1. PHP Einführung

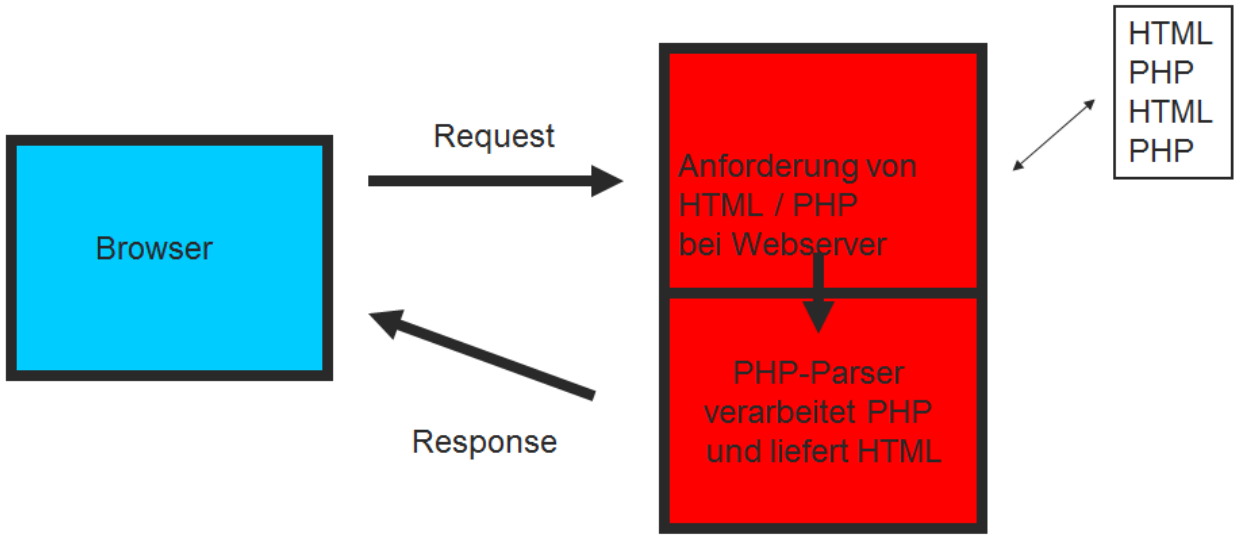
- PHP steht für Hypertext Preprocessor.
  Es ist eine Skriptsprache für Webentwicklung. Das Skript wird von Server interpretiert und abgearbeitet. Ergebnis wird an den Client geschickt und dort interpretiert (IE, FF, …)

  - Entwickelt von: PHP Group, http://www.php.net
    Aktuelle Version: 5.3.0
    Dateiendung: .php
    Grundlegend ist Syntax mit C / C++ vergleichbar

- HTML
  Man benötigt einen simplen Webserver
  z.B.: Apache, IIS, …
  HTML Dateien werden nur übermittelt

- PHP
  Man benötigt einen simplen Webserver und den PHP Interpreter.
  ES gibt eine PHP-Anbindung für alle gängigen Webserver.

Verarbeitung einer PHP-Seite



2. Installation von PHP

Die Installation und Konfiguration der einzelnen Komponenten leistet das Gesamtpacket XAMPP. Es wird von "Apache Friends " bereitgestellt und beinhaltet unter anderem:

- Apache Webserver
- PHP Interpreter
- MySQL Datenbank
- Interface zur Administration dieser Komponenten

Installationsvorschlag:

- Download der neuesten Version: 12/2011: XAMPP 1.7.7 als ZIP
- Entpacken dieses Files direkt in einem Root-Laufwerk z.B.: c:\xampp
- Start von Apache & MySQL mittels \xampp\xampp_start.exe
- **Webverzeichnis: c:\xamp\htdocs**

Anmerkungen zum Webverzeichnis:

- Das Webverzeichnis ist das veröffentlichte Verzeichnis auf das die IP-Adresse oder Domain des Webservers zeigt.
- Darunter sollte sich die notwendige Verzeichnisstruktur befinden
- Dieses Verzeichnis wird lokal mit **localhost** (127.0.0.1) angesprochen.
  z.B:: localhost/htl/tinf2/beispiel1.html => http://localhost/htl/tinf2/beispiel1.html
  (das Protokoll http wird standardmässig hinzugefügt).

Anmerkungen zu PHP-Dateien:

PHP Dateien enden mit ".php" und befinden sich, wie HTML Dateien im **Webverzeichnis des Webservers**. In einer PHP Datei kann sich PHP Code und HTML Code befinden.

PHP Code wird mit "<?php" eingeleitet und mit "?>" abgeschlossen und können beliebig oft in einer PHP Datei vorkommen.

Text außerhalb eine PHP Bereichs wird als HTML interpretiert.

Anmerkungen zuXAMPP:

Das Installationspaket XAMPP beinhaltet folgende Programme:

```
Apache, MySQL, PHP + PEAR, Perl, mod_php, mod_perl, mod_ssl, OpenSSL,
phpMyAdmin, Webalizer, Mercury Mail Transport System for Win32, Ming, FileZilla
FTP Server, mcrypt, eAccelerator, SQLite, WEB-DAV
```

2.1. erste Beispiele:

reine Textausgabe:

```
<?php
        echo "Hello World<br>";
        echo "mein Name ist Susanne Kalliany<br>";
? >
```

Testen der PHP-Installation:

Detaillierte Informationen über die PHP Installation und aller aktivierten Module:

```
<?php
        phpinfo();
? >
```

Einfache PHP-Seite:

```
<!DOCTYPE html>
<html>
<head>
        <title>Grundgerüst einer PHP-Seite</title>
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
</head>
<body>
<p>Hier kommt der Text von HTML</p>
<?php
        echo "Nun kommt der Text von PHP!";
?>
</body>
</html>
```

Einfaches Beispiel eines Formulars:

Startseite: select.html

```
<html>
<form method="post" action="2_if-select-action.php">

<p>Bitte Alter eingeben: &amp;nbsp;&amp;nbsp;
<input type="text" name="uebergabe_alter"></p>

<p>Hobby: &amp;nbsp;&amp;nbsp;
<select name="hobby">
    <option selected>Lesen</option>
    <option>Fernsehen</option>
    <option>Sonstiges</option>
</select>

<p><input type="submit" name="absenden" value="Senden"></p>
</form>
</html>
```

dazugehöriges Action-Script: 2_select_action.php

```
<?
var_dump($_POST);                                   // ACHTUNG: nur für Testzwecke!
echo "<br>";
$alter = $_POST['uebergabe_alter'];                 // von textbox -> Variable
$hobby = $_POST['hobby'];                           // von Listenfeld -> Variable
if ( $alter == null || is_numeric($alter)==false || $alter<=0)
   echo "Geben Sie ein g&amp;uuml;ltiges Alter ein";

else
{
     if ( $alter <18 )
         echo "Sie sind unter 18 ";

     else if ( $alter >18 ){
         echo "Sie sind volljährig!";
         echo "<a href='http://www.php.net'>PHP Beispiele</a>erteilt";
     }
}

echo "<br>";
if ($hobby=="Lesen")
    echo "Bravo, das Hobby ist Lesen";

else if ($hobby==“Sport")
   echo "$hobby ist super“;

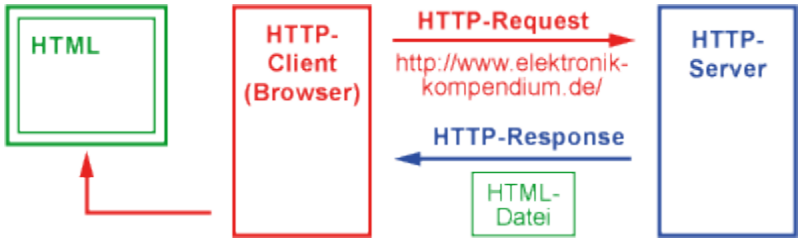
?>
```

3. HTTP - Hypertext Transfer Protocol

Quelle: http://www.elektronik-kompendium.de/sites/net/0902231.htm

HTTP ist das Kommunikationsprotokoll im World Wide Web (WWW). Die wichtigsten Funktionen sind Dateien vom Webserver anzufordern und zum Browser zu schicken. Der Browser übernimmt dann die Darstellung von Texten und Bilder und kümmert sich um das Abspielen von Audio und Video.

3.1. Das Hypertext Transfer Protocol (HTTP) im Schichtenmodell



HTTP (Hypertext Transfer Protocol)	
Familie:	Internetprotokollfamilie
Einsatzgebiet:	Datenübertragung, Hypertext u. a.
Port:	80/TCP
HTTP im TCP/IP-Protokollstapel:	
Anwendung	**HTTP**
Transport	TCP
Internet	IP (IPv4, IPv6)
Netzzugang	Ethernet, Token Bus, Token Ring, FDDI ...
Standards:	RFC 1945 (HTTP/1.0, 1996) RFC 2616 (HTTP/1.1, 1999)

Quelle: Wikipedia

3.2. Wie funktioniert HTTP?

Die Kommunikation findet nach dem Client-Server-Prinzip statt. Der HTTP-Client (Browser) sendet seine Anfrage an den HTTP-Server. Dieser bearbeitet die Anfrage und schickt seine Antwort zurück. Diese Kommunikation zwischen Client und Server findet auf Basis von Meldungen im Text-Format statt. Die Meldungen werden standardmäßig über TCP auf dem Port 80 abgewickelt. Die Meldungen werden Request und Response genannt und bestehen aus einem Header und den Daten. Der Header enthält Steuerinformationen. Die Daten entsprechen einer Datei, die der Server an den Client schickt oder im umgekehrten Fall Nutzereingaben, die der Client zur Verarbeitung an den Server übermittelt.

HTTP-Adressierung

Der HTTP-Client sendet eine URL welche auf eine Datei auf dem HTTP-Server verweist:

http://Servername.Domainname.Top-Level-Domain:TCP-Port/Pfad/Datei
z. B. *http://www.elektronik-kompendium.de:80/sites/kom/0902231.htm*

Die URL besteht aus der Angabe des Transport-Protokolls *"http://"*. Dann folgt der Servername (optional) und der Domainname mit anschließender Top-Level-Domain (TLD). Die Angabe zum TCP-Port ist optional und nur erforderlich, wenn die Verbindung über einen anderen Port, als dem Standard-Port 80 abgewickelt wird. Pfade und Dateien sind durch den Slash *"/"* voneinander und von der Server-Adresse getrennt. Folgt keine weitere Pfad- oder Datei-Angabe schickt der Server die Default-Datei der Domain.

Sind Pfad und/oder Datei angegeben, schickt der HTTP-Server diese Datei zurück. Ist diese Datei nicht existent, versucht er es mit einer Alternative. Gibt es keine, wird die Standard-Fehlerseite (Error 404) an den HTTP-Client übermittelt.

HTTP-Request

Der HTTP-Request ist die Anfrage des HTTP-Clients an den HTTP-Server. Ein HTTP-Request besteht aus den Angaben Methode, URL und dem Request-Header. Die häufigste Methoden sind GET und POST. Dahinter folgt durch ein Leerzeichen getrennt die URL und die verwendete HTTP-Version. In weiteren Zeilen folgt der Header und bei der Methode POST durch eine Leerzeile (!) getrennt die Formular-Daten.

HTTP-Methoden

Jeder HTTP-Request durch den Client wird durch die Angabe einer Methode eingeleitet. Die Methode weißt den Server an, was er mit dem Request machen soll. Es sind folgenden Methoden vorgesehen:

- **GET:** Anforderung einer Datei (z.B: HTML) vom Server, die Quelle wird durch die URL adressiert. Damit können auch Formular-Daten übermittelt werden, diese werden in codierter Form der URL angehängt. URL und Formular-Daten sind durch ein Fragezeichen (?) voneinander getrennt.
- **POST:** Zur Übermittlung von Formular-Daten an ein Programm (Skript) verwendet. Die Daten werden im Entity-Bereich getrennt durch eine Leerzeile vom Header übertragen.

- HEAD, PUT, OPTIONS, DELETE, TRACE, CONNECT

4. Variable

Eine Variable beginnt mit einem Dollarzeichen ($), anschließend ist zwingend ein Buchstabe oder "_". Bei der Definition werden keine Datentypen angegeben, der Typ wird anhand der Variablenzuweisung bestimmt.

PHP ist case-sensitiv, Groß-Kleinschreibung muss beachtet werden.
z.B.: $var und $vAr sind unterschiedliche Variable

unset($var)	löschen einer Variablenbelegung
isset($var)	prüft, ob die Variable belegt ist; nicht null
empty($var)	prüft ob die Variable einen Wert enthält (kein Wert bei: „“, 0, null, false)
is_numeric($var)	prüft, ob die Variable eine Zahl oder ein numerischer String ist

5. Datentypen und Typisierung der Variablen

Folgende Datentypen gibt es in PHP:

- **Skalarer Datentypen (speichern einzelne Variable):**

boolean	*true, false*
integer	*32bit System: $2^{32}/2= +/-2*10^{9,}$* *64bit System: $2^{64}/2= +/-1,8*10^{19}$*
float, double, real	*i.a. 64-bit IEEE (plattform-abhängig)*
string	*8-bit Zeichen (Ascii)* *‘XXX‘ oder “XXX“*

- **Zusammengesetzte Datentypen:**

  array
  object

- **Spezielle Datentypen:**

resource	*Referenz auf externe Quellen*
NULL	*Typ ohne Wert*

Der Datentyp wird i.a. nicht explizit angegeben, er muss auch nicht deklariert werden. Der Datentyp wird durch den zugewiesenen Wert bestimmt.

```
$a= 1;                  // $a ist integer
$b= 3.4;                // $b ist float
$c= $a+$b;              // $c ergibt 4.4
```

Bei Operationen mit Zahlen unterschiedlichen Types hat das Ergebnis den Höherwertigen Typ.

```
$i= 1;
$b= 2.1;
echo $i+$b;             // Ergebnis float: 3.1
echo $i.$b              // Ergebnis string: 12.1
```

5.1. gängige Funktionen

Ausgabe einer formatierten Zahl

string number_format (float $number [, int $decimals = 0])
z.B.: echo number_format($c,2): $c= 3.1567 => 3.16

Rundungsfunktionen

floor($var)	Abrunden
ceil($var)	Aufrunden
round($var)	Runden

mathematische Funktionen

sqrt($var)	Qaudratwurzel
pow($var,n)	n-te Potenz von $var
pi()	Wert von Pi

Zufallszahl

int rand (int min,int max) Liefert eine Zufallszahl zwischen min und max (inklusive)

Funktion var_dump($var)

Datentyp und Wert der Variable $var; bei arrays wird über alle Einträge geloopt
var_dump($_POST) gibt die Hashmap des Post-arrays aus
z.B.: array(3) { ["uebergabe_alter"]=> string(2) "30" ["hobby"]=> string(5) "Lesen" ["absenden"]=> string(6) "Senden" }

5.2. Typumwandlung (Casting):

$x= 12;	*x: integer*
$x = (float) $y;	*Umwandlung von Integer nach float*
	Variable $x bekommt Wert von $y

erlaubtes casting:
(int), (integer), (bool), (boolean), (float), (double), (real), (string),(array), (object),

6. Verzweigungen

6.1. IF

```
if ($bedingung) {
        // PHP Code
} else if  ($alternative) {
        // PHP Code
} else  {
        // PHP Code

}
```

If - Konstrukt (alternative)

```
echo ($x == 10) ? "Ja" : "Nein";
```

6.2. Verzweigung -Switch

Entscheidung, wenn mehrere bestimmte Werte geprüft werden müssen.

```
switch($variable) {
        case '100‘: // Code für Wert '100‘
                break;
        case '200‘: // Code für Wert '200‘
                break;
}
```

Break sorgt für Ende der Abarbeitung im Switch. Fehlt der Break-Befehl, werden alle Befehle danach im Switch noch ausgeführt. Kann auch gewollt sein:

7. Operatoren

Operatoren wie in Java:

+	Addieren
-	Subtrahieren
*	Multiplizieren
/	Dividieren
%	Modulo
++	Inkrement
--	Dekrement
+=, -=, /=, *=, …	Operation und Zuweisung in einem

Vergleichsoperatoren:

== Vergleich ob der Wert identisch ist

=== Vergleich ob Wert identisch ist und Operatoren den gleichen Datentyp haben.

Logische Operatoren:

|| (or), && (and), xor, ! (not)

8. $_POST - Array

Bei Fomular mit methode=“post“ beinhaltet das $_POST-Array alle Formularfelder (Textfelder, Radiobuttons, ...)

1. Textfelder: Jedes Textfeld ist mit seinem Namen im ein Element des $_POST-Arrays
2. Radiobuttons
    a. Alle Buttons einer Gruppe haben den gleichen Namen, daher nur eine Auswahl.
    - Der value findet sich im $_POST-Array unter dem Index des Namens.
    - Wenn kein Button ausgewählt ist, ist kein Wert im $_POST-Array -> überprüfen mit isset!

- z.B.: if (!isset($_POST['fav'])) echo 'Bitte auswählen!
';
- Das kann man durch Vorauswahl (Attribut checked="checked") vermeiden.

9. Operatoren mit Variablen

Zuweisung einer Variable (by Value)

```
$var = 10;
$var2 = $var;
$var2 = 20;
```

$var hat den Wert 10, $var2 hat den Wert 20

Zuweisung einer Variable (by Reference)

```
$var = 10;
$var2 = & $var;
$var2 = 20;
```

$var hat den Wert 20, $var2 hat den Wert 20

Doppeltes $ Zeichen

Variable wird ausgewertet und Wert der Variable als tatsächlicher Variablennamen herangezogen.

z.B:
```
$gehalt = 1000;
$variable = "gehalt";
echo $$variable;
```

Ergebnis: Es wird "1000" ausgegeben

Operatoren mit Variablen

Ein Funktionsaufruf wird in einer Variablen gespeichert.

z.B.:
```
$wert = "hallo";
$funktion = "strlen";
echo $funktion($wert);
```

Es wird die Funktion strlen("hallo") aufgerufen. Diese gibt den Wert "5" aus.

10. Strings

Strings sind Zeichenketten, sie werden mit einem Punkt (.) zusammengefügt und mit " (double tick) oder ' (tick) definiert.

z.B.: $var = "Hallo"."wie"."gehts";

echo "$var" oder echo $var: OUTPUT: "Hallowiegehts"
echo '$var' OUTPUT: $var

Bei einer Wertzuweisung (oder Ausgabe mittels echo) werden in doppelten Anführungszeichen enthaltene Variable ausgewertet und deren Werte eingefügt, bei einfachen Anführungszeichen nicht!

z.B.: echo "<td class=\"C1\"></td>";
echo "<td class='C3'></td>";

Werden in einem String Hochkomma benötigt kann das durch Verwendung von " als äussere Hochkomma und ‘ als innere oder durch Vorsetzen von \ vor dem " geschehen.

Ein Zeichen eines strings:

$str= "Spengergasse";
$str[0]: S
$str[3]: n

ltrim(string $str): entfernt whitespaces links des Strings
rtrim(string $str): entfernt whitespaces rechts des Strings
trim(string $str): entfernt whitespaces links und rechts des Strings

substr(string $str, int $start [,int $len]): gibt einen Teil eines Strings zurück
$str: Vorgabe-String
$start: Start-Position des Rückgabe-Strings
$len: Länge des zurückgegeben Strings, optional, sonst bis Ende

str_replace(string $str, string $alt, string $neu): ersetzt alle Vorkommen des Suchstrings durch einen anderen String
$alt: Suchstring
$neu: Ersetzungswerte
$str: String in welchem ersetzt wird

strpos(string $str, string $suche [, int $startpos]): sucht das erste Vorkommen des Suchstringes
$str: String in welchem gesucht wird
$suche: zu suchende Zeichenkette
$startpos: Startposition für Suche

strlen(string $str): Anzahl der Buchstaben der Zeichenkette $str

11. Schleifen und Arrays

11.1. Schleifen

While - Schleife

Schleife, die so oft wiederholte wird, wie die Bedingung $bedingung zutrifft.

```
while  ($bedingung) {
        // PHP Code
}
```

Do-While - Schleife

Schleife, die so oft wiederholte wird, wie die Bedingung zutrifft. Die Bedingung wird jedoch erst am Ende geprüft.; der Inhalt der Schleife wird mindestens einmal durchlaufen

```
do {
        // PHP Code
} while  ($bedingung)
```

For-Schleife

```
for ($start; $bedingung; $increment) {
        // PHP Code
}
```

z.B.:

```
for ($x = 1; $x < 10; $x++) {
        echo  "Aktueller Wert von x = ".$x;
}
```

11.2. Arrays

Arrays sind Objekte, die mehrere Elemente beinhalten können. Elemente können durch einen eindeutigen Namen (assoziatives Array) oder durch eine Nummer (numerisches Array) identifiziert werden.

Array anlegen

```
$arr = array();                 // Erzeugt ein leeres  Array
$arr = array(10, 20, 30);       // Erzeugt Array mit  {10, 20, 30}
```

Array befüllen

```
$arr[10] = "Hallo";             // Befüllt Arrayindex 10 mit dem Wert "Hallo"
$arr[] = "wie geht's?";         // Erzeugt einen neuen Arrayindex und befüllt ihn mit "wie geht's?"
```

Assoziatives Array anlegen und befüllen

```
$arr = array("Eins" => "One", "Zwei" => "Two", "Drei" => "Three");
$arr["Vier"] = "Four";          // fügt ein zusätzliches Element an
```

Ausgabe aller Arrayelemente

```
print_r($arr);                  // Abfrage aus einem Array
$arr["stadt"]                   // Liefert den Wert "Wien"
```

11.3. Array sortieren

sort(array $arry [,$sortierTyp])	Normale Sortierung der Werte Schlüssel geht verloren
asort(array $arry [,$sortierTyp])	Sortierung nach den Werten, die Zuordnung zum Schlüssel bleibt erhalten
ksort(array $arry [,$sortierTyp])	Sortierung nach dem Schlüssel
rsort(array $arry [,$sortierTyp])	Inverse Sortierung der Werte Schlüssel geht verloren

Sortiertypen:	SORT_REGULAR	Werte werden lt. Typ interpretiert; Zahl als Zahl, ...
	SORT_NUMERIC	Werte werden als Zahlen interpretiert
	SORT_STRING	Werte werden als Zeichenketten interpretiert

11.4. Sortier-Beispiele für Arrays

Array sortieren

```
$arr = array(        "b" => "Rapid",
                     "a" => "Salzburg",
                     "c" => "Austria")
```

sort($arr) liefert:
```
("0" => "Austria",
"1" => "Rapid",
"2" => "Salzburg")
```

asort($arr) liefert:
```
("c" => "Austria",
"b" => "Rapid",
"a" => "Salzburg")
```

ksort($arr) liefert:
```
("a" => " Salzburg ",
"b" => "Rapid”,
"c" => " Austria ")
```

rsort($arr) liefert:
```
("0" => "Salzburg",
"1" => "Rapid",
"2" => "Austria")
```

11.5. For-Each - Schleife

Schleife für iterierbare Elemente (z.B.: Array)

```
$arr = array();
$arr['a‘] = "Erster Eintrag";
$arr['xx‘] = "Zweiter Eintrag";

foreach($arr as $key => $value) {
        echo "Element $key mit Wert $value";
}
```

liefert: Element a mit Wert Erster Eintrag
Element xx mit Wert Zweiter Eintrag

11.6. Arrayfunktionen

in_array($suche, array $array [, bool $typ]): Suche nach Werten in einem Array
return: true falls Wert vorhanden, sonst false

array_search($suche, array $array [,bool $typ])
return: Key des Wertes falls vorhanden, , sonst false

in_array und array_search werden typensicher geprüft (===) wenn $typ= true

array_key_exists($gesucht, array $array)
return: true falls Wert vorhanden, sonst false

$arr= explode(";",$stVar);
die String Variable $stVar wird in das Array $arr mit Trennzeichen ";" aufgesplittet
z.B.: $stVar= "rot;gelb;blau";
$arr= explode(";",stVar): $arr[0]= "rot";
$arr[1]= "gelb";
$arr[2]= "blau";

sizeof($arr), count($arr): Anzahl der Elemente des Arrays; idente Funktionen

Weitere Funktionen sind in http://www.php.net/manual/de/book.array.php beschrieben.

12. Modularisierung

12.1. PHP Funktionen

- ```
  <?php
  function func1($arg_1, $arg_2)
  {
      .....
      return $retval;
  }
  ?>
  ```

- **Übergabe der Pararamter „by value":**

  ```
  <?php
  function func1($string)          {
          $string.= 'and something extra.';
  }

  $str = 'This is a string, ';
  func1($str);
  echo $str;                  // OUTPUT: 'This is a string',
                                 Variabale $str wurde nicht verändert
  ?>
  ```

- **Übergabe der Pararamter „by reference ":**

  ```
  <?php
  function func1(&$string)          {
          $string .= 'and something extra.';
  }

  $str = 'This is a string, ';
  func1($str);
  echo $str;                  // OUTPUT: 'This is a string, and something extra.',
                                 Variabale $str wurde durch Funktion verändert
  ?>
  ```
  ```

12.2. Gültigkeitsbereich (scope) von Variablen

- **local:** gilt nur innerhalb einer Funktionen
- **global:** ausserhalb der Funktionen, in einer Funktion Zugriff durch global $var
- **superglobal**: An allen Stellen im Skript sichtbar. Sind Arrays, die Daten von PHP bzw. über den Webserver beinhalten und somit dem Programmierer zugänglich gemacht werden
  $_POST, $_GET, $_COOKIE, $GLOBALS (ACHTUNG: $GLOBALS ohne „underscore“)

12.3. Zugriff auf eine externe Datei

Die Funktion fopen öffnet eine Datei zum Schreiben und/oder Lesen und gibt ein Dateihandle zurück. Mit diesem kann mit den Funktionen fgets, fputs usw. auf die Datei zugegriffen werden.

z.B.: **$fh = fopen("c:\\verzeichnis\\ressource.txt", "r");**

$fh	Dateihandle, wenn Datei nicht vorhanden wird false zurückgegeben
"r"	Modus read
"a"	Modus write, neue Inhalt wird angefügt (Dateizeiger vor Schreiben ans Ende)
"w"	Modus write, eventueller Inhalt wird überschrieben

weitere Funktionen:

fclose($fh); Schließt die Datei mit handle $fh

string fgets (resource $handle): Liest eine Zeile von der Position des Dateizeigers am Dateiende ist wird false zurückgegeben.
ACHTUNG: man muss CR (Ascii Code 13) und LF (Ascii Code 10) am Zeilenende entfernen; die Funktion **trim** entfernt white-spaces am Anfang und Ende der Zeile – z.B. zeile= trim($zeile); oder mit **str_replace** – z.B. $text= str_replace($zeile,chr(13),"",);

int fputs (resource $handle , string $str) Schreibt den Inhalt einer Zeichenkette string in die Datei, auf welche das handle zeigt.
Retour: Anzahl der geschriebenen Bytes, bei Error false.

z.B.: Öffne die Datei woerter.txt und zähle ihre Zeilen

```
$fh= @fopen("woerter.txt","r");
 if ($fh==false)
        Fehlermeldung!
else  {
        $zz= 0;
        while (( $buffer = fgets ( $fh)) != false ) {
                $zz++;
        }
        fclose($fh);
}
```

@fopen: Die Warnung bei nicht vorhandener Datei wird unterdrückt -> mit Abfrage auf false kann Fehlermeldung ausgegeben werden.

Alternative: Einlesen der gesamten Datei in ein array - $lines

$lines = file ("c:/import/RLDaten.dpe");

Gibt die Datei in einem Array zurück. Jedes Element des Arrays entspricht einer Zeile in der Datei, das Zeilenende wird nicht entfernt. Im Fehlerfall wird FALSE zurückgegeben.

12.4. Inkludieren externer Bibliotheken / Skripts

PHP Skripte können andere Dateien zur Laufzeit inkludieren. Mehrfach verwendeter Code kann so in eine Datei ausgelagert werden. Externe Bibliotheken / Skripte müssen ebenfalls auf diese Art eingebunden werden

Inkludieren mit include(Datei) oder require(Datei)

Sucht die Datei und liest den Code aus der Datei an die Stelle der include Anweisung.
include: Warnung bei Fehler (z.B. Datei nciht vorhanden)
require: wenn Datei nicht vorhanden wird Skript mit einem Fehler abgebroche

Problem

Wenn include ('DateiXY‘) öfters ausgeführt wird, kann es z.B. passieren, dass eine Variable, Funktion, Klasse öfters definiert wird => FEHLER

Lösung: include_once (Datei);
require_once (Datei);

include_once, require_once haben dieselbe funktionalität wie include bzw. require , nur wird eine Datei bei mehrmaligem Aufruf nur einmal inkludiert!

13. Superglobales

Sind Arrays, die Daten von PHP bzw. über den Webserver beinhalten und somit dem Programmierer zugänglich gemacht werden. Es gibt:

- Arrays mit Daten über den Webserver / PHP
- Arrays mit Daten über Benutzereingaben

Diese arrays beginnen immer mit $_.

13.1. $_POST, $_GET

$_POST Array

Liefert Inhalte, die mittels POST übertragen werden (z.B. aus einem HTML Formular). Wenn es z.B. das Formularfeld "name " gibt, wird dieses mit $_POST['name'] ausgelesen.

$_GET Array

Wie beim $_POST Array, nur bei der GET Anweisung.

13.2. $_SESSION

Eine Session wird beim ersten Start des Clients (Browser) an einem PHP-Server angelegt und identifiziert diesen eindeutig.

Anwendungsgebiete sind:

- Übergabe von Informationen zwischen PHP Skripts auf einer Webseite.
- Anmeldung als User über eine bestimmte Zeit.

Wenn das Browserfenster geschlossen wird oder nach Ablauf einer gewissen Zeitspanne (ohne Aktion, standardmäßig meistens 30 Minuten) wird die Session beendet und alle Variablen des Arrays $_SESSION werden gelöscht.

Achtung:

session_start muss vor der ersten Ausgabe gestartet werden, eine leere Zeile (Zeilenumbruch) oder ein Space vor session start ergint schon folgende Fehlermeldung: *Cannot send session cookie - headers already sent.*

Bsp. zum Arbeiten mit Sessions:

```
<?php
session_start();
$_SESSION["berechtigter_User"] = 1;
$_SESSION["Username"] = $username;
?>
```

Auf jeder weiteren Seite, auf der wir die Session durch die Verwendung von session_start() aufrufen, können wir jetzt diese Variablen nutzen. Wir können also z. B. den User begrüßen, ohne erneut seinen Usernamen abfragen zu müssen. Voraussetzung ist nur session_start zu Beginn.

```
<?php
session_start();
echo "Hallo ",$_SESSION["Username"],"!";
?>
```

Beenden der Session:

```
<?php
$_SESSION = array();
session_destroy();
?>
```

13.3. $_REQUEST

Dieses Array ist ein Sammelarray für Benutzereingaben / Werte, es beinhaltet unter anderem Werte aus folgenden Arrays:

- $_POST
- $_GET
- $_COOKIE

13.4. $_COOKIE

Cookies sind kleine Informationsspeicher (files) auf dem Client. Sie haben eine bestimmte Lebensdauer.

Anwendungsbeispiele sind:

Speichern von Kundendaten
Session ID speichern
Einkaufswagen im Onlineshop

z.B.: $_COOKE['Keksname'] liefert das entsprechende Cookie.

13.5. $_FILES

Beinhaltet Informationen über alle Dateien, die auf den Server hochgeladen wurden. Jedes Arrayelement beinhaltet folgende Informationen

- name Name der Datei
- type Mime-Type
- tmp_name Temporärer Name auf dem Server
- error Fehlercode
- size Größe der Datei in Byte

Beispielausgabe mittels print_r($_FILES)

```
Array
(
    [uploadFile] => Array
    (
        [name] => datei.txt
        [type] => text/plain
        [tmp_name] => C:\xampp\tmp\php102.tmp
        [error] => 0
        [size] => 102
    )
)
```

13.6. $_SERVER

Enthält Informationen über den HTTP Header der aktuellen Anfrage, sowie lokale Pfade des Betriebssystems und Skriptes.

Wichtige Einträge sind:

- $_SERVER['DOCUMENT_ROOT‘] Webverzeichnis
- $_SERVER['PHP_SELF‘] Relativer Pfad im Webverzeichnis
- $_SERVER['HTTP_USER_AGENT‘] Identifier über verwendeten Browser

$_SERVER['HTTP_REFERER‘] wenn verlinkt - > letzte Website, sonst leer

13.7. $GLOBALS

Enthält alle Referenzen aus den superglobalen Arrays

z.B. $GLOBALS['_POST‘]['file‘]

Unter anderem sind folgende $GLOBALS bei der Fehlersuche von Interesse:

- __LINE__ Aktuelle Zeilennummer
- __FILE__ Aktueller Pfad des ausgeführten Skripts
- __FUNCTION__ Aktuelle Funktion