KNF

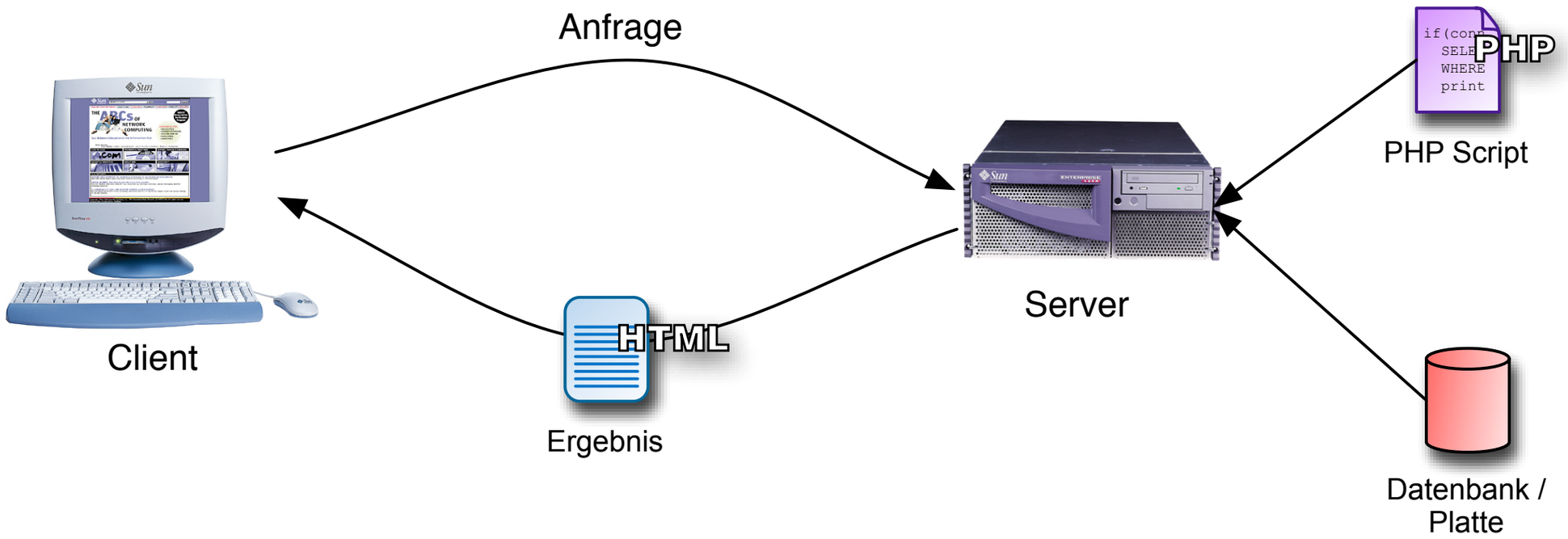
PHP

Dynamische Webseiten mit PHP - Ein Einstieg in die vermutlich populärste Skriptsprache für das WWW

PHP

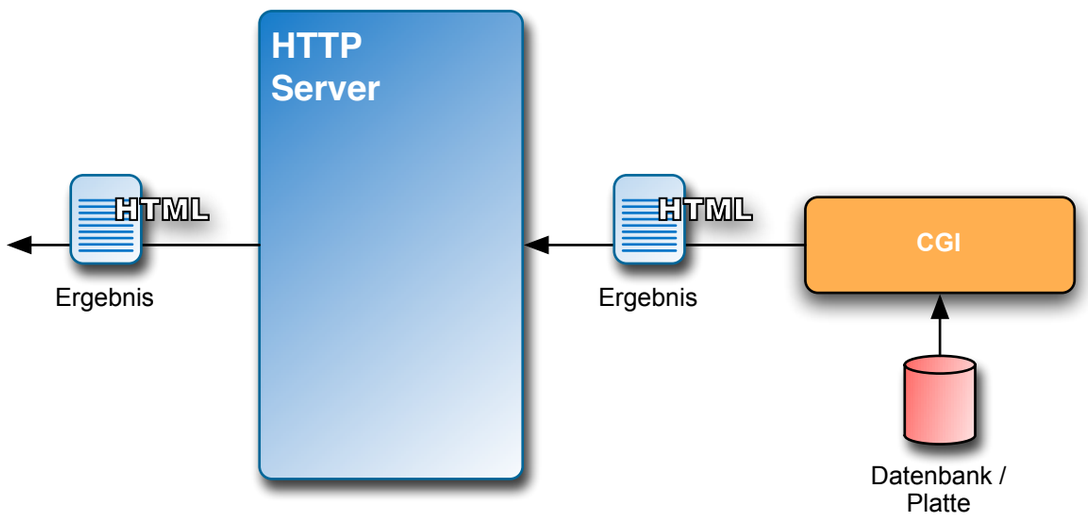
- Personal Home Page
- PHP: Hypertext Preprocessor
- Rasmus Lerdorf (* 22. November 1968 in Godhavn/Qeqertarsuaq, Grönland) schrieb die erste Version von PHP „PHP/FI“: „Personal Home Page/Forms Interpreter“
- Interpretierte Sprache
- Ausgerichtet auf Bearbeitung von Web-Anforderung
- einfache Variablen (automatische Typ-Konvertierung)
- Funktionen
- Seit 4.0 Objekt-orientierte Programme möglich
- große Funktionsbibliothek (z.B. MySQL/Postgres/Oracle/LDAP, XML, XSLT, SOAP, HTTP, Mail, Date, Streams

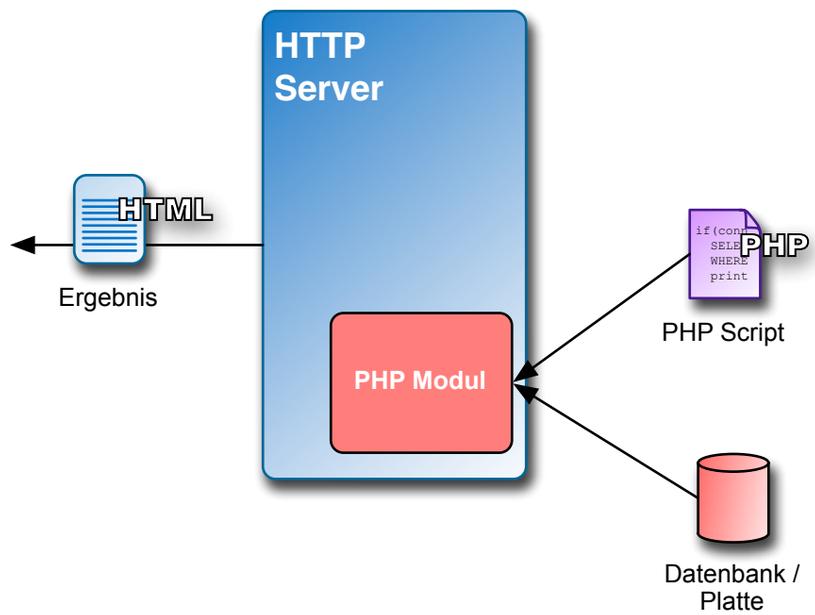
PHP-Seiten



- Server erhält Anfrage
- Server liest PHP-Skript und interpretiert dieses
- durch Skript werden eventuell weitere Dateien und Skripte eingebunden, Datenbank-Abfragen getätigt oder Web-Requests geschickt
- Server schickt Ergebnisdokument an den Client

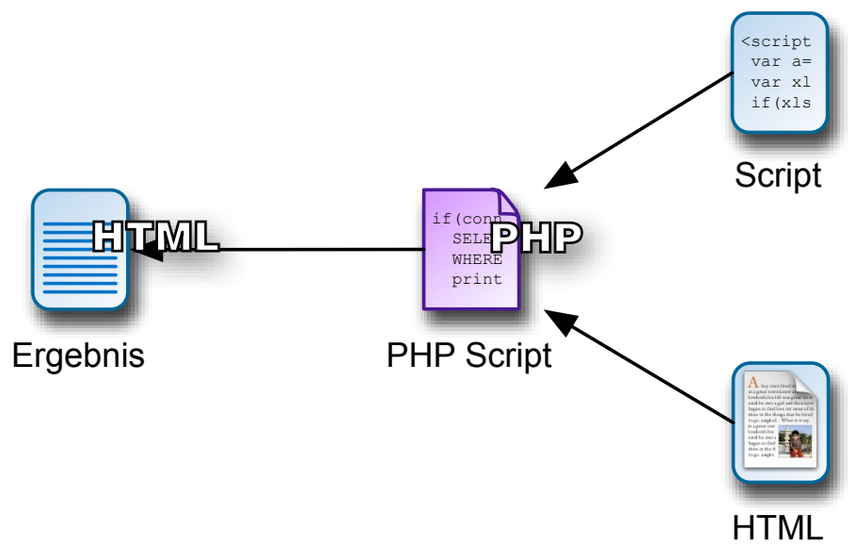
CGI im Vergleich zu "embedded" PHP





- Das „CGI“ muss vom Webserver für jeden Request gestartet werden
- Das CGI liefert nahezu fertiges Dokument zurück, der Webserver bearbeitet die gelieferten Daten kaum
- „PHP“ wird für gewöhnlich im Webserver ausgeführt und muss nicht für jeden Request neu gestartet werden
- Die „PHP“-Umgebung liefert bereits Schnittstellen zu Datenbanken, Files, URLs usw.

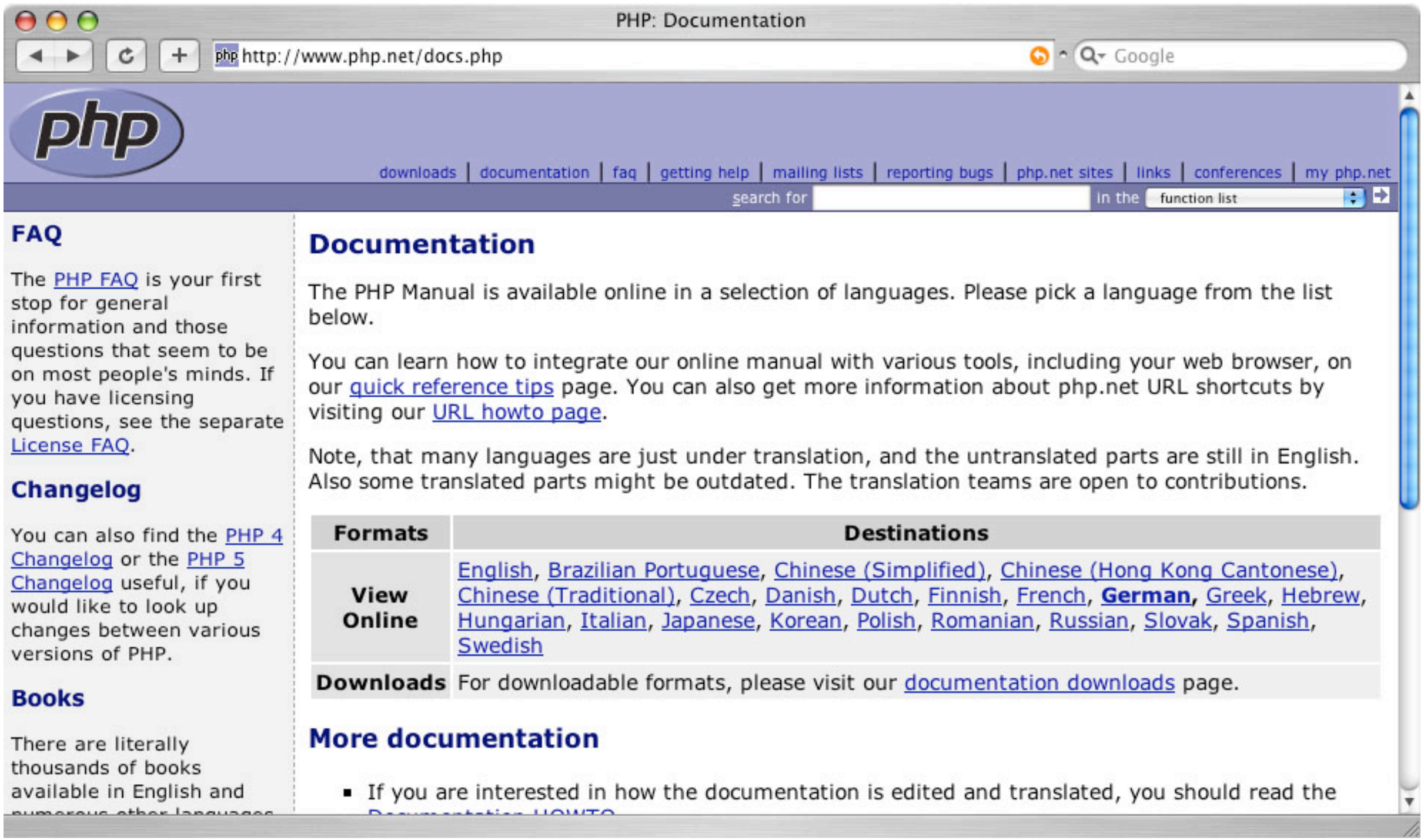
Text und Code kombiniert



- HTML im Code
- Code in HTML

```
<body>
<h1>Willkommen</h1>
Wir begrüßen sie am <?php echo $date; ?> auf unseren Webseiten.
</body>
```

Dokumentation



- http://www.php.net/

KNF

Erste Schritte

PHP Grundwissen

Hello World

```
<html>
<head>
   <title>Hello</title>
</head>
<body>
<?php
      echo "hello world";
?>

<hr>
<?php echo "(C) 2008 by KNF"; ?>
</body>
</html>
```

- Code-Teile werden in "<?php" und "?>" eingeschlossen
- Texte und Variablen werden mit "echo" oder "print" ausgegeben
- Statements werden mit Semikolon (";") beendet
- In einer Zeile können mehrere Statements stehen

```
<?php
   echo "Hallo, "; echo "wie geht es Dir?";
?>
```

Variablen

- Variablen beginnen mit dem Dollar-Zeichen ("$")
- Die Namen von Variablen müssen mit einem Buchstaben oder Unterstrich beginnen und können danach beliebige Kombinationen von Buchstaben, Zahlen und Unterstrichen enthalten
- Einer Variablen wird mit dem Gleichheitszeichen ("=") ein Wert zugewiesen
- Variablen mit unterschiedlicher Groß- und Kleinschreibung sind verschieden

```
$name = "Joerg";

$Name = "Joerg Kinzebach";

$anzahl = 20;

$user01 = "Joerg";

$user02 = "Thomas";

$_name = "Udo";

$2user = "20";  !UNGÜLTIG!


echo $name;
    Joerg
echo $Name;
    Joerg Kinzebach
```

Variablen und deren Ausgabe

- Doppelte Anführungszeichen erlauben die Ausgabe von Variablen (" ")
- Einfache Anführungszeichen geben den Text ohne Ersetzungen aus (' ')

```
$name = "Joerg";

echo "Ich bin $name";
    Ich bin Joerg

echo 'Ich bin $name';
    Ich bin $name
```

- echo und print erzeugen keinen Zeilenumbruch
- Mittels "\n" kann ein Zeilenumbruch ausgegeben werden

```
echo "Ich bin $name";
echo "und zeige hier die Folien";
    Ich bin Joergund zeige hier die Folien

echo "Ich bin $name\n";
echo "und zeige hier die Folien";
   Ich bin Joerg
   und zeige hier die Folien
```

Vereinfachte Variablen-Ausgabe und Code-Inseln

- Vereinfachte Ausgabe von Variablen außerhalb einer Code-Insel:

```
Hallo <?=$name ?>, herzlich Willkommen.
```

- alternative Arten eine Code-Insel aufzuspannen:

```
<?php
   echo "Hallo!";
?>


<? echo "Hallo!"; ?>


<% echo "Hallo!"; %>



<script language="php">
   echo "Hallo!";
</script>
```

Operatoren

- Arithmetische Operatoren

-$a	Negierung	Negativer Wert von $a
$a + $b	Addition	Summe von $a und $b
$a - $b	Subtraktion	Differenz von $a und $b
$a * $b	Multiplikation	Produkt von $a und $b
$a / $b	Division	Quotient von $a und $b
$a % $b	Modulus	Rest von $a geteilt durch $b

- Zuweisungen

$a = 1;	Numerische Wertzuweisung
$a = "test";	String-Zuweisung
$a -= 5;	Abgekürzte Subtration
$a += 5; $a *= 2;	Abgekürzte Addition, Multiplikation
$b .= " und";	Anhängen eines Strings

- Vergleichs Operatoren

$a == $b	Gleich	liefert true wenn $a gleich $b
$a === $b	Identisch	liefert true wenn gleicher Inhalt und Typ
$a != $b	Ungleich	liefert true wenn $a nicht gleich $b
$a <> $b	Ungleich	liefert true wenn $a nicht gleich $b
$a < $b	Kleiner Als	liefert true wenn $a kleiner als $b ist
$a > $b	Größer Als	liefert true wenn $a größer als $b ist
$a <= $b	Kleiner Gleich	liefert true wenn $a kleiner oder gleich $b ist
$a >= $b	Größer Gleich	liefert true wenn $a größer oder gleich $b ist

Variablentypen

- String (Zeichenketten)
- Integer (Ganzzahlen)
- Float (Fließkommazahlen)
- Array (Datenmenge/Tabelle)
- Object (Mischung aus Funktionen und Daten)
- Resource (beliebige externe Information, z.B. eine Grafik)

- Der Typ einer Variablen wird soweit möglich von PHP automatisch beim Zugriff umgewandelt

```
$name = "KNF";
$alter = 11;

$string = $name . " ist " . $alter . " Jahre alt.";
echo $string;

  KNF ist 11 Jahre alt.

$jahr = "2008";
$anfangsjahr = $jahr - $alter;
print "Angefangen hat alles $geburtsjahr";

  Angefangen hat alles 1997
```

$$variable und Referenzen

```
$ampel_1 = "rot";
$ampel_2 = "rot";
$ampel_3 = "rot";

$aktuell = "ampel_2";

$$aktuell = "gruen";

print $ampel_2;
  gruen
```

- mit $$variable wird auf die Variable zugegriffen, deren Name in $variable steht

- bei einer Zuweisung von einer Variablen zur anderen werden normalierweise Kopien der Daten übergeben, will man dies vermeiden muss man den Referenzoperator =& verwenden

```
$ampel_1 = "rot";
$ampel_2 =& $ampel_1;

print $ampel_2;
  rot

$ampel_1 = "gruen";
print $ampel_2;
  gruen
```

KNF

Praxis

erste eigene PHP-Seite

1. Übung

- Seite mit einer Text-Ausgabe aus PHP

Netzwerk-Verbindung
http://php.franken.de/edit

Resource auf Server editieren
~/uebung1/index.php

PHP-Code erstellen

Aufruf der erstellten Seite
http://php.franken.de/user00/uebung1/

phpinfo()

- Ausgabe der Version, Einstellungen und verfügbaren Erweiterungen von PHP

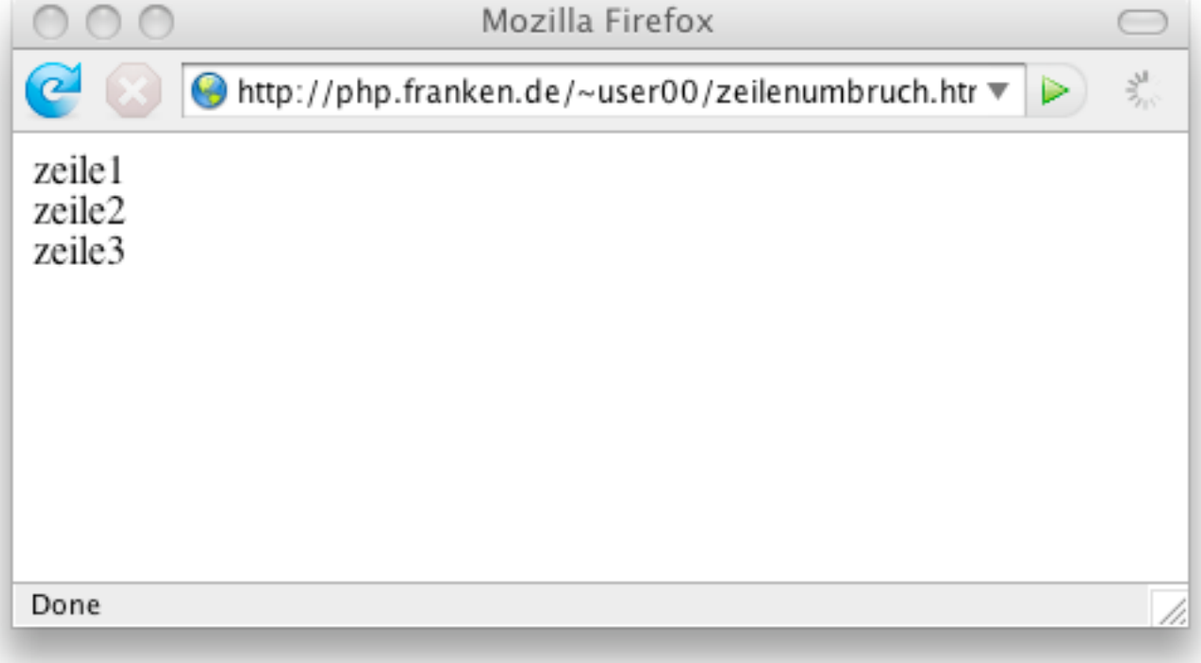
```
<?php
    phpinfo();
?>
```

System	Linux php 2.6.18-3-686 #1 SMP Thu Dec 14 18:16:18 CET 2006 i686
Build Date	Nov 3 2006 21:49:22
Configure Command	'../configure' '--prefix=/usr' '--with-apxs2=/usr/bin/apxs2' '--with-config-file-path=/etc/php4/apache2' '--enable-memory-limit' '--disable-debug' '--with-regex=php' '--disable-rpath' '--disable-static' '--with-pic' '--with-layout=GNU' '--with-pear=/usr/share/php' '--enable-calendar' '--enable-sysvsem' '--enable-sysvshm' '--enable-sysvmsg' '--enable-track-vars' '--enable-trans-sid' '--enable-bcmath' '--with-bz2' '--enable-ctype' '--with-db4' '--with-iconv' '--enable-exif' '--enable-filepro' '--enable-ftp' '--with-gettext' '--enable-mbstring' '--with-pcre-regex=/usr' '--enable-shmop' '--enable-sockets' '--enable-wddx' '--disable-xml' '--with-expat-dir=/usr' '--with-xmlrpc' '--enable-yp' '--with-zlib' '--without-pgsql' '--with-kerberos=/usr' '--with-openssl=/usr' '--with-zip=/usr' '--enable-dbx' '--with-mime-magic=/usr/share/misc/file/magic.mime' '--with-exec-dir=/usr/lib/php4/libexec' '--without-mm' '--without-mysql' '--without-sybase-ct'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php4/apache2/php.ini
PHP API	20020918

HTML: Zeilenumbrüche

```
<html>
<head>
   <title>Seitentitel</title>
</head>
<body>
  zeile1
  zeile2
  zeile3
</body>
</html>
```


```
<html>
<head>
   <title>Seitentitel</title>
</head>
<body>
  zeile1<br>
  zeile2<br>
  zeile3<br>
</body>
</html>
```



- Zeilenumbrüche in Quelltext von HTML-Seiten führen nicht zu Umbrüchen in der Ausgabe, für die Ausgabe müssen diese "erzwungen" werden

Kommentare

- Kommentare dürfen nur in der PHP-Insel stehen!
- PHP kennt drei Arten Kommentare einzufügen:

```
# einzeiliger Kommentar, wie auch in PERL
# echo "Sie sind $name.";

// einzeiliger Kommentar wie von C/C++/Java bekannt
// echo "Sie sind $name.";

/*
 * Mehrzeiliger Kommentar
 */

/*
$name = $vorname . ' ' . $nachname;
echo "Hallo $name";
*/
```

- Verschachtelung von /* */ nicht möglich!

```
/*
  /*
   * Zusammensetzen des Namens mittels Vor- und Nachname,
   * die durch ein Leerzeichen getrennt werden
   */
  $name = $vorname . ' ' . $nachname;
  echo "Hallo $name!";
*/
```

Funktionen

- Erstellen von eigenen Funktionen um wiederkehrende Aufgaben schnell zu lösen
- Wiederverwertung von Code-Teilen in mehreren Projekten

```
// einfachste Funktion, liefert immer 1 zurück
function eins() {
   return 1;
}


echo "Das Ergebnis ist " , eins();
```

Das Ergebnis ist 1

```
<?php

function ueberschrift() {
   echo "<h1>Meine Überschrift</h1>\n";
}

?>

<html><head><title>Mein PHP</title></head><body>


<? ueberschrift(); ?>
<p>Im März veranstaltet der KNF wieder eine Reihe von Aktionen</p>

<? ueberschrift(); ?>
<p>Außerdem stehen einige Termine, wie zum Beispiel der Stammtisch an</p>

</body></html>
```

Funktionsparameter

- Bei der Deklaration der Funktion werden die Parameternamen festgelegt
- Innerhalb der Funktion stehen die Parameter als Variablen zur Verfügung

```
function name($nachname,$vorname) {
   return $vorname . ' ' . $nachname;
}


echo "Hallo " . name("Meier","Hans");

    Hallo Hans Meier
```

```
<?php

function ueberschrift($txt) {
   echo "<h1>$txt</h1>\n";
}

?>

<html><head><title>Mein PHP</title></head><body>

<? ueberschrift("Veranstaltungen"); ?>
<p>Im März veranstaltet der KNF wieder eine Reihe von Aktionen</p>

<? ueberschrift("Termine"); ?>
<p>Außerdem stehen einige Termine, wie zum Beispiel der Stammtisch an</p>

</body></html>
```

Geltungsbereich von Variablen (Scope)

- In Funktionen oder Objekten definierte Variablen sind nur dort gültig
- Globale Variablen stehen nicht implizit in Funktionen und Objekten zur Verfügung

```
$nummer = 5;

function meintest($parameter)
{
    $nummer++;
    echo "Nummer in Funktion: $nummer\n";
}

echo "Nummer zuvor: $nummer\n";
meintest($nummer);
echo "Nummer danach: $nummer\n";
```

```
Nummer zuvor: 5
Nummer in Funktion: 1
Nummer danach: 5
```

- Mit dem Schlüsselwort "global" können globale Variablen in eine Funktion geholt werden:

```
function meintest($parameter)
{
    global $nummer;
    $nummer++;
    echo "Nummer in Funktion: $nummer\n";
}
```

Einbinden weiterer Dateien/Skripten

- require(), require_once()
  Bindet die als Argument übergebene Datei an der Stelle des Aufrufs in das Aufrufende Dokument ein
- include(), include_once()
  Bindet wie require() die angegebene Resource ein, bricht aber nicht ab, wenn diese nicht gefunden wurde

```
<?php
   require "/kopf.inc";
   require "/fuss.inc
?>

<? seiten_kopf("Meine Seite"); ?>

<h1>Inhalt</h1>

<p>Mein Seitentext</p>

<? seiten_fuss(); ?>
```

Verzweigungen & Bedingungen

Code-Ausführung steuern

Bedingungen

```php
$limit = 100;

if ($wert > $limit) {
    echo "Limit überschritten!";
} else {
    echo "Der Wert ist okay.";
}
```

```php
$alter = 18;

if ($alter < 6) {
    echo "kein Schulkind";
} elseif ($alter < 10) {
    echo "kein Teen";
} elseif ($alter < 18) {
    echo "ein Teen";
} else {
    echo "Volljährig";
}
```

```php
$ampel = "rot";

if ($ampel == "rot") {
    stop();
} elseif ($ampel != "gruen") { // gelb?
    bereit();
} else {
    go();
}
```

Fallunterscheidungen

- if ... elseif Blöcke vereinfachen

```
switch ($ampel) {
    case "rot":
        stop();
        break;
    case "gelb":
        ready();
        break;
    case "gruen":
        go();
        break;
    default:
        echo "Ich verstehe die Ampel nicht!";
}
```

- "break" beendet einen Block

- ohne "break" am Ende eines Case-Blocks wird der nächste Fall ebenfalls abgearbeitet.

Schleifen

wiederholte Code-Ausführung

While-Schleife

- Ausführung des Schleifenkörpers, solange Bedingung wahr ist
- Bedingung wird vor dem Abarbeiten des Schleifenkörpers überprüft

```
while ($wochentag == "montag") {
    jammern();
}


$i = 0;
while ($i <= 10) {
    print "Nummer $i\n";
    $i++;
}
```

```
while (1) {
   print "Endlosschleife!";
}


// Warten auf Montag
while (1) {
    if (getWochentag() == "Montag") {
        break;
    }
    warten();
}

// Besser:
while (getWochentag() != "Montag") {
    warten()
}
```

Do-While

- Schleifenkörper wird ausgeführt und danach geprüft, ob der Körper nochmals durchlaufen werden soll.

```
$i = 1;
do {
    print "Nummer $i\n";
    $i++;
} while (i <= 5);
```

- im Gegensatz zur "normalen" While-Schleife, wird bei do ... while der Körper mindestens einmal durchlaufen

For-Schleife

- Deklaration + Bedingung + Aktion
- Schleifenkörper wird ausgeführt solange Bedingung wahr ist
- Aktion wird nach der Ausführung des Schleifenkörpers abgearbeitet

```
for ( $i = 1; $i <= 10; $i++ ) {
    print "Nummer $i\n";
}


$ende = 0;
for ( $i = 0; $ende != 1; $i++) {
    if ($i > 10) {
         $ende = 1;
    } else {
         print "Nummer $i\n";
    }
 }
```

An die Editoren ... fertig ... Los!

zweite Praxisübung

2. Übung

- Erstellen einer PHP-Seite, die in einer Tabelle die Ergebnisse von
  f(x,y) = x + y
  darstellt.

- weniger mit der Programmierung vertraute Workshop-Teilnehmer sollten eventuell auf die Beschriftung von x und y verzeichten

HTML-Tipp:

```
<table border="1">
  <tr><td>2</td><td>3</td></tr>
  <tr><td>3</td><td>4</td></tr>
</table>
```

	x=1	**x=2**	**x=3**	**x=4**	**x=5**	**x=6**	**x=7**	**x=8**	**x=9**	**x=10**
y=1	2	3	4	5	6	7	8	9	10	11
y=2	3	4	5	6	7	8	9	10	11	12
y=3	4	5	6	7	8	9	10	11	12	13
y=4	5	6	7	8	9	10	11	12	13	14
y=5	6	7	8	9	10	11	12	13	14	15
y=6	7	8	9	10	11	12	13	14	15	16
y=7	8	9	10	11	12	13	14	15	16	17
y=8	9	10	11	12	13	14	15	16	17	18
y=9	10	11	12	13	14	15	16	17	18	19
y=10	11	12	13	14	15	16	17	18	19	20

Arrays

Multidimensionale Tabellen und mehr...

Einführung zu Arrays

- Der Datentyp "Array" kann mit der Funktion array( ) oder mit dem Operator [ ] erzeugt werden
- Ein Array ist ein Datentyp und gleichzeitig Kontainer für Daten
- Variablen innerhalb eines Arrays werden als Element bezeichnet

```
<?php
$wochentage = array("Montag","Dienstag","Mittwoch",
     "Donnerstag", "Freitag","Samstag","Sonntag");

print "Die Woche hat " . count($wochentage) .  "Tage.";

?>

<h1>Inhalt des $wochentage Arrays:</h1>
<pre>
<? print_r($wochentage); ?>
</pre>
```

0	Montag
1	Dienstag
2	Mittwoch
3	Donnerstag
4	Freitag
5	Samstag
6	Sonntag

- Array-Index beginnt bei "0"

```
print "Es ist " . $wochentage[2];

    Es ist Mittwoch
```

Assoziative Arrays

- PHP vergibt automatisch numerische "Schlüssel" zu Elementen von Arrays
- Es können auch eigene Schlüssel für die Adressierung von Elementen verwendet werden, ein sogenanntes "Assoziatives Array" ensteht. Diese werden auch oft als "Hash" bezeichnet

```
<?php
$wochentage = array(
  "mon" => "Montag",
  "die" => "Dienstag",
  "mit" => "Mittwoch",
  "don" => "Donnerstag",
  "fre" => "Freitag",
  "sam" => "Samstag",
  "son" => "Sonntag");

print "Heute ist " . $wochentage["son"];

?>

    Heute ist Sonntag
```

- Erzeugung mittels "Array-Operator":

```
<?php
  $monate["jan"] = "Januar";
  $monate["feb"] = "Februar";
  $monate["mär"] = "März";
?>
```

Array-Funktionen

- Liste aller Keys eines Assoziativen Arrays: array_keys( );

```
<? print_r(array_keys($wochentage)); ?>
```

- Suche nach dem Schlüssel für einen Element-Inhalt: array_search( );

```
<? print array_search("Mittwoch",$wochentage); ?>
```

- Existiert ein Wert in einem Array: in_array( );

```
<?
    $mitglieder = array("Klaus","Hans","Udo","Martin");
    $suche = "Martin";

    if (in_array($suche,$mitglieder)) { print "$suche ist Mitglied."; }
?>
```

- Iteration durch alle Werte eines Arrays:

```
<? foreach ($wochentage as $tag) {
        print $tag . " ";
   }
?>
```

Montag Dienstag Mittwoch Donnerstag Freitag Samstag Sonntag

```
<?  foreach ($wochentage as $key => $value) {
        print $key . '=' . $value . ' ';
    }
?>
```

mon=Montag die=Dienstag mit=Mittwoch don=Donnerstag fre=Freitag sam=Samstag son=Sonntag

Superglobals

- An PHP von Außen übergebene Variablen stehen Skripten als spezielle Arrays zur Verfügung. Da sie in jedem Kontext (in Objekten, in Funktionen usw.) existieren werden sie als "Superglobals" bezeichnet.

$_GET	Variablen, die dem Webserver von einem Formular mittles 'GET' Request übergeben wurden.
$_POST	Alle dem Webserver mittels 'POST' Request übersendeten Formulardaten
$_FILES	Variablen, die durch einen File-Upload erzeugt wurden
$_COOKIE	Alle Cookies, die mit dem HTTP-Request übertragen wurden
$_REQUEST	Alle mittels GET, POST, Upload gesendeten Variablen inklusive Cookies kombiniert. Da hier nicht mehr zwischen Übertragenen und lokalen Variablen unterschieden werden kann, ist $_REQUEST oft eine unsichere Methode um an Daten zu gelangen, die nicht bewusst vom Nutzer stammen.
$_SESSION	Wenn für den Nutzer eine Session aufgebaut wurde, enthält dieses Array alle in der Session gesetze Variablen.
$_SERVER	Variablen mit Informationen des Webservers und andere Variablen, die sich direkt auf die aktuelle Ausführung des PHP-Skripts beziehen
$_ENV	Vom System oder der Shell gesetzte Umgebungsvariablen
$GLOBALS	Alle globalen Variablen in Ihrem Skript, inklusive der obigen Superglobals

Superglobals: $_SERVER

HTTP_REFERER	URL der Seite, von der aus der User einem Link gefolgt auf das aktuelle Skript gefolgt ist (wird vom Nutzer gesandt und ist daher nicht vertrauenswürdig)
HTTP_USER_AGENT	Identifikations-String der Software des Nutzers, die den Request erzeugt hat
PATH_INFO	Wenn dem Skript in der URL ein Pfad angehängt wurde, steht er in diesem Element
PHP_SELF	Name des soeben ausgeführten PHP-Skripts
REQUEST_METHOD	Die zum Abruf des Skripts benutzte Methode: GET oder POST
QUERY_STRING	Der URL als GET-Request angehängte Parameter als String (z.B: "user=joerg&login=true&valid=false")

```
Dieses Skript ist <?=$_SERVER['PHP_SELF']?>

<?
    if (isset($_SERVER['PATH_INFO'])) {
      echo " und sie wollten " . $_SERVER['PATH_INFO'];
    }

    if (isset($_SERVER['HTTP_USER_AGENT'])) {
       echo ",  sie verwenden " . $_SERVER['HTTP_USER_AGENT'];
    }
?>
```

- Austesten obigen Codes in "uebung3"
- Wie erfahre ich die IP-Adresse des Anfragenden?

Tricks mit PATH_INFO

- Mittels Apache URL-Rewriting einen Pfad auf index.php verweisen, hierzu in die Apache-Config z.B. folgendes eintragen:

```
RewriteRule ^/~(user..)/(uebung3)(/.*)$ /home/$1/public_html/$2/index.php$3
```

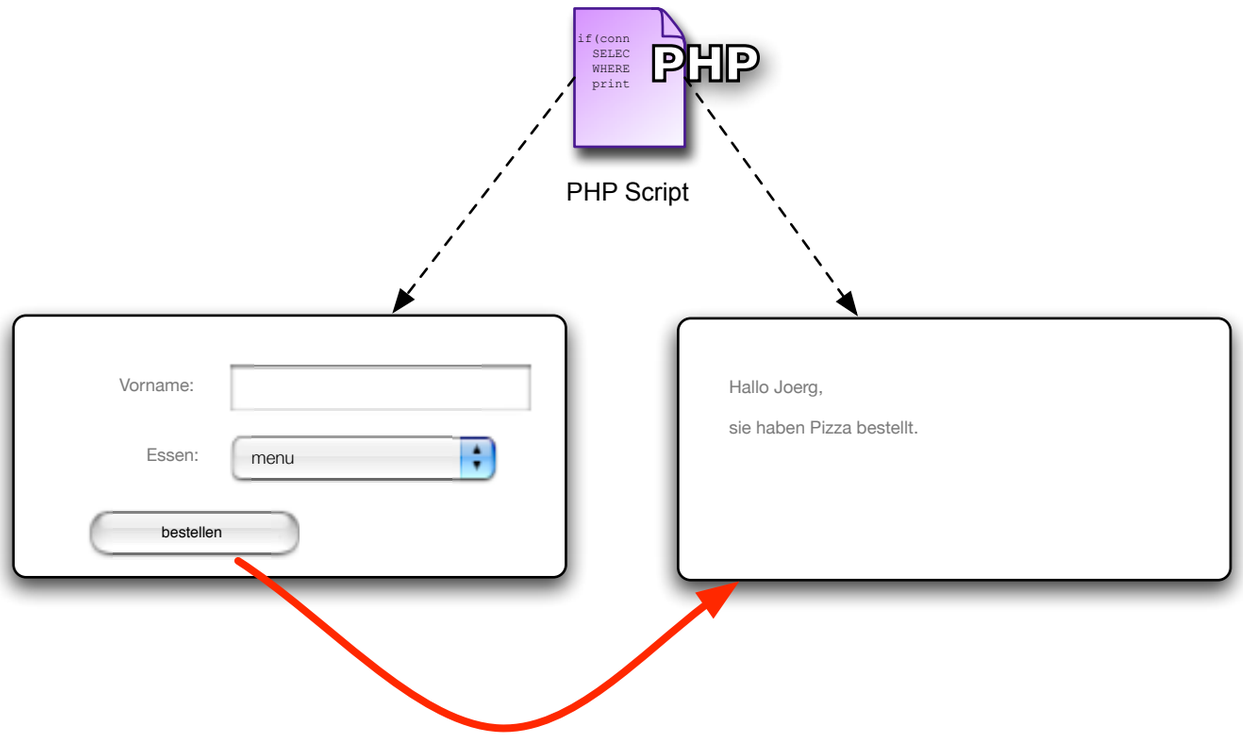
- Bei Zugriff auf /~user00/uebung3/foo wird durch obige Regel das PHP-Skript ~user00/public_html/uebung3/index.php aufgrufen und der Pfad angehängt
- Das Skript erhält den optionalen Zusatz-Pfad als PATH_INFO übergeben, somit können Zugriffe auf virtuelle Pfade ermöglicht werden bzw. Parameter in einer schöneren URL versteckt werden

  http://php.franken.de/~user00/uebung3/urlaubsbilder/2008

  anstatt

  http://php.franken.de/~user00/uebung3/index.php?path=/urlaubsbilder/2008)

4. Übung



- Web-Formular das die Eingabe eines Namens und die Auswahl einer Speise erwartet.
- Wenn Nutzer eine gültige "Bestellung" abschickt soll ihm seine Wahl angezeigt werden.
- Alle Aktionen sollen durch ein einziges Skript erledigt werden.
- Auf fehlende Angaben soll der Nutzer hingewiesen werden

vorbereitetes Gerüst und Formular liegt in uebung4/index.php

Security

- In der letzten Übung wurden vom Nutzer erhaltene Daten ungeprüft Ausgegeben!
  - Auf keinen Fall zum Öffnen von Dateien oder URLs verwenden!
  - Nicht direkt an Datenbank-Abfragen übergeben"
  - Niemals ungeprüfte Daten von Nutzern an Shell-Kommandos als Parameter übergeben!
  - Auch bei reiner Ausgabe können die Daten noch zu XSS-Angriffen genutzt werden

- Von Nutzern erhaltene Daten sollten immer einer Plausibilitätsprüfung unterzogen werden

```
<?
   /*
    * Regex zur Überpruefung des Nutzernamens, Nutzernamen beginnen
    * mit Buchstaben a-Z potentiell gefolgt von einigen Ziffern
    */
   if (!preg_match("/^[a-zA-Z]+\d*$/",$username)) {
       echo "Ungültiges Zeichen in der Zeichenkette";
       $username = '';
   }
?>
```

Security

- Testen des Typs von Variablen

```
$var = "joerg";
if (ctype_lower($var)) {
   print "Kleinbuchstaben";
} elseif (ctype_upper($var)) {
   print "Grossbuchstaben";
}
```

ctype_alnum( )	Buchstaben und Zahlen (A-Z, a-z, 0-9)
ctype_alpha( )	Buchstaben (A-Z, a-z)
ctype_digit( )	Zahlen (0-9)
ctype_cntrl( )	ASCII-Steuerzeichen
ctype_print( )	Darstellbare (druckbare) Zeichen ausser Leerzeichen
ctype_space( )	Leerzeichen, Tabulatoren, Zeilenbrüche usw.
ctype_xdigit( )	Zahlen im Hexformat (0x2a)

- Magic Quotes

  Einstellung in php.ini: Wenn aktiviert werden Anführungszeichen, Backslashes und Null-Zeichen automatisch gequoted, so daß Strings keine für z.B. Datenbankaufrufe gefährlichen Zeichen enthalten (unvollständig!)

Security

- Sonderbehandlung von Speziellen Zeichen:
  - mysql_escape_string( )

    Fügt Quotes für Zeichen in, die in SQL-Statements eine spezielle Bedeutung haben
  - escapeshellcmd( )

    Setzt Quotes vor Zeichen, die ein Shell-Kommando dazu veranlassen könnten, weitere oder andere Kommandos auszuführen (z.B. # & ; ` | * ? ~ < > ^ ( ) [ ] { } $ \ \x0A and \xFF. ' und ")

- Ausgabe von speziellen Zeichen:
  - htmlspecialchars( )

    Wandelt HTML-Sonderzeichen in HTML-Escapes um
  - strip_tags( )

    Entfernt HTML- und PHP-Tags von einem String

```
$var = "<br>test <b>was tut dieses</b><br>";
print $var . "<br>";
print htmlspecialchars($var) . "<br>";
print strip_tags($var) . "<br>";
```

Security

- Vom Nutzer gelieferte oder veränderte Pfadangaben validieren

```
<?php
  $basispfad = "/var/www/";
  $datei = $basispfad . $_GET['datei']; // <- "../../etc/passwd"

  print "Eingabe: " . $_GET['datei']  . "<br>";
  print "Datei: " . $datei . "<br>";
  print "Tatsachliche Datei: " . realpath($datei) . "<br>";
?>
```

```
Eingabe: ../../etc/passwd
Datei: /var/www/../../etc/passwd
Tatsachliche Datei: /etc/passwd
```

- UTF-8 "versteckte" Zeichen vom Nutzer

  utf8_decode( ) wandelt ISO-8859-1 Zeichen in einem UTF-8-codierten String in einbytige ASCII-Zeichen um

- Buffer-Overflows

  Wie in PERL ist es zur Laufzeit nicht möglich Speicher zu alloziieren, somit sind Buffer-Overflows in PHP-Code nicht möglich (aber in PHP selbst oder in Erweiterungen)

Assoziative Arrays als Return-Wert

- Assoziative Arrays sind als objektähnliche Datenspeicher und damit als Rückgabewert von Funktionen nützlich

```
<?php
    function mitglied_aus_datenbank($name) {
       /* ... Datenkbankfunktionen hier */

       $mitglied['name'] = $db['givenname'] . ' ' . $db['surname'] ;
       $mitglied['wohnort'] = $db['city'];
       $mitglied['status'] = $db['memberstatus'];

       return $mitglied;
    }
?>


<?  $user = mitglied_aus_datenbank($login); ?>

<table>
  <tr><td>Mitglied:</td><td><?=$login ?></td></tr>
  <tr><td>Status:</td><td><?=$user['status'] ?></td></tr>
  <tr><td>Name:</td><td><?=$user['name'] ?></td></tr>
  <tr><td>Wohnort:</td><td><?=$user['wohnort'] ?></td></tr>
</table>
```

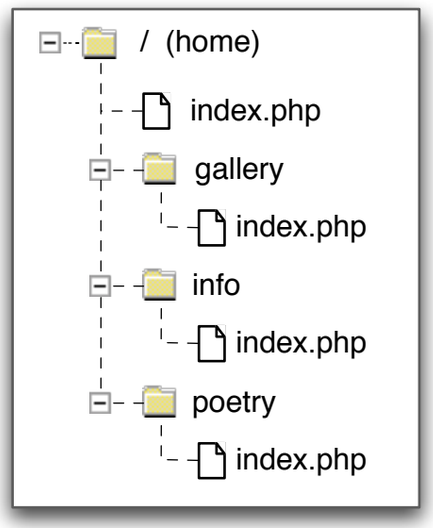
Multidimensionale Arrays

- Komplizierte Datenstrukturen (Objekte in Java, structs in C, Datenbanken können abgebildet werden)
- Arrays in Arrays

```
<?php
    $dokumente = array(
      'a101' => array(
            'author'   => 'Joerg Kinzebach',
            'format'   => 'word',
            'title'    => 'PHP Documentation',
            'filename' => 'a101.doc'
          ),
      'b174' => array(
            'author'   => 'Joerg Kinzebach',
            'format'   => 'text/plain',
            'title'    => 'Teilnehmerliste'
            'filename' => 'teilnehmer.txt'
          ),
      'c554' => array(
            'author'   => 'Joerg Kinzebach',
            'format'   => 'exel',
            'title'    => 'Abrechnung'
            'filename' => 'billing.xls'
          )
      );

      $dokumente['d731']['author'] = 'Erich Mustermann';
      $dokumente['d731']['format'] = 'text/plain';
      $dokumente['d731']['title'] = 'python howto';
      $dokumente['d731']['filename'] = 'pyhton.txt';
?>
```

5. Übung

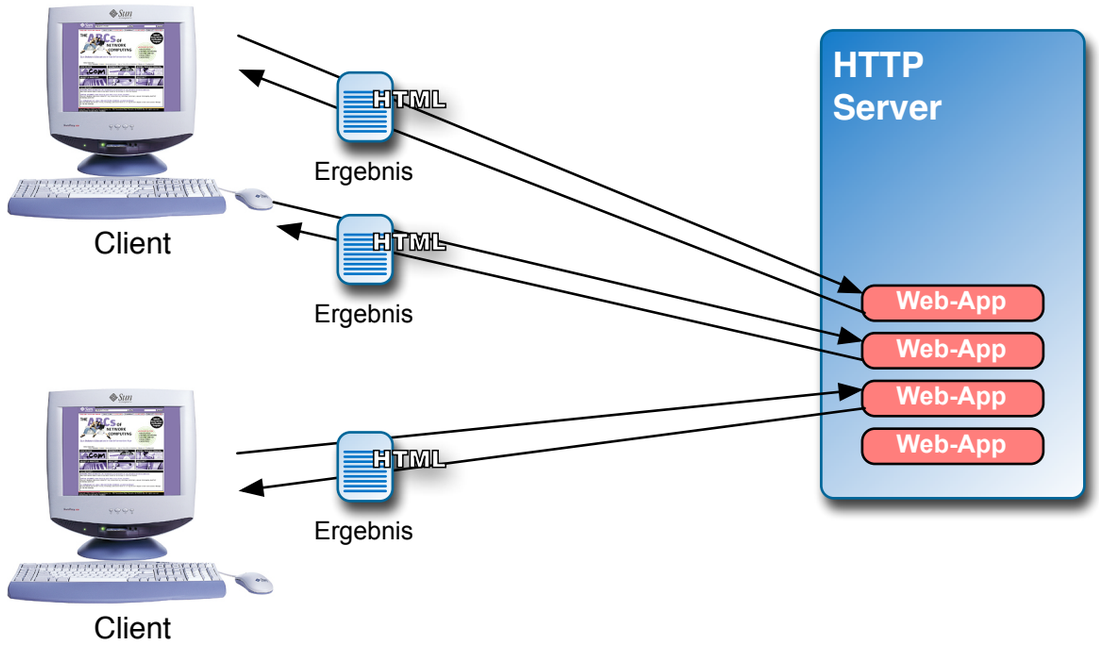
- Erstellen eines Skripts "menu.php", das von Webseiten eingebunden werden kann um ein Menu darzustellen.
- Das Menu soll in dem Skript zentral mittels eines assoziativen Arrays definiert werden.
- Als fortgeschrittene Version könnte der zur aktuelle Position passende Menu-Eintrag anders dargestellt werden um den Nutzer auf seinen Standort im Web-Baum aufmerksam zu machen



- Die Verzeichnisstruktur inkl. index.php ist bereits vorgegeben und angelegt, die Menupunkte sollen lauten "Home", "Gallery", "Info", "Poetry"
- Die Keys eines Menupunkts könnten z.B. "pfad" und "name" lauten
- Die Pfadangaben der Links sollten absolut ausgeführt werden, damit sie "von überall aus" funktionieren
- Es bietet sich an die Darstellung in eine Funktion auszulagern

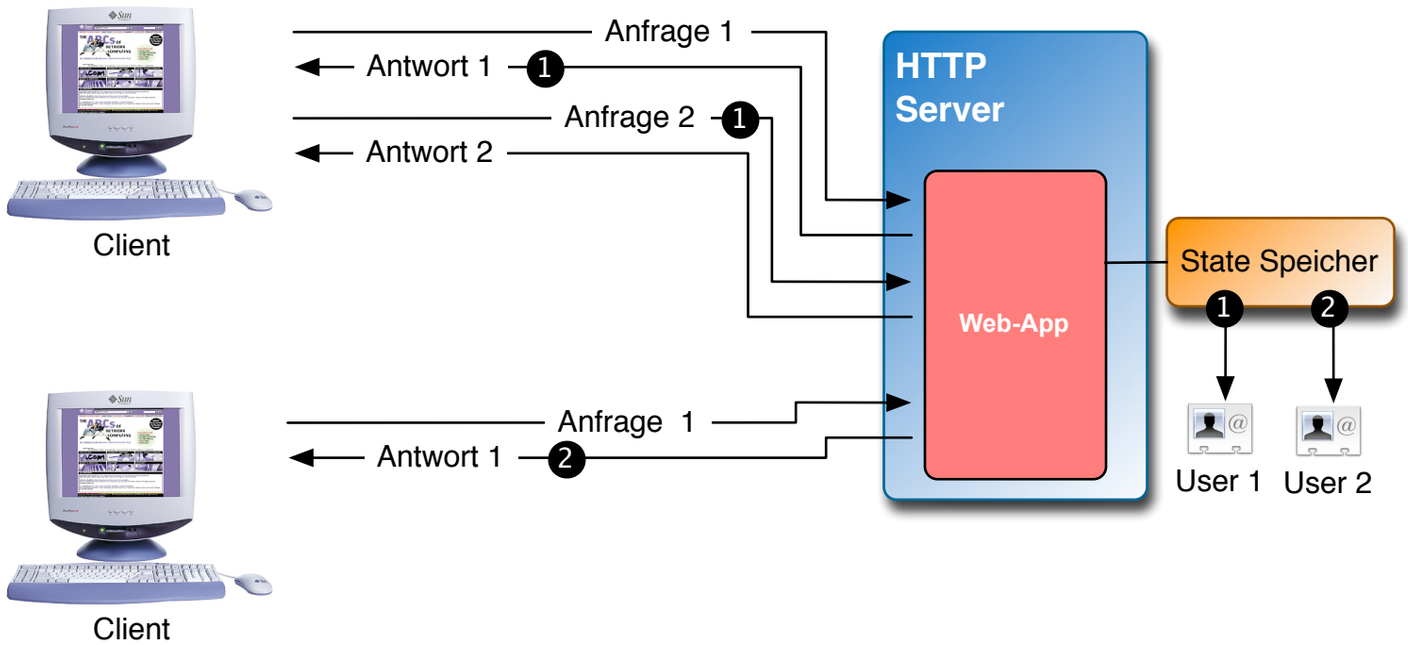
Cookies und Sessions

- Zugriffe mit dem HTTP-Protokoll sind "Stateless", der Server "Erstzugriffe" nicht von "Folgezugriffen" unterscheiden und daher keine Daten zwischen zwei Anfragen speichern.



- Cookies sind Daten, die der Server an den Client schickt und darum bittet, dass der Client diese Daten bei weiteren Anfragen wieder mit an den Server schickt.
- Cookies werden auf Domains beschränkt und können ein Verfallsdatum enthalten
- Daten in Cookies werden immer als Key = Value Paare gehalten

Cookies und Sessions



Cookies ermöglichen die Zuordnung von Anfragen zu persistentem State

- Clients mit Anfrage ohne gesetztem Cookie werden als "Erstanfrager" erachtet, für sie wird der State initialisiert und sie erhalten ein identifizierendes Cookie mit der Antwort
- Intern wird einem Cookie ein persistenter Datenspeicher zugewiesen
- Bei nächstem Request des Clients wird das Cookie mitgeschickt, anhand dessen der Anfrager identifiziert wird und sein State für die Bearbeitung der Anfrage wiederhergestellt werden kann

Beispiel für die Verwendung von Cookies

```
<?php
/*
 * Wenn noch kein Cookie "KNF_TEST" gesetzt ist, setze es mit der aktuellen
 * Uhrzeit (in Sekunden seit 1.1.1970), erlaube dem Client das Cookie nach
 * 86400 Sekunden (1 Tag) zu loeschen
 */
if (! isset($_COOKIE["KNF_TEST"])) {
        $wert = time();
        setcookie("KNF_TEST", $wert, time() + 86400);
}
?>
<html>
<head>
<title>Cookie Test</title>
</head>
<body>
<?
    // Wenn das KNF_TEST Cookie gefunden wurde, gebe das darin gespeicherte
    // Datum formatiert aus
    if (isset($_COOKIE["KNF_TEST"])) {
        print "Ihr Erstzugriff war am " . date("d.m.Y H:i:s",$_COOKIE["KNF_TEST"]);
    } else {
        print "dies ist Ihr erster Zugriff";
    }
?>
</body>
</html>
```

- Cookies gehören zu den "HTTP-Header"-Daten, diese können nicht mehr gesendet werden, wenn der Header abgeschlossen wurde und bereits Daten geschickt wurden

Sessions mit PHP

```
<?php session_start(); ?>
<html><head><title>Session Test</title></head>
<body>
  <form method="post" action="<?=$_SERVER['PHP_SELF']?>">
    <input type="text" name="daten">
    <input type="submit">
  </form>
<?
  if (isset($_REQUEST['daten'])) {
      print "Sie haben gerade \"" . htmlspecialchars( $_REQUEST['daten']) .
            "\" eingegeben.<br>";
  }
  $_SESSION["access" . time()] = $_REQUEST['daten'];
  print "Bisherige Eingaben:<br>";
  foreach ($_SESSION as $time => $eingabe) {
      if (substr($time,0,6) == "access") {
          $time = substr($time,6);
          print date("H:i:s", $time) . ": " . htmlspecialchars($eingabe) . "<br>";
      }
  }
?>
</body>
</html>
```

- PHP bietet Unterstützung für Sessions über Cookies und URL-Rewriting
- Bevor "Body"-Daten geschickt werden muss das Session-Handling mit session_start( ); aktiviert werden, danach werden Daten in dem globalen Array $_SESSION persistent gehalten

Datei-Upload

- File-Uploads werden in $_FILES kommuniziert, die Dateien werden in ein temporäres Verzeichnis geladen und müssen danach verschoben werden

```
<?php
if (!isset($_REQUEST['upload'])) {
  ?>
    <form enctype="multipart/form-data" action="upload.php" method="POST">
    <input type="hidden" name="MAX_FILE_SIZE" value="100000" />
    Datei auswählen: <input name="upload1" type="file" /><br />
    <input type="submit" name="upload" value="Upload starten" />
    </form>
  <?
} else {
    // In welches Verzeichnis soll der Upload verschoben werden?
    $zielverzeichnis = "uploads/";

    // nur Buchstaben, Ziffern, "-", "_" und "." im Dateinamen zulassen
    // und den Rest durch einen Unterstrich ersetzen
    $name = preg_replace('/[^a-z0-9_\-\.]/i', '_',
                         basename( $_FILES['upload1']['name']));

    $zieldatei = $zielverzeichnis . basename( $_FILES['upload1']['name']);

    if (move_uploaded_file($_FILES['upload1']['tmp_name'],
                           $zieldatei))
    {
        echo "Die Datei ".  basename( $_FILES['upload1']['name']) .
            " wurde auf dem Server gesichert.\n";
    } else{
        echo "Es gab ein Problem beim Datei-Upload, bitte erneut versuchen!\n";
    }
}
```

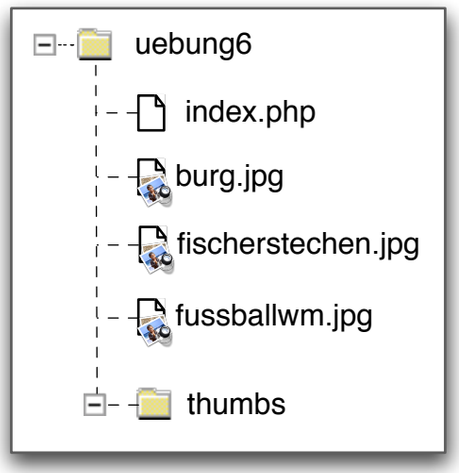
Verzeichnis-Operationen

- Zum Lesen eines Verzeichnisses wird ein Directory-Handle benötigt
- Von dem Handle können mittels readdir( ) einzelne Filenamen gelesen werden
- Mit closedir( ) wird das Handle wieder freigegeben
- Die gelesenen Dateinamen sind ohne Pfadangaben

```
// Ein "Directory Handle" für das Verzeichnis holen:
$dh = opendir($dir);

// Einen Dateinamen aus einem Verzeichnis holen:
$filename = readdir($dh);

// Ist die "Datei" ein Verzeichnis?
if (is_dir("${dir}/${file}")) {
    echo "DIR";
}

// Lesen des Verzeichnisses beenden
closedir($dir);

// Extension (.ext) einer Datei holen:
//    $filename wird in ein Array zerlegt indem der
//    Name bei jedem '.' aufgetrennt wird, danach
//    das letzte Element des Arrays holen, dies sollte
//    die File-Extension sein
$extension  = array_pop(explode('.', $filename));
```

6. Übung

- Inhaltslisting für ein Verzeichnis mit Bildern.
  - Für jede Datei soll die Größe angezeigt werden
  - Bilder sollen als sogenannte "Thumbnails" angzeigt werden
  - Thumbnails sollen nur einmal erzeugt werden und für weitere Zugriffe zwischengespeichert werden (Unterverzeichnis "thumbs")
  - Nach Möglichkeit soll die Bildgröße mit angezeigt werden




```
// Größe einer Datei ermitteln:
$size = filesize($filename);
```

```
// Beispiel-Code um ein Thumbnail erstellen:
$image = imagick_readimage("picture.jpg");
if ($image) {
   imagick_blur($image,1.4,0.9);
   imagick_sample($image,160,160,"160x160");
   imagick_writeimage($image,"thumbnail.jpg");
}
```