prov. Inhalt
$muster_inh = "/^[A-Z]{1,1}$/";                // Prüfschablone
if (preg_match($muster_inh, $inh_p) == 0) {  //Prüfanweisung
    $inh = "";          // leer wenn Inhalt nicht akzeptiert
    }
    else{
    $inh = $inh_p;  // akzeptierter Inhalt wird weitergegeben
    }
Prüfschablonen-Vorschläge:
Textinhalt (3-25 Ziffern): "/^[A-Za-z0-9-_.*]{3,25}$/"
-> Umlaute bieten Probleme! Hier besser invertiert prüfen mit ^[^0-9]
Artikel-Nummer: "/^[A-Za-z0-9-_.]{2,8}$/"
Postleitzahl: "/^[A-Z0-9-]{4,8}$/"
Mailadressen: "/^[a-zA-Z0-9-_.]+@[a-zA-Z0-9-_.]+\.[a-zA-Z]{2,4}$/"
Reine Längenüberprüfung von Text: "/^[^$!]{3,25}$/"
```

11.4.6 Überprüfen der Mail-Formulareingabe mit Schablone:

```
$mail_p = $_POST['mail'];                       // prov. Mailadresse
$muster_mail = "/^[a-zA-Z0-9-_.]+@[a-zA-Z0-9-_.]+\.[a-zA-Z]{2,4}$/";
if (preg_match($muster_mail, $mail_p) == 0) {   //Prüfanweisung
    $mail = "";                 // leer wenn Inhalt nicht akzeptiert
    }
    else{
    $mail = $mail_p;     // akzeptierter Inhalt wird weitergegeben
    }
```

11.4.7 Entfernen von html-Tags aus Formulareingaben:

```
(Bem. evtl. nach erfolgter Prüfung 11.4.6 nicht mehr nötig)

$inhalt = isset($_POST['inh']) ? htmlspecialchars(trim($_POST['inh'])):"";
```

12. Sessionen:

Bei Sessionen wird im Server ein Sessionseintrag gemacht. Dieser Eintrag kann auf nachfolgenden PHP-Seiten abgefragt oder ergänzt werden. Der Sessionsinhalt wird normalerweise automatisch per Cookie (sog. temporäre Sitzungscookies) oder - wenn Cookies nicht gehen - per URL-Anhang auf die nächste PHP-Seite weiter-gegeben. Sessionen und deren Cookies müssen am Schluss einer Sitzung gelöscht werden!

12.1 DB-Initialisierung

Aufruf der DB-Einstellungen in der Datei „php.ini“ in XAMP/PHP mittels der Funktion phpinfo(). Dieser Aufruf kann mit der im Verzeichnis zu Kurs 1 und Kurs 2 abgelegten Seite phpinfo.php abgerufen werden und erlaubt die Veränderung der entsprechenden Einstellungen. Einstellungen bezüglich Sessionen unter dem Abschnitt „Session“ gemäss Seite Pr 68.
Speziell: Letzter Punkt bez. „Session.use_trans_sid = On“ kann bei meinem lokalen Server nicht wie gewünscht auf „On“ gestellt werden! Daher wird bei fehlendem Cookie kein URL-Anhang mit den Cookiedaten weitergegeben!

12.2 Sessionen starten / öffnen

Start der Session mit session_start(); **auf jeder Seite** - die in die Sessionsbehandlung einbezogen werden soll - **vor Beginn des html-Headers** (Grund: Weil cookies aufgerufen werden!).
Diese Funktion legt ein Cookie namens PHPSESSID an oder hängt bei im Browser ausgeschalteter Cookiebehandlung ein PHPSESSID=abc123 an die URL an (Beachte 8.1 -> Spezial).
Die Verwendung des folgenden Moduls **sessionsheader.inc.php** löst im CMBasic die Einrichtprobleme auf dem Server auf elegante Weise:

```
<?php
ini_set(„session.use_cookies“, 1);
ini_set(„session.use_only_cookies“,  1);
ini_set(„session.use_trans_sid“, 0);
session_start()
?>
```

12.3 Beifügen eines Sessions-Namens

Diese Funktion geht nur für die jeweilige Seite und ist nicht übertragbar auf weitere Seiten!

```
<?php
session_name('Bezeichnung');
session_start();
```

12.4 Ansehen der temporären Session im Server

Alle Sessionen und ihr Inhalt können jederzeit unter C:\xampp\tmp angeschaut werden.

12.5 Beenden einer Session

Sessionen können auf folgende Art sauber gelöscht werden:

```
<?php
session_start();                    // Session starten im Server
session_unset();                    // löschen der Variablen der Session
if (isset ($_COOKIE['PHPSESSID'])) {        // löschen des evtl. vorhandenen
setcookie("PHPSESSID","", time() -86400);     COOKIE im Browser
}
Session_destroy();                  // löschen der Session im Server
?>
```

12.6 Überprüfen der Sessions-ID

A) wenn im Browser COOKIES erlaubt sind.

1. PHP-Dokument in Browser aufrufen
2. Adressfeld mit „**javascript:alert(document.cookie)**“ versehen und „Enter“ drücken
3. Es erscheint eine Alert-Box mit PHPSESSID = abc123.

B) wenn im Browser COOKIES ausgeschaltet sind bzw. bei erstmaligem Aufruf der Seite

1. Quelltext öffnen
2. Im Formularwurde ein verstecktes Feld mit dem Namen PHPSESSID eingefügt.

12.7 Eintragen von Daten unter einer Sessions-ID

Datenweitergabe mittels Sessionen funktioniert nur zwischen PHP-Seiten! Die Session ist ein Array und wird auch dementsprechend gefüllt!
Bei Aufruf einer Session wird der Inhalt einer Session nicht verändert. Er kann aber mit folgenden Schritten gezielt ergänzt werden:

```
<?php
session_start();                                    // Session starten oder aufrufen
$_SESSION['Name'] = $_POST['Name'];                 // Array "Name" wird hier mit der
                                                       Formulareingabe gefüllt
$_SESSION['Seite'] = basename($_SERVER['PHP_SELF']);        // Array „Seite" wird mit
                                                              Seitenbezeichnung
                                                              gefüllt:
$_SESSION['Kriterium'] = $Kriterium;            // Array „Kriterium" wird mit Daten gefüllt
```

12.8 Abfragen der unter einer Sessions-ID abgelegten Daten

Das Auslesen des Session erfolgt durch Angabe der Array-Felder wie z.B. [‚Name']:

```
$name = $_SESSION['Name']
$page = $_SESSION['Seite']
$Krit = $_SESSION['Kriterium']
```

12.9 IP-Adressenvergleich

Mit dieser Funktion lässt sich im Server die IP-Adresse ermitteln und in einem Sessions-Array [ipadr] ablegen

```
$_SESSION['ipadr'] = $_SERVER['REMOTE_ADDR'];
```

Das fertige Script sieht wie folgt aus:

```
<?php
session_start();
if (!isset($_SESSION['ipadr']))  {                          // wenn IP nicht eingetragen
    $_SESSION['ipadr'] = $_SERVER['REMOTE_ADDR'];           // erfolgt Eintrag
}
else if ($_SESSION['ipadr'] != $_SERVER['REMOTE_ADDR'])  {          // wenn IP nicht korrekt
    session_unset();                                        // Sessionsinhalt wird gelöscht
    if (isset($_COOKIE['session_name()']))  {               // auch Cookie wird gelöscht
        setcookie(session_name(), "", time() -86400);
    }
    session_destroy();                                      // Session wird gelöscht
}
?>
```

12.10 Verwenden von Sessionen als Zugangskriterium

Mit dieser Funktion lassen sich Seiten für Unberechtigte unsichtbar machen. Der zu schützende Teil kann nur eingesehen oder bearbeitet werden, wenn die Überprüfung der Session zu einem positiven Resultat führt:

```
<?PHP
Include_once = "sessionheader.inc.php";
if (isset($_SESSION['right'])  &&  $_SESSION['right'] == 4) {
-> zu versteckender Scriptteil
}
?>
```

13. COOKIES:

Cookies werden vom PHP-Server zum ansteuernden Browser gesendet wenn eine Session eröffnet wird. Sie dienen zur Uebermittlung der vollständigen Session, können aber auch nur zur Uebermittlung eines Kriteriums verwendet werden. Cookies sollten am Schluss einer Sitzung immer gelöscht werden, sonst leben sie weiter!
Browser können für Cookies gesperrt sein. In diesem Fall kann das Kriterium oder die Session per URL-Anhang weitergegeben werden, wenn der Server das erlaubt und dazu eingerichtet ist. Ist das auch nicht möglich so können Cookie-Anwendungen von diesen Browser nicht verwendet werden!

13.1 Cookie setzen bzw. senden

```
<?PHP
setcookie(„Name“, ““ , time();                    // immer am Anfang einer Seite!!
?>
```

13.2 Cookie abfragen

In gewissen Fällen muss das Cookie überprüft und abgefragt werden können. Es kann nach Vorhandensein und Name bzw. ID abgefragt werden:

```
<?PHP
If isset($COOKIE[‚name’]) {          // wird TRUE bei vorhandenem Cookie
$cookie = $_COOKIE['name']);        // Abfrage des Name bzw. der ID
echo “ Der Names des Cookies lautet $cookie”;
}
?>
```

13.3 Cookie löschen

Cookies können nicht gelöscht werden; sie müssen mit einer bereits abgelaufenen Zeit überschrieben werden. Die dazu verwendete Zeitangabe von -86400 deutet auf eine Ablaufzeit von gestern. Dadurch wird das Cookie nicht mehr lebensfähig und gelöscht.

```
<?PHP
Setcookie(„Name“, ““ , time() -86400;
?>
```

13.4 Verwenden von Cookies als Zugangskriterium

Mit diesem Script lassen sich Seiten für Unberechtigte unsichtbar machen. Der zu schützende Teil kann nur eingesehen oder bearbeitet werden wenn die Überprüfung des Cookie zu einem positiven Resultat führt:

```
<?PHP
if (isset($_COOKIE['name']) && $_COOKIE['name'] == "a1234x7") {
-> zu versteckender Scriptteil
}
?>
```

14. Formulare:

Beginn aller Formulare generell mittels <form "name" action="auswertung.php"> und Abschluss mittels </form>. Fehlt der Name des Formulars und ist unter action keine Zielseite (nur action=""), so geht der POST-Versand auf den Anfang der vorliegenden Seite.

Verwendete Formular-Tags:

Auswahlfeld 1):	<input type="radio" name="x" value="">
Text-Eingabefeld 2):	<input type="text" name="x" value="">
Verborgenes Feld 2):	<input type="hidden" name="y" value="">
Meldungsfeld:	<textarea name ="z" cols="m" rows="n">
Sendefeld:	<input type="submit" value="Absenden">
Auswahl- & Dropdownliste	<select name="xyz" size="1"> <option>AA</> <option>BB</> <option>CC</> </select>

1) Die Auswahl erfolgt mit runden oder viereckigen Checkboxes, welche durch den type unterschieden werden.
2) Dieser Eintrag erlaubt das Eingeben von Zahlen und Buchstaben. Die Länge des Eingabefeldes lässt sich einstellen. Die eingetragene Information wird im Feld "value" übertragen.

Anhang 4: Was ist UTF-8 und warum soll man es verwenden?

Unicode ist die offizielle Implementierung für ISO 10646, ein universeller Standard zur Kodierung von Texten in verschiedenen Sprachen. UTF-8 steht für Unicode Transformation Format - 8 und ist ein Industriestandard für die Darstellung von Zeichen mit einer variablen Länge zwischen 1 und 4 Oktetten (Byte). UTF-8 wurde eingeführt, um einen Nachteil der Unicode-Kodierungen UCS-2 und UCS-4 (Universal Multiple-Coded Character Set) mit je 16 (65.536 verschiedene Zeichen) und 32 Bit (rechnen Sie selbst) codierten Zeichen auszuräumen. Durch die dort vorgeschriebene feste Bitbreite müssen für die Darstellung vieler in unserem Kulturkreis oft gebrauchter Glyphen viele führende Nullbits eingeführt werden. Daher sind mit diesen Zeichensätzen kodierte Dokumente oft ungleich größer als mit dem gebräuchlichen 8-Bit Zeichensatz ISO-8859-1 kodierte Dokumente UTF-8 hingegen ist hingegen in der Lage, ASCII-Zeichen, die den Großteil der von uns gebrauchten Zeichen ausmachen, mit 8 Bit darzustellen - vollkommen analog zu ISO-8859-1 oder Latin-1, wie der Zeichensatz auch oft genannt wird. UTF-8 kodierte Dokumente unterscheiden sich in der Größe daher kaum von Latin-1 kodierten Dokumenten. Einziger Unterschied - UTF-8 kodiert die deutschen Umlaute mit zwei Oktetten statt nur mit einem, woher vermutlich auch das Ammenmärchen rührt, mit UTF-8 könne man keine Umlaute darstellen.

Der Zeichensatz hat daran keine Schuld. Im Gegenteil - er kann deutlich mehr als nur unsere Umlaute darstellen. Eine Tabelle von Sprachen und Glyphen mit Angaben zur Unterstützung durch Unicode finden Sie auf der Unicode Webseite. Sehen Sie sich diese Webseite ausführlich an. Unicode schenkt damit einer Vielzahl von Menschen die Freiheit, ihre Glyphen in Computersystemen zu verwenden - und zwar interoperabel. Mit UTF-8 ist die Notwendigkeit, den Zeichensatz umzustellen, nur weil Ihnen ein Freund oder Geschäftspartner aus Griechenland eine E-Mail schreibt, Vergangenheit. Der brasilianische oder tschechische Bekannte kann Ihnen Details zu seiner Sprache erklären, ohne daß Sie erraten müssen, von welcher Glyphe er gerade sprechen mag. Zu guter Letzt können auch Sie ihrem amerikanischen Freund "these funny looking characters", nämlich "the umlauts", per Computer ohne weiteren Aufwand erklären und näherbringen.

Ein weiterer wichtiger Punkt ist die Tatsache, daß der Latin-1 Zeichensatz kein Euro-Symbol darstellen kann. Aus diesem Grund wurde ISO-8859-15 eingeführt, das ein um genau ein Byte (das Eurosymbol) erweitertes Latin-1 darstellt. Latin-1 sollten Sie daher heutzutage nur noch sehr eingeschränkt verwenden, stattdessen sollten Sie auf ISO-8859-15 oder gleich das wesentlich flexiblere UTF-8 umsteigen.

Nicht nur die Tatsache, daß mittlerweile die meiste Software inkl. Bibliotheken auf Unicode umgestellt sind, sondern auch die zunehmende Internationalisierung (i18n) sprechen eigentlich eine deutliche Sprache.

Abschließend seien noch einmal die wichtigsten Punkte wiederholt:

- Ein Dokument aus unserem Kulturkreis ist mit UTF-8 ungefähr gleich groß wie mit ISO-8859-1.
- ISO-8859-1 ist nicht in der Lage, das Euro-Zeichen darzustellen - der Zeichenvorrat ist erschöpft.
- ASCII ist eine Teilmenge von UTF-8

Anhang 5: Weitere File-Funktionen mit Beispielen

fopen

Befehl

resource **fopen** (string $filename, string $mode [, int $use_include_path [, resource $zcontext]])

Version (PHP 4, PHP 5)

Beschreibung

Mit fopen() kann man eine Datei (filename) öffnen.

Folgende Verbindungsmöglichkeiten gibt es:
• "http://" - Öffnen per http
• "ftp://" - Öffnen per ftp
• "php://stdin" - Öffnen per stdio stream
• "php://stdout" - Öffnen per stdio stream
• "php://stderr" - Öffnen per stdio stream
• Alles andere - Öffnen vom lokalen Dateisystem

Der Parameter mode legt fest, auf welche Weise und für welche Zugriffsarten die Datei geöffnet wird.
Folgende Werte gibt es:

- a - Öffnet die angegebene Datei nur zum Schreiben und positioniert den Dateizeiger auf das Ende der Datei. Sollte die angegebene Datei nicht existieren, so wird versucht sie anzulegen.
- a+ - Öffnet die angegebene Datei zum Lesen und Schreiben und positioniert den Dateizeiger auf das Ende der Datei. Sollte die angegebene Datei nicht existieren, so wird versucht sie anzulegen.
- r - Öffnet die angegebene Datei zum Lesen und positioniert den Dateizeiger auf den Anfang der Datei.
- r+ - Öffnet die angegebene Datei zum Lesen und Schreiben und positioniert den Dateizeiger auf den Anfang der Datei.
- w - Öffnet die angegebene Datei zum Schreiben und positioniert den Dateizeiger auf den Anfang der Datei. Die Länge der Datei wird auf 0 Byte gesetzt. Sollte die angegebene Datei nicht existieren, so wird versucht sie anzulegen.
- w+ - Öffnet die angegebene Datei zum Lesen und Schreiben und positioniert den Dateizeiger auf den Anfang der Datei. Sollte die angegebene Datei nicht existieren, so wird versucht sie anzulegen.

Für die Bearbeitung von Binärdateien können Sie an jede dieser Modusdefinitionen ein "b" anhängen. Wird der optionale Parameter use_include_path auf 1 gesetzt, so wird auch innerhalb des Include-Pfads (wird in der php.ini bestimmt) nach der Datei gesucht.
Im Erfolgsfall gibt die Funktion einen Dateizeiger zurück. Sollte das Öffnen der Datei scheitern, so wird false zurückgeliefert.

Beispiel

```
<?PHP
$fp = fopen('/www/users/php/counter.txt','r'); //Absoluter Pfad
$fp = fopen('counter.txt','r');// Relativ Pfad
$fp = fopen('/www/users/php/selfphp.gif','wb');
$fp = fopen('http://www.selfphp3.de/','r');
$fp = fopen('ftp://user:password@selfphp3.de/','r');
$fp = fopen('d:/daten/statistik.csv','a');
?>
```

Ausgabe `// Keine Ausgabe // Es werden die Möglichkeiten vorgestellt`

fgets

Befehl

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

Version (PHP 4, PHP 5)

Beschreibung

Mit fgets() kann man aus einer Datei (fp) eine Zeile mit der Länge length (in Byte von vorne) auslesen. Sollte die Zeile länger sein als der in length vorgegebene Wert, so wird die Zeile bis zur angegebenen Länge gelesen und der Rest abgeschnitten.Kommt es beim Lesen der Datei zu einem Fehler, so wird false zurückgeliefert.

Beachten Sie bitte, dass es sich bei dem Dateizeiger fp um einen gültigen Zeiger auf eine offene Datei handeln muss, der mit fopen(), fsockopen() oder popen() erzeugt wurde.

Beispiel 1

```
<?PHP  // Inhalt der Datei counter.txt: 52369
$zaehler_anmelden = 'counter.txt';
$fp = fopen($zaehler_anmelden,'r');
$zahl = fgets($fp,10);
fclose($fp);
$zahl++;
echo 'Counterstand:<br>';
echo $zahl;
?>
```

Ausgabe 1

```
Counterstand: 52370
```

Beispiel 2

```php
<?PHP // Inhalt der Datei counter.txt: 52369
$zaehler_anmelden = 'counter.txt';
$fp = fopen($zaehler_anmelden,'r');
$zahl = fgets($fp,2);
fclose($fp);
$zahl++;
echo 'Counterstand:<br>';
echo $zahl;
?>
```

Ausgabe 2

```
Counterstand: 53
```

fread

Befehl

string **fread** (resource $handle, int $length)

Version (PHP 4, PHP 5)

Beschreibung

Mit fread() kann man Binärdaten aus einer Datei (fp) lesen. Der zweite Parameter für die Länge (length) bestimmt, wie viel der Datei gelesen werden soll (von vorne max. bis zum Dateiende).

Beispiel 1

```php
<?PHP
//Inhalt der Datei statistik.csv:
/*01.08.2001#62.159.232.250#www.selfphp3.de
02.08.2001#212.82.34.222#www.selfphp4.de
02.08.2001#http://suchen.abacho.de#62.159.232.250
03.08.2001#http://www.fireball.de#212.185.44.15*/

$zaehler_anmelden  'statistik.csv';
$fp = fopen($zaehler_anmelden, 'r');
$str = fread($fp,filesize($zaehler_anmelden));
fclose($fp);
echo $str;
?>
```

Ausgabe 1

```
01.08.2001#62.159.232.250#www.selfphp3.de_
02.08.2001#212.82.34.222#www.selfphp4.de_
02.08.2001#http://suchen.abacho.de#62.159.232.250_
03.08.2001#http://www.fireball.de#212.185.44.15
```

Beispiel 2

```
<?PHP
//Inhalt der Datei statistik.csv:
/*01.08.2001#62.159.232.250#www.selfphp3.de
02.08.2001#212.82.34.222#www.selfphp4.de
02.08.2001#http://suchen.abacho.de#62.159.232.250
03.08.2001#http://www.fireball.de#212.185.44.15*/

$zaehler_anmelden = 'statistik.csv';
$fp = fopen($zaehler_anmelden, 'r');
$str = fread($fp,10);
fclose($fp);
echo $str;
?>
```

Ausgabe 2 `01.08.2001`

fputs

Befehl

int **fputs** (resource $handle, string $str [, int $length])

Version (PHP 4, PHP 5)

Beschreibung

Mit fputs() kann man bestimmte Daten (str) an die aktuelle Position des Dateizeigers in eine Datei (fp) schreiben. Wird der dritte optionale Wert für die Länge (length) des Strings nicht angegeben, so wird der gesamte String an die Position des Dateizeigers geschrieben (siehe Beispiel 1). Wird die Länge angegeben, so wird der String auf die vorgegebene Länge gekürzt (siehe Beispiel 2).

Beispiel 1

```
<?PHP //Inhalt der Datei counter.txt: 52386
$zaehler_anmelden = 'counter.txt';
$fp = fopen($zaehler_anmelden,'r');
$zahl = fgets($fp,10);
fclose($fp);
```

```php
$zahl++;
$fp = fopen($zaehler_anmelden,'w');
flock($fp,2);
fputs($fp,$zahl);
flock($fp,3);
fclose($fp);
echo $zahl;
?>
```

Ausgabe 1 `Counterstand: 52387`

Beispiel 2

```php
<?PHP //Inhalt der Datei counter.txt: 52387
$zaehler_anmelden = 'counter.txt';
$fp = fopen($zaehler_anmelden,'r');
$zahl = fgets($fp,3);
fclose($fp);
$zahl++;
$fp = fopen($zaehler_anmelden,'w');
flock($fp,2);
fputs($fp,$zahl,3);
flock($fp,3);
fclose($fp);
echo $zahl;
?>
```

Ausgabe 2 `Counterstand: 524`

flock

Befehl

bool **flock** (resource $handle, int $operation [, int $&wouldblock])

Version (PHP 4, PHP 5)

Beschreibung

Mit flock() kann man eine Datei (fp) für bestimmte Zugriffe (operation) verriegeln. Dies ist oftmals sehr nützlich, um Dateien, die Sie gerade beschreiben wollen, vor dem Zugriff von anderen Usern zu schützen. Beachten Sie bitte, dass es sich bei dem Dateizeiger fp um einen gültigen Zeiger auf eine offene Datei handeln muss, der mit fopen(), fsockopen() oder popen() erzeugt wurde.

Folgende Arten der Verriegelung sind im Parameter operation möglich:
• LOCK_SH (1) - Verriegelung für Lesezugriff
• LOCK_EX (2) - exklusive Verriegelung für Schreibzugriffe
• LOCK_UN (3) - Gibt eine Verriegelung wieder frei

• LOCK_NB (4) - Verhindert, dass die Funktion während der Verriegelung blockiert. Diese Konstante können Sie zusätzlich zu den anderen 3 Konstanten angeben.

Im optionalen Parameter wouldblock gibt die Funktion den Wert true zurück, falls die Funktion während der Verriegelung blockieren müsste. Im Erfolgsfall gibt diese Funktion true zurück, sonst false.

Beispiel

```
<?PHP  //Inhalt der Datei counter.txt: 52369
$zaehler_anmelden ='counter.txt';
$fp = fopen($zaehler_anmelden,'r');
$zahl = fgets($fp,10);
fclose($fp);
$zahl++;
$fp = fopen($zaehler_anmelden,'w');
flock($fp,2);
fputs($fp,$zahl);
flock($fp,3);
fclose($fp);
echo $zahl;
?>
```

Ausgabe `52370`

rewind

Befehl

int **rewind** (resource $handle)

Version (PHP 4, PHP 5)

Beschreibung

Mit rewind() kann man die Position des Dateizeigers auf den Anfang der Datei (fp) setzen. Es ist darauf zu achten, dass es sich bei fp um einen gültigen Dateizeiger handeln muss.
Im Erfolgsfall gibt die Funktion true zurück, sonst false.

Beispiel

```
<?PHP
$fp = fopen('members.txt','r');
for($x = 1; $x < 4; $x++)
{
echo $x.':';
```

```
echo fgets($fp,50);
echo '<br>';
}
for($x = 4; $x < 7; $x++)
{
rewind($fp);
echo $x.':';
echo fgets($fp,50);
echo '<br>';
}
fclose($fp);
?>
```

Ausgabe

```
1: Dieser Text wird gespeichert
2:
3:
4: Dieser Text wird gespeichert
5: Dieser Text wird gespeichert
6: Dieser Text wird gespeichert
```

touch

Befehl

int **touch** (string $filename [, int $time [, int $atime]])

Version (PHP 4, PHP 5)

Beschreibung

Mit touch() kann man versuchen, Datum und Uhrzeit der letzten Änderung einer Datei (filename) zu ändern. Wird der optionale Zeit-Parameter (time) im UNIX-Timestamp-Format (Beginn der UNIX-Epoche - 01.01.1970 , 0:00:00 GMT) angegeben, so wird dieses Datum für die Änderung genutzt. Sollten Sie keinen Wert für die Zeit (time) angeben, so wird mit der Funktion time() die aktuelle Zeit ermittelt und für die Änderung genommen. Falls die von Ihnen angegebene Datei nicht existieren sollte, so wird sie angelegt und mit dem Änderungsdatum versehen. Im Erfolgsfall gibt diese Funktion true, sonst false zurück.

Beispiel 1

```
<?PHP
$date = time();
$datei = 'zukunft.php';
if(touch($datei))
{
echo'Datum und Uhrzeit auf '.strftime('%e.%m.%Y-
```

```
%R',$date).'geändert';
}
else
{
echo 'Datum und Uhrzeit konnten nicht geändert werden.';
}
?>
```

Ausgabe 1 `Datum und Uhrzeit auf 7.08.2001 - 16:13 geändert`

Beispiel 2

```
<?PHP
$date = mktime(12,35,32,12,30,2008); //30.12.2008 - 12:35:32 Uhr
$datei = 'zukunft.php';
if(touch($datei,$date))
{
echo 'Datum und Uhrzeit auf ' .strftime('%e.%m.%Y -%R',$date) . '
geändert';
}
else
{
echo 'Datum und Uhrzeit konnten nicht geändert werden.';
}
?>
```

Ausgabe 2 `Datum und Uhrzeit auf 30.12.2008 - 12:35 geändert`

fclose

Befehl

bool **fclose** (resource $handle)

Version (PHP 4, PHP 5)

Beschreibung

Mit fclose() kann man eine offene Datei (fp) schließen. Bei erfolgreichem Schließen der Datei wird true, sonst false zurückgeliefert. Beachten Sie bitte, dass es sich bei dem Dateizeiger fp um einen gültigen Zeiger auf eine offene Datei handeln muss.

Beispiel

```php
<?php
$zaehler_anmelden = 'counter.txt';
$fp = fopen($zaehler_anmelden,'r');
$zahl = fgets($fp,10);
fclose($fp);
?>
```

Ausgabe `Keine Ausgabe`