

# Information Systems

## Foundations, Third Class

---

### Contents

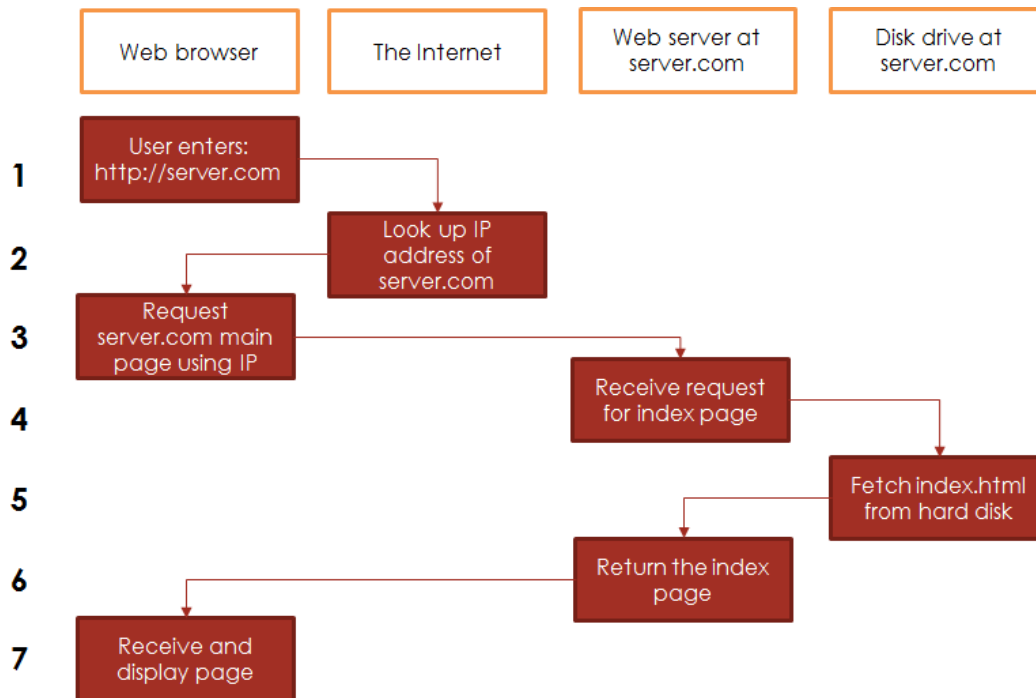
<b>1</b>	<b>Introduction to Dynamic Web Content</b>	<b>3</b>
1.1	The Request/Response Procedure . . . . .	3
1.1.1	The basic client/server request/response sequence . . . . .	3
1.1.2	The dynamic client/server request/response sequence . . . . .	4
<b>2</b>	<b>Hypertext Preprocessor (PHP)</b>	<b>5</b>
2.1	XAMPP . . . . .	5
2.1.1	Installing XAMPP . . . . .	5
2.1.2	Using XAMPP . . . . .	5
2.1.3	Including PHP . . . . .	5
2.1.4	Showing and Formatting Text . . . . .	6
2.1.5	Using Variables in PHP . . . . .	6
2.1.6	Using Functions in PHP . . . . .	6
2.1.7	Useful Links . . . . .	7
2.2	Variables and Operators . . . . .	8
2.3	Output and Strings . . . . .	10
2.3.1	Single Quote . . . . .	10
2.3.2	Double Quote . . . . .	10
2.3.3	heredoc . . . . .	11
2.3.4	String Functions . . . . .	11
2.4	Arrays . . . . .	13
2.4.1	Numeric array . . . . .	13
2.4.2	Associative array . . . . .	13
2.4.3	Multidimensional array . . . . .	14
2.4.4	Array functions . . . . .	14
2.5	Functions . . . . .	16
2.5.1	Call-by-value or call by reference . . . . .	16
2.5.2	The class DateTime . . . . .	16
2.6	Working with forms . . . . .	19
2.6.1	GET or POST . . . . .	19
2.6.2	Superglobals . . . . .	19
2.6.3	PHP self-request . . . . .	19
2.6.4	Cookies . . . . .	20

2.6.5	Session . . . . .	20
2.7	Working with files . . . . .	22
2.7.1	Files and folders . . . . .	22
2.7.2	Reading folders . . . . .	22
2.7.3	Reading and writing files . . . . .	22
2.7.4	Uploading files . . . . .	24
2.8	Task “Casting” . . . . .	26
2.9	Task ”Codebreaker” with PHP . . . . .	27
2.10	Object-Oriented Programming with PHP . . . . .	28
<b>3</b>	<b>Task ”Ars Electronica Center”</b>	<b>32</b>
<b>4</b>	<b>MySQL with XAMPP</b>	<b>33</b>
4.1	Installation . . . . .	33
4.1.1	Website . . . . .	33
4.1.2	XAMPP Distribution . . . . .	33
4.2	First Statements . . . . .	33
<b>5</b>	<b>PHP and MySQL</b>	<b>34</b>

# 1 Introduction to Dynamic Web Content

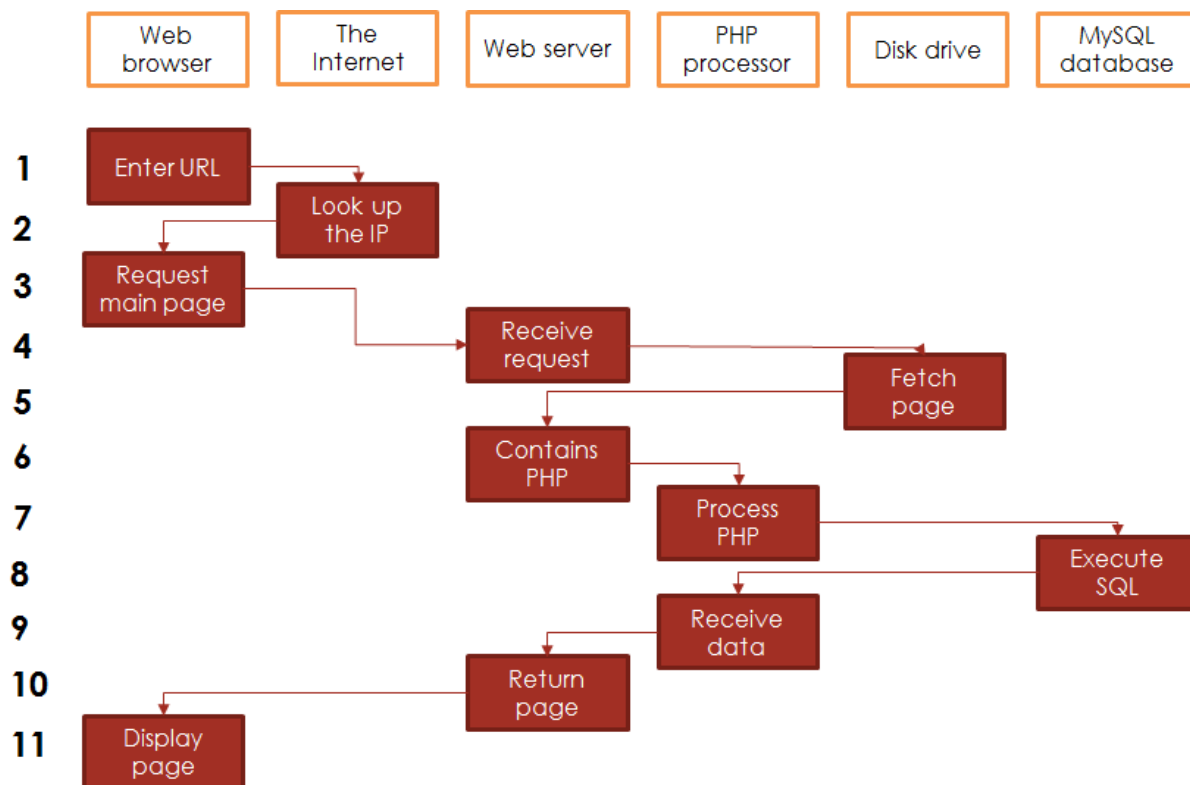
## 1.1 The Request/Response Procedure

### 1.1.1 The basic client/server request/response sequence



1. The user enters `http://server.com` into the browser's address bar.
2. The user's browser looks up the IP address of `server.com`.
3. The browser requests the main Web site at `server.com`.
4. The Web server at `server.com` receives the request for the index page.
5. The Web server fetches the Web page on its hard disk.
6. The Web page is retrieved by the server and returned to the browser.
7. The user's browser displays the Web page.

### 1.1.2 The dynamic client/server request/response sequence



1. The user enters *http://server.com* into the browser's address bar.
2. The user's browser looks up the IP address of *server.com*.
3. The browser requests the main Web site at *server.com*.
4. The Web server at *server.com* receives the request for the index page.
5. The Web server fetches the Web page on its hard disk.
6. Having the Web page in memory, the Web server realizes that it is a file incorporating PHP scripting and handles the page over to the PHP interpreter.
7. The PHP interpreter executes the PHP sourcecode.
8. Some of the PHP contains MySQL statements, which the PHP interpreter handles over to the MySQL database engine.
9. The MySQL database executes the SQL statements and returns the results back to the PHP interpreter.
10. The PHP interpreter returns the results of the executed PHP sourcecode and the results from the MySQL database, to the Web server.
11. The Web server returns the page to the requesting client (browser), which displays it.

## 2 Hypertext Preprocessor (PHP)

PHP has been developed since 1994. It is open source and free of charge. For test purposes, you will have to install a Web server on your machine, because a PHP-interpreter is requested for its successful usage.

### 2.1 XAMPP

You find XAMPP on following Web site: <http://www.apachefriends.org/en/xampp.html>.

XAMPP is a package of all those technologies, which are requested in case of server-side Web site development. XAMPP is short for Webserver **A**pache with the database **M**ySQL / SQLite and the scripting languages **P**erl und **P**HP; the **X** stands for different platforms. The big advantage of XAMPP is that you have your own Web server on your machine and that is suitable for learning and experimenting with dynamic Web sites.

#### 2.1.1 Installing XAMPP

**Exercise.** Download and install the product XAMPP from following Web site: <http://www.xampp.org>.

#### 2.1.2 Using XAMPP

- After installation, open the file *xampp-control.exe*. While working do not close the control panel.
- Start the Apache web server by clicking the corresponding Start-Button.
- Open a Web browser and enter *localhost*. The welcome page will be presented. Here, you can inform you about the options of the web server and test some PHP-demo programs.
- The working directory of the web server is *htdocs*. Use this directory for working sub-directories, for instance, *php-test*. When opening a php-file in the web browser, enter *localhost/php-test/filename.php*.
- Stop the web server by clicking the Exit-Button and close the control panel.

**Exercise.** Surf on the Web and find out, which databases PHP supports.

#### 2.1.3 Including PHP

You can use PHP directly with HTML. If you are developing dynamic Web sites with PHP, use the file extension *.php* for your HTML files.

**Exercise.** Develop your first php-file *myfirst.php*. Use following script:



```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3 <html>
4 <head>
5     <title>I am learning PHP.</title>
6 </head>
7 <body>
8 <?php
9     echo "Hello World! This is my first PHP-Script.!";
10 ?>
11 </body>
12 </html>
```

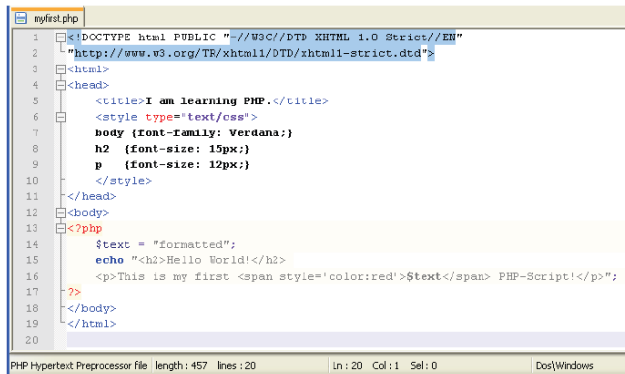
*Remark.* Meaning:

- Line 8 to 10: A PHP-Script starts with *<?php* and ends with *?>*. In an HTML file you can use as many scripts as you want.
- Line 9: The statement *echo* has the effect that the text within the quotation marks is presented. Instead of using quotation marks you can use the statement *print()*. Each PHP-statement has to be completed with a semicolon (;).

### 2.1.4 Showing and Formatting Text

As already mentioned you can either show text with *echo* or *print()*. If you want to format text you will have to use HTML and CSS.

**Exercise.** Format text with CSS. Use following script:



```
1 <?DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3 <html>
4 <head>
5 <title>I am learning PHP.</title>
6 <style type="text/css">
7 body {font-family: Verdana;}
8 h2 {font-size: 15px;}
9 p {font-size: 12px;}
10 </style>
11 </head>
12 <body>
13 <?php
14 $text = "formatted";
15 echo "<h2>Hello World!</h2>";
16 <p>This is my first <span style='color:red'>$text</span> PHP-Script!</p>;
17 <?>
18 </body>
19 </html>
20
```

*Remark.* Meaning:

- Line 14 to 16: Each variable has to start with \$. When using echo you can mix text and variables. To connect strings you can use the point operator: "This is a ".\$variablename." text."
- Line 16: Do not forget to use apostrophe in CSS. If you use quotation marks in the text you will have to indicate it with the backslash (\).

### 2.1.5 Using Variables in PHP

Consider following rules for the declaration of variables:

- First sign: \$
- Letters and numerics
- No umlauts, blanks, symbols
- Distinction between upper-case and lower-case characters

Examples for variable names and types:

- \$size = 1.70;
- \$age = 25;
- \$location = "Paris";
- \$student = true;

### 2.1.6 Using Functions in PHP

When declaring functions in PHP, you should use following syntax:

- <?php function functionName() { scripting code to be executed; } ?>

When calling functions in PHP, you should use following syntax:

- <?php phpinfo(); ?>

There are some specific functions, which allow to keep scripting code in external files:

- `<?php include 'myfunctions.php'; ?>`
- `<?php require 'myfunctions.php'; ?>`
- `<?php require ('myfunctions.txt') ; ?>`

Include and require are identical, there is just one difference: require will produce a fatal error (E\_COMPILE\_ERROR) and stop the script, whereby include will only produce a warning (E\_WARNING) and the script will continue.

**Exercise.** Write a PHP-script that uses these different functions. Declare functions and variables in an external file, e.g. 'myfunctions.php', and use it in your PHP-script. Write respectively a single function for the header, the main part and the footer for a simple Web site introducing yourself. All functions shall be available in 'myfunctions.php'. Use variables for your data. Use one common CSS for the whole Web site.

### 2.1.7 Useful Links

<http://www.w3schools.com/php/default.asp>

<http://www.selfphp.info/>

<http://www.php.net/>

## 2.2 Variables and Operators

PHP is a loosely typed language. That means that a variable does not have to be declared before a value is added to it. PHP automatically converts the variable to the correct data type during runtime. Consider that this fact is a possible source of errors.

PHP supports the following data types:

- Integer - used for whole numbers
- Float - used for real numbers
- String - used for strings of characters
- Boolean - used for true or false
- Array - used for storing multiple data items
- Object - used for storing instances of classes

PHP allows defining constant values. You can assign a constant value to a variable by using the function `define()`:

```
define("Pi", "3.1415"); echo(Pi);
```

How to set and get data types of variables - an example:

```
<?php
define("lf","<br />"); //defines lf as line feed
define("Pi","3.1415"); //defines Pi as constant value
echo(gettype(Pi) .lf); //defines once again the data type of Pi (string)
$a = 1;
$b = 2;
echo("Type a = " .gettype($a) .lf); //of which data types are a and b?
echo("Type b = " .gettype($b) .lf); //both variables are recognized as integer
$a = 1.0;
$b = 2.0;
echo("Type a = " .gettype($a) .lf); //of which data types are a and b?
echo("Type b = " .gettype($b) .lf); //both variables are recognized as double
$pi = $b*asin($a);
$zahl = Pi;
settype($zahl,"double"); //defines the data type of $zahl as double - this is an explicit type cast
echo("Type of $zahl = " .gettype($zahl) .lf);
$diff = $pi - $zahl;
echo("difference = " .gettype($diff) ." value = " . $diff);
//you combine strings by using the concatenation-operator within echo
?>
```

There are six basic scope rules in PHP, whereby the term scope defines where a variable is visible in a PHP script:

- Constants - once declared - are always visible globally.
- Global variables are visible in the whole script but not inside functions (except they are declared as global in the function - see next point).
- Variables inside functions, which are declared as global (keyword `global`), refer to the global variables with the same name.
- Variables defined inside functions and declared as static are invisible outside the function but keep their value when calling this function the next time.
- Variables defined inside functions are only visible inside these functions.
- Built-in superglobal variables (arrays `$_GET`, `$_POST`, ...) are visible in the whole script.



Operators are used similar to other programming languages. We know:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Bit-operators
- Cast-operators

Statements are composed of variables, function calls and operators, which rule their combinations. You can influence the order of execution by using parenthesis. In the following you will find a script with some calculations.

```
<?php
define("lf","<br />");
echo ('<h3>Chapter <em>calculations.php</em></h3>');
echo('<pre>');
$a = 1.0;
$b=2.0;
$pi = $b*asin($a);
echo ('Calculate pi = ' . $b . ' * asin(' . $a . ') = ' . $pi);
$r=1.0;
$u = $b * $r * $pi;
$f = $r * $r * $pi;
echo(lf . 'Radius : ' . $r);
echo(lf . 'Circumference 2pi*r : ' . $u);
echo(lf . 'Circular area pi*r*r: ' . $f);
echo('</pre>');
?>
```

**Exercise.** Get used to variables and operators. Implement the proposed examples and familiarize you with the syntax.

Then, write a PHP-script that presents following true-or-false-table (truth table) to the user. Use Boolean variables for producing this truth table.

#### Truth Table in PHP

AND	true	false
true	true	false
false	false	false

OR	true	false
true	true	true
false	true	false

## 2.3 Output and Strings

You can produce output either with `echo` or `print()`.

You can format text by using `printf()`. Most important format specifications are **s** for strings, **f** for float and **d** for integer.

```
<?php
    $pi = 3.1415;
    $circle = 180.0;
    $rho = $circle/$pi;
    $i = 24;
    echo ('Constant values pi = ' . $pi . ' rho = ' . $rho . '<br/>');
    print "Constant values pi = $pi rho = $rho <br/>";
    printf('<pre>Constant values pi = %4.2f <br/> rho = %15.2f i = %10d</pre>', $pi, $rho, $i);
?>
```

A useful method to produce output is to combine several strings with the help of one variable and to use the function `echo`.

```
<?php
    $ouput = '<html><head><title>Test PHP Echo-Function</title></head>';
    $ouput .= '<body><h1>Headline</h1>';
    $ouput .= '<p>Text</p>';
    $ouput .= '</body></html>';
    echo ($ouput);
?>
```

You can use HTML-Code as well as the special characters `<` and `>` in strings without any problems. You can specify a string by using following methods:

- single quote
- double quote
- heredoc syntax

### 2.3.1 Single Quote

The simplest form to produce strings is to use single quotes. In this case variables are considered and executed as simple string as well.

### 2.3.2 Double Quote

When using double quotes, PHP is able to recognize and execute variables. Besides, PHP recognizes escape sequences for specific characters:

Character sequence	Description
<code>\n</code>	linefeed
<code>\r</code>	carriage return
<code>\t</code>	horizontal tab
<code>\\</code>	backslash
<code>\'</code>	single quote
<code>\"</code>	double quote
<code>\[0-7]1,3</code>	the sequence of characters matching the regular expression is a character in octal notation
<code>\x[0-9A-Fa-f]1,2</code>	the sequence of characters matching the regular expression is a character in hexadecimal notation

### 2.3.3 heredoc

Another possibility to delimit strings is the **heredoc** syntax (<<<).

```
<?php
$str = <<<EOD
Example of string s
panning multiple lines
using heredoc syntax.
EOD;
echo $str;
?>
```

**Exercise.** Get used to the three different possibilities for displaying strings. Implement the proposed examples, get used to the syntax and highlight differences. Show, which escape-sequences PHP supports and make an output with **heredoc**.

### 2.3.4 String Functions

PHP supports several functions to work with strings. In the following, some examples are introduced. For more information have a look at, for instance, <http://www.php.net>.

#### **strlen**

```
int strlen(str);
```

Returns the length of a string.

#### **strcmp**

```
int strcmp(str1, str2);
```

Compares two strings (case-sensitive) and returns following values:

- < 0 if str1 is smaller than str2
- 0 if both strings are equal
- > 0 if str1 is bigger than str2

#### **substr**

```
string substr(source, start, [length]);
```

Returns a substring dependent on the requested start position (if there is a negative value for start, the returned string starts from the character that is situated at the end of the string minus start). The length parameter is optional. In case that no length is presented the sub string will be returned til the end of the original string.

#### **strpos**

```
int strpos(source, find, start);
```

Returns the first time a character (case-sensitive) or string is found in another string.

#### **trim**

```
trim(source, [charlist]);
```

Removes whitespaces and other predefined characters from both sides of a string.

#### **strtolower, strtoupper**

- `string strtolower(source);`
- `string strtoupper(source);`

Convert all characters of a string either to lowercase or uppercase.

**sprintf**

```
string sprintf(format, arg1, [arg2], [arg++]);
```

Formats strings corresponding to the particular parameters. Works as in C.

**str\_replace**

```
string str_replace(find, replace, string, [count]);
```

Replaces all searched characters in another string.

**explode**

```
string explode(separator, string, [limit]);
```

Allows to split of a string by using pre-defined separators.

**htmlspecialchars - htmlentities**

- **htmlspecialchars** - Converts some predefined characters to HTML code.
- **htmlentities** - Converts characters to HTML entities.

**Exercise.** Get used to the introduced string functions. Find an example for each string function and get used to the syntax.

1. Store the string Francis\' bike into a variable. Display the string without the backslash by using the function **stripslashes()**.
2. Store the string <h2> into a variable. Convert the HTML tags into HTML entities by using **htmlspecialchars**. Have a look at the differences in the output.
3. Can you use both functions **htmlspecialchars** and **stripslashes** in one single statement?
4. Write a statement by using **strstr()**. The statement shall replace all uppercase letters to lowercase letters in a string.
5. Following string is given: Hat der alte Hexenmeister \n sich doch einmal wegbegeben! \nUnd nun sollen seine Geister \nauch nach meinem Willen leben! \nSeine Wort' und Werke \nmerkt' ich, und den Brauch, \nund mit Geistesstärke \ntu ich Wunder auch.  
Use the function **n12br()** to display the line breaks in correct HTML code.

**Exercise. Anagram**

Two words are anagrams if they consist of the same letters in any other order (e.g dt. Maus and Saum; cloud and could). Write a method is Anagram(s1, s2) that checks if the two strings s1 and s2 are anagrams. Return the result as boolean. Upper and lower case can be ignored.

## 2.4 Arrays

Arrays are storage sections, in which you can address several values with a single variable. An array consists of elements, which are combinations of key-value pairs. The key has to be unique and may be a string or a number. The array-element may be a string, boolean, number or an array. Mathematical spoken a one-dimensional array is a vector and a two-dimensional array is a matrix. You can address elements by using row- and column indices.

We consider numeric, associative and multidimensional arrays.

### 2.4.1 Numeric array

There are three different techniques when declaring arrays:

```
$personen[] = "Simon";  
$personen[] = "Charly";  
$personen[] = "Hector";
```

```
$data = array();  
$data[] = "First entry";  
$data[] = "Second entry";
```

```
$personen = array("Simon", "Charly", "Hector");
```

Let us start with a simple example. All keys are produced automatically with sorted numbers. All values are strings.

```
<?php  
$player = array("Neuer", "Hummels", "Martinez", "Ribéry", "Mueller",  
               "Rodriguez", "Boateng", "Lewandowski",  
               "Robben", "Coman", "Wagner");  
  
sort($player);  
$sum = count($player);  
for ($i=0; $i<$sum; $i++) {  
    echo("Nr.: $i $player[$i]<br/>");  
}  
?>
```

Alternatively, you can use `foreach()` in combination with `$key` and `$wert`.

```
<?php  
echo ('<table border=1 cellspacing=3 cellpadding=3>');  
foreach ($player as $key => $wert) {  
    echo('<tr><td>' . $key . '</td><td>' . $wert . '</td></tr>');  
}  
echo ('</table>');  
?>
```

### 2.4.2 Associative array

When using associative arrays or hashes you address array-elements with the help of strings as keys. Have a look at several examples

```
$personen["P1"] = "Simon";  
$personen["P2"] = "Charly";  
$personen["P3"] = "Hector";  
//Output  
echo $personen["P1"] . "<br/>";  
echo $personen["P2"] . "<br/>";  
echo $personen["P3"] . "<br/>";
```

```
$personen = array(
    "P1"⇒"Simon",
    "P2"⇒"Charly",
    "P3"⇒"Hector");
//Output
echo $personen["P1"] . "<br/>";
echo $personen["P2"] . "<br/>";
echo $personen["P3"] . "<br/>";
```

Some more specific examples:

```
$personen = array(1⇒"Simon","Charly","Hector");
//Output
echo $personen[1];
```

```
$personen = array(10⇒"Simon","Charly","Hector");
//Output
echo $personen[10];
echo $personen[11];
echo $personen[12];
```

```
$personen = array(10⇒"Simon","Charly",17⇒"Hector");
//Output
echo $personen[10];
echo $personen[11];
echo $personen[17];
```

```
$personen = array(-10⇒"Simon","Charly",17⇒"Hector");
//Output
echo $personen[-10];
echo $personen[0];
echo $personen[17];
```

If you want to delete arrays just use `unset()`.

### 2.4.3 Multidimensional array

In multidimensional arrays you can specify not only simple datatypes (numbers, boolean, strings) as elements but more complicated ones such as objects or arrays.

```
<?php
$dvd = array();
$dvd[] = array('title'⇒"Ocean's Eleven",'artist'⇒"George Clooney",'year'⇒2001);
$dvd[] = array('title'⇒"Mission Impossible",'artist'⇒"Tom Cruise",'year'⇒2009);
$dvd[] = array('title'⇒"Chocolate",'artist'⇒"Johnny Depp",'year'⇒2008);
$dvd[] = array('title'⇒"Ice Age",'year'⇒2002); /* has no index 'artist' */
print_r($dvd);
?><br/><br/>
<?php
echo $dvd[0]['artist'];
?>
```

### 2.4.4 Array functions

There are many pre-defined array functions such as `count()`, `unset()`, `array_key_exists()`, `in_array()`, `array_search()`. For more detail have a look at <http://php.net/manual/de/ref.array.php>.

#### **implode()**

string implode(\$separator , \$array)

A very useful function is `implode()`, which allows converting arrays into strings. The array elements are combined by using the `$separator` sign.

```
echo '<h2>implode()</h2>';
$names = array("Max", "Lisa", "Ulli");
echo "Separate names using semicolon: <br>";
$nameStr = implode("; ", $names);
echo $nameStr;
echo "<br><br>";
echo "One name in one row: <br>";
echo implode("<br>", $names);
echo '<h2>explode()</h2>';
```

### **explode()**

```
array explode($separator, $str)
```

The function `explode()` allows converting strings to arrays. The string will be separated by the `$separator` sign. So, this sign determines the elements.

```
echo '<h2>explode()</h2>';
$text = "Max,Lisa,Ulli";
$names = explode(",", $text); //Convert a string to an array
echo "<pre>";
var_dump($names); //Delivers infos about the array
echo "</pre>";
//exchange an element of the array
$names[1] = "Helena";
//convert the array back to a string
$text = implode(", ", $names);
echo $text;
```

**Exercise.** Get used to arrays.

1. Compose an associative array with five elements. Use as index the car reference key and as value the corresponding town, e.g. AM -> Amstetten. Show all elements of the array and especially this for the index "ME".
2. Display the array from 1. with following orders: `sort`, `rsort`, `ksort`.
3. Implement an associative array called `playernumber` for the players from chapter 2.4.1. The key of the array-element represents the player's name, e.g. Neuer⇒1.
  - (a) Display all players with name and player number.
  - (b) Display all players with player number and name ordered by player number (ascending).
  - (c) Player Mueller should leave the team. Instead player Alaba enters the team. Alaba has player number 27. Show the right exchange of the players by displaying the new team with the help of an HTML-table: first column -> player's number; second column -> player's name.
4. Implement a script, which proposes numbers for lottery (6 from 45). The script should choose and display numbers from 1 to 45. Use array functions.

## 2.5 Functions

You call a function by using the function's name followed by the ordered list of parameters. You declare a function by using the keyword `function`. The result of a function is defined by using the keyword `return`.

We start with a simple example.

```
<?php
    $w = 3.0; $h = 4.0; $d = 5.0;
    $room = capacity($w, $h, $d);
    echo ("Volume: $room");
    function capacity ($length, $width, $height) {
        $volume = $length * $width * $height;
        return $volume;
    }
?>
```

### 2.5.1 Call-by-value or call by reference

To exchange data via functions you can either use call-by-value or call by reference. If you use the operator `&` in the parameter list, changes will be made to the corresponding variable in the function as well as in the calling programm.

An example with call by reference:

```
<?php
    $w = 3.0; $h = 4.0; $d = 5.0;
    $room = 0.0;
    capacity($w, $h, $d, $room);
    echo ("Volume: $room");
    function capacity ($length, $width, $height, &$amp;volume) {
        $volume = $length * $width * $height;
        return;
    }
?>
```

Another example by using an array:

```
<?php
    $cuboid = array(3.0, 4.0, 5.0, 0.0);
    capacity($cuboid);
    echo ("Volume: $cuboid[3]");
    function capacity (&$q) {
        $q[3] = $q[0] * $q[1] * $q[2];
        return;
    }
?>
```

### 2.5.2 The class DateTime

Since PHP5 there is the class `DateTime`, which offers different date and time functions. It is possible to add and subtract timestamps. In the following, we present an example that answers the question: How many days will have to be passed until 24th December 20xx?

```
<?php
    $today = new DateTime();
    $inthefuture = new DateTime('2020-12-24');
    $daysto = new DateInterval('P1D'); /* P periods 1 day */
    $daysto = $inthefuture->diff($today); /* method diff() delivers the difference between $inthefuture
and $today as DateInterval object */
    echo ('Today: ' . $today->format('Y-m-d') . '<br/>');
    echo ('until: ' . $inthefuture->format('Y-m-d') . '<br/>');
    echo ('will be ' . $daysto->format('%Y year(s) %m month(s) and %d day(s)') . '<br/>');
?>
```



With the function `mktime()` you can produce UNIX-timestamps. Consider the order of the arguments: `mktime(hour, minute, second, month, day, year)`. Besides, `mktime()` will check if the arguments deliver a valid date. This is important in case of user entries.

```
<?php
$timestamp = mktime(0,0,0,12,12,2013);
echo $timestamp; /* will deliver 1386802800 */
?>
```

**Timestamp formats** To make timestamps more readable you can use the function `date()`.

```
<?php
echo date('l jS \of F Y h:i:s A');
echo date(DATE_RFC822);
?>
```

Format Sign	Description	Example
d	Day of the month, two-digit with leading zero	01 to 31
D	Day of the week with three letters	Mon to Sun
F	Month as whole word, such as January, ...	January to December
g	12-hours-format, without leading zero	1 to 12
G	24-hours-format, without leading zero	0 to 23
h	12-hours-format, with leading zero	01 to 12
H	24-hours-format, with leading zero	00 to 23
i	Minutes with leading zero	00 to 59
j	Day of the month without leading zero	1 to 31
l (small 'L')	Day of the week in words	Sunday to Saturday
L	Leap year or not	1 for leap year, else 0
m	Month as digit with leading zero	01 to 12
M	Month as word with three letters	Jan to Dec
n	Month as digit without leading zero	1 to 12
s	Seconds with leading zero	00 to 59
t	Number of days for the given month	28 to 31
w	Numeric value for the day in a week	0 (for Sunday) to 6 (for Saturday)
W	ISO-8601 Week number of the year, the week starts on Monday	e.g. 42 (the 42. week in the year)
Y	Four-digit year	e.g. 2020
y	Two-digit year	e.g. 12
z	The day of the year	0 to 365

**Exercise.** Get used to functions and the class `DateTime`.

- Implement functions for following calculations:
  - Calculation of surface area for square, rectangle, circle and triangle.
  - Calculation of volume for cone, pyramid and cube.
  - Calculation of velocity: e.g. a crane raises weight in 24 seconds 32 meters high. Calculate the middle velocity in m/min.
- Implement a simple calculator for the calculations of two values: addition, subtraction, multiplication, division and raising a power. Use functions for the calculations.
- Implement a script that formats the current timestamp as follows.

04	Day of the month, two-digit with leading zero
Sun	Day of the week with three letters
November	Month as whole word
10	12-hours-format, without leading zero
10	24-hours-format, without leading zero
10	12-hours-format, with leading zero
10	24-hours-format, with leading zero
47	Minutes with leading zero
5	Day of the month without leading zero
Monday	Day of the week in words
1	Leap year or not
11	Month as digit with leading zero
Nov	Month as word with three letters
43	Seconds with leading zero
30	Number of days for the given month
0	Numeric value for the day in a week
44	ISO-6801 Week number of the year, the week starts on Monday
12	Four-digit year
307	The day of the year

4. Implement a script that displays the calendar for the month December 20xx. Display the weekday, the date and the kind of holiday, e.g. Monday, 24.12.20xx, Christmas. Give sundays and holidays a special format.
5. Change the script of the calendar by using German names. The weekday and month should be displayed in German, e.g. Montag, 24. Dezember 20xx, Weihnachten.

## 2.6 Working with forms

Forms are a great tool for communicating with users via interactive and dynamic web sites. The user can enter data and request responses from the web server in different situations.

### 2.6.1 GET or POST

For transferring data there are two possibilities: either GET or POST. If you use GET the values of formular fields will be part of the URL. Therefore, they are visible for the user. If you use POST the values of formular fields will be embedded into the message itself. So, the advantage is that you can transfer more data with POST than with GET and the values are invisible for the user.

```
<html>
<head><title>htmlformular.html</title></head>
<body>
<p>Enter username and password</p>
<form name="enter" action="formdata.php" method="POST">
<input type="text" name="user" value="" length="8" maxlength="24"/>
<input type="password" name="pw" value="" length="8" maxlength="24"/>
<input type="submit" name="go" value="Send It!"/>
</form></body></html>
```

**Aufgabe.** Implement this example by using method POST and using method GET. Describe the difference!

### 2.6.2 Superglobals

There are different superglobals, which can be used everywhere within PHP and called in each script. At the moment, there are \$\_POST, \$\_GET or \$\_REQUEST for evaluating HTML formular values.

```
<?php
$user = trim($_POST['user']);
$password = trim($_POST['pw']);
echo ('Username: ' . $user . '<br/>');
echo ('Password: ' . $password . '<br/>');
?>
```

**Aufgabe.** Implement this example by using supergloblas \$\_POST, \$\_GET or \$\_REQUEST. Describe the differences!

### 2.6.3 PHP self-request

It is possible to host the HTML web site and the PHP script in one single file. That means that when the user clicks in the HTML file on the submit button (= sends the file to the web server) the file is interpreted on the web server through the PHP script as well. So, you have to decide either the user has to see the HTML form or the response from the web server.

```
<?php
if (isset($_POST['go'])) {
// ... response of the web server requested
} else { ?>
<form name="enter" action="<?php echo $_SERVER['PHP_SELF'] ?>" method="POST">
<input type="text" name="user" value="" length="8" maxlength="24"/>
<input type="password" name="pw" value="" length="8" maxlength="24"/>
<input type="submit" name="go" value="Send It!"/>
</form>
<?php } // end of the else-block
?>
```

**Aufgabe.** Get used to working with forms.

1. Implement this script by giving the response from the web server to the client. You should check if the user has entered correctly his username and password. That means: has the user entered username and password at all and is in both cases the word length between 8 and 24 signs. If the formular is not correct the user will get a corresponding answer and will get another try. In the other case the user should get a welcome page with his username and password. Use the function `strip_tags()` to read the superglobal `$_POST`. This function eliminates HTML-tags.
2. Create a PHP program that registers a marathon runner for the Linz marathon event in April. The interface should accept all necessary personal information (name, address, gender, age, and T-shirt size). Whenever possible, use HTML objects that restrict input (e.g. select object for T-shirt size). Validate all information in the interface tier using both HTML5 and JavaScript. If the information is valid, pass the information to the server (business rule tier). Once all information has been accepted, the program will display the cost of entering the marathon (€ 50). Any shirts over XL will add an additional charge of €5. Any marathon runner 65 or older will be charged €10 less.

#### 2.6.4 Cookies

You can work with cookies in PHP. You can create, read or delete cookies. Cookies are a simple means for storing or retrieving information from the client.

You can use `strip_tags()` for defining a cookie that is transferred with the HTTP header information. You have to send the cookie before you send any other information to the client.

```
<?php
if (!isset($_COOKIE['TestCookie'])) {
    $value = 1;
    setcookie("TestCookie", $value);
    echo "new value: $value";
} else {
    $value = $_COOKIE['TestCookie'];
    if ($value >10)
        setcookie("TestCookie", "", time()-3600); /* cookie's life time shall be 1 hour */
    else
        setcookie("TestCookie", $value+1); /* write cookie with new value */
    echo "old value: " . $_COOKIE['TestCookie'];
} // end of the else-block
?>
```

#### 2.6.5 Session

The Session-Management is a powerful tool in PHP. It enables to keep the relationship between the web server and a particular web client over a longer time, e.g. for realising a shopping basket.

The Session-Management handles automatically the creation of session ids and the storage of the corresponding values.

Registration of a variable

```
<?php
session_start();
if (!isset($_SESSION['counter'])) {
    $_SESSION['counter'] = 0;
} else {
    $_SESSION['counter']++;
}
echo $_SESSION['counter'];
?>
```

Deletion of the variable

```
<?php
session_start();
unset($_SESSION['counter']);
?>
```

Redirect to another page

```
<?php session_start(); ?>
<?php if (isset($_POST['go'])) {
if (isset($_POST['myname'])) {
$_SESSION['myname'] = $_POST['myname'];
$redirect_url = 'http://localhost:8080/forms/answer.php';
header("Location: $redirect_url");
} } else { ?>
<form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="POST">
Name: <input type="text" name="myname" value="">
<br> Username: <input type="text" name="username" value="">
<br> Password: <input type="password" name="pwd" value="">
<br> <input type="submit" name="go" value="Register">
</form>
<?php } ?>
```

## 2.7 Working with files

PHP allows you getting access to the file system of a Web server. You can write, read or delete files.

### 2.7.1 Files and folders

To work with files and folders you have to know their names and structures. Therefore, you are provided with a set of pre-defined functions in PHP that offer you access to this information.

In the following, we illustrate a simple example:

```
<?php
// ( True or False )
echo file_exists ("C:/xampp/php/php.exe") . "<br/>";
// Size
echo filesize ("C:/xampp/php/php.exe") . "<br/>";
// Type
echo filetype ("C:/xampp/php/php.exe") . "<br/>";
echo is_dir ("C:/xampp/php") . "<br/>";
echo is_executable ("C:/xampp/php/php.exe") . "<br/>";
echo is_file ("C:/xampp/php/php.exe") . "<br/>";
if ( chdir ("C:/xampp")) {
    echo " Verzeichnis gewechselt !";
}
echo mkdir ("C:/xampp/neu" ,0700);
?>
```

### 2.7.2 Reading folders

Important to know is the structure of a folder:

```
<?php
$handle = opendir('.');
echo " Verzeichnis - Handle : $handle <br>";
echo " Dateien :<br>";
while ( false !== ( $file = readdir ( $handle ))) {
    echo"$file <br>";
}
closedir ( $handle );
?>
```

### 2.7.3 Reading and writing files

Working with files in PHP is similar to working with files in C.

The access to a file follows a particular schema:

- Open the file for reading or writing.
- Read content (read), re-write content (write) or append content (append).
- Close the file.

The corresponding PHP-code is as follows:

- Open the file: `$file = fopen("test.txt","r");`

Parameters:

- `r` access for reading, file must exist
- `r+` access for reading and writing
- `w` access for writing, the file will be overwritten, the file will be created if it does not exist
- `w+` access for reading and writing, content will be overwritten
- `a` access for writing, content will be appended
- `a+` access for reading and writing, content will be appended

- Read a single row with maximum amount of bytes from the file: `$row = fgets($file ,1000);`
- Write a single row to the file: `fputs($file, "Hello World!");`
- Close the file: `fclose($file);`

If you want to read the whole content of a single file, there are two possibilities:

- Read content with iteration

```
<?php
$file = fopen("test.txt","r");
while (!feof($file)) {
    echo fgets($file, 1000);
    echo "<br />";
}
fclose($file);
?>
```

- Read content with a single statement

```
<?php
echo file_get_contents("test.txt");
?>
```

In the following, we present some examples. Have a look for further information according to file functions in PHP on [www.php.net/manual/en/](http://www.php.net/manual/en/).

```
<?php
$datei = fopen("info.txt","r"); /* open the file in read-only mode and get the file handler */
echo fread($datei,1000); /* read the file with length of 1000 bytes for the file handler */
fclose ( $datei ); /* close file handler */
/* error message will be exchanged by your own message */
$datei = fopen("info.txt","r") or die ("Not able to open info.txt!");
echo fread($datei ,1000);
fclose($datei);
$datei = fopen("info.txt ","r");
echo fgets($datei ,1000); /* get a line from the file handler with length of 1000 bytes */
fclose($datei);
$datei = fopen("info.txt","r");
$zeile = true; /* read line for line of the file */
while ($zeile) {
    $zeile = fgets($datei,100);
    echo "<b> $zeile <b><br>";
}
fclose($datei);
$zeilen = 0;
if ($datei = fopen("daten.txt","r")) {
    while (! feof($datei)) {
        if (fgets($datei ,1048576)) {
            $zeilen ++;
        }
    }
}
echo $zeilen;
fclose($datei);
$datei = fopen("daten.txt","w");
echo fwrite($datei, "Hello World!" ,100);
fclose($datei);
$datei = fopen("daten.txt","r+");
fseek($datei,50,SEEK_CUR); /* set the handler to a particular position in the file */
echo ftell($datei) . "<br >"; /* get the position of the handler in the file */
fseek($datei,15,SEEK_CUR );
echo ftell($datei) . "<br >";
fclose ($datei);
?>
```

### 2.7.4 Uploading files

If you want to upload files with PHP from the user's client to the Web server, you first of all need an HTML formular that provides some particular fields. The hidden field "size" defines the maximum size of the uploaded file. The attribute `enctype` defines the kind of file transfer to the Web server. With action the PHP-script itself will be transferred through the variable `$_SERVER['PHP_SELF']` via POST to the Web Browser.

In the following, we present the PHP-script for uploading a file from the client to the server:

```
<?php /* execute PHP-script if the submit-button has already been clicked */
if (isset($_POST['submit'])) {
    $uploadfolder = 'daten'; /* define the upload folder */
    $filename = $_FILES['myfile']['name'];
    echo ('PHP-Script: <b> ' . $_SERVER['PHP_SELF'] . '</b><br/><br />');
    echo ('Upload-Folder: <em> ' . $uploadfolder . '</em><br/>');
    echo ('Filename: ' . $filename . '<br /><br />');
    echo ('$_FILES: <br />');
    foreach ( $_FILES as $key => $wert) {
        foreach ( $_FILES[$key] as $inx => $wert) {
            echo ('[' . $inx . ']: ' . $wert . '<br />');
        }
    }
    echo ('<br />');
    /* show server dates */
    echo ('$_SERVER: <br />');
    foreach ( $_SERVER as $key => $value) {
        echo ('Server: ' . $key . ' ' . $value . '<br />');
    }
    echo ('$_POST: <br />');
    foreach ( $_POST as $key => $value) {
        echo ('[' . $key . ']: ' . $value . '<br />');
    }
    echo ('<br/>');
    // error messages
    $errMsg="";
    if (empty($filename))
        $errMsg = 'Filename is empty!<br />';
    if ($_FILES['myfile']['size'] > $_POST['size'])
        $errMsg .= 'Size of file too big! <br/>';
    if (file_exists($uploadfolder . '/' . $filename))
        $errMsg .= 'File already exists.<br />';
    if (!preg_match('^[a-zA-Z0-9._-]*$', $filename))
        $errMsg .= 'There are special characters in the filename!<br />';
    // upload?
    if (strlen($errMsg)==0) {
        if (move_uploaded_file($_FILES['myfile']['tmp_name'], "$uploadfolder/$filename")) {
            echo ($uploadfolder . '/' . $filename . ' upload' . '<br />');
            // show content of target folder
        }
        else {
            echo ('Error: ' . $_FILES['myfile']['error']);
            echo ($errMsg);
        }
    }
    else {
        echo ('Error: ' . htmlspecialchars($filename) . '<br />');
        echo ($errMsg);
        // no file
    }
}
else {
    ?>
    <form enctype="multipart/form-data" action="<?php echo $_SERVER['PHP_SELF'];>" method = "POST">
    <input name="myfile" type="file" />
    <input name="size" type="hidden" value="1048576" />
    <input name="submit" type="submit" value="upload the file" />
    </form>
    <?php
}
?>
```



**Exercise.** Get used to work with files and folders.

1. Implement a PHP script that shows a simple visitor counter for the Web site. Each time a visitor attends the page, the counter will be incremented by one. Additionally, you should log the client's IP address and the time of access (address and time).
2. Implement a PHP script that allows to read CSV-files and to present the content in formatted HTML. (Hint: CSV stands for Comma Separated Values or also for Character Separated Values because dates are separated according to a particular character.)
  - (a) Create a table of addresses in Excel and save the file with the extension .csv. The table should have following columns: surname, firstname, streetname, streetnumber, postal code and town. Find some entries for the addresses.
  - (b) Open the csv-file in notepad++ and you will see that each line in the table corresponds to a single line in the csv-file. You can read each line in the file with the help of arrays in PHP. (Hint: `fgetcsv`)
  - (c) The script shall implement following behaviour:
    - i. Open the file with reading access.
    - ii. Read the content of the file via using an array.
    - iii. Present the content of the file in HTML by using a `<table>`.
    - iv. Close the file correctly.
3. Implement a web site that allows users to upload a file from the client to the server. No matter, which file the user selects on the client, the name of the uploaded file on the web server has to be as follows: `firstname_surname.xxx`. All uploaded files have to be stored in one common folder on the web server.

## 2.8 Task “Casting”

FM4 plans to promote young music talents more intensively in future. It is projected to play their music more often on the radio. Potential candidates will be searched online. Subsequently, FM4 assigns you to design and develop an application form for the candidates.

**Requirements** The application form has to implement following requirements:

The candidate has to be able

- to enter first name, surname, date of birth, postal address and e-mail address. All fields are mandatory.
- to define different music instruments, which he/she can play. There are following options: piano, drums, guitar, bass, transverse flute, trumpet, saxophone and other. The candidate will be able to enter “beginner” or “advanced” for each music instrument. The selection of music instruments is mandatory (at least one music instrument has to be selected).
- to upload a photograph of himself/herself. This is mandatory.
- to select foreign languages, which he/she speaks. There are following options: Englisch, French, Italian, Spanish, Czech and Russian. Multiple choice has to be possible. The selection of languages is optional.
- to give an explanation why he/she will attend the casting.
- to have the possibility to subscribe to the casting-newsletter.
- to send his/her data to FM4.
- to redefine or reenter data in case of “errors”.

**Your Job** Design and develop the requested formular for FM4 in HTML with consideration of CSS and PHP (maybe JavaScript)!

For layout use an extern, unique CSS. Consider that all formular fields have to be marked in terms of colour. Use tables to layout text and formular fields properly.

Keep in mind your target group! How shall the application form look like if you want to convince, for instance, yourself to attend the casting?

## 2.9 Task "Codebreaker" with PHP

**Exercise.** In this exercise the game Codebreaker shall be implemented. In this game the user should guess a four-digit code. The code contains the letters A, B, C, D, E, F, G whereby each letter occurs only once. After each guess the user gets a hint about his proposed code (combination of letters).

- A red point indicates that the user's proposed letter is part of the code and situated at the right position.
- A black point indicates that the user's proposed letter is part of the code but not situated at the right position.
- A white point indicates that the user's proposed letter is not part of the code.

Example:

The example shows three screenshots of the Codebreaker game interface:

- Screenshot 1:** Title "Codebreaker (New Game)". Text: "Versuchen Sie den Code zu erraten. Sie haben noch 10 Versuche." Below are four empty input fields and a "check" button.
- Screenshot 2:** Title "Codebreaker (New Game)". Text: "Versuchen Sie den Code zu erraten. Sie haben noch 7 Versuche." Below is a table showing three guesses and their hints, followed by four empty input fields and a "check" button.
 

Guess	A	B	C	D	Hint
1					● ○ ● ●
2	A	E	C	D	● ○ ● ●
3	E	B	C	D	● ● ● ●
- Screenshot 3:** Title "Codebreaker (New Game)". Text: "Versuchen Sie den Code zu erraten. Sie haben noch 4 Versuche." Below is a table showing six guesses and their hints, followed by the text "Sie haben gewonnen."
 

Guess	A	B	C	D	Hint
1					● ○ ● ●
2	A	E	C	D	● ○ ● ●
3	E	B	C	D	● ● ● ●
4	E	B	D	C	● ● ● ●
5	D	B	E	C	● ● ● ●
6	C	B	D	E	● ● ● ●

You should fulfill following requirements:

- Work with sessions.
- At the beginning of the game the code will be built up randomly with respect to the letters A, B, C, D, E, F, G.
- The user has 10 attempts to guess the code.
- The user's input is handled with the help of text fields.
- After each guess the user will get a hint about his proposed code. The hints are visualised by different colours (see above).
- The different attempts are shown and the corresponding hints are visualised (see above).
- The user has the possibility to start a new game at any time.
- Take care of commenting your code accordingly!!!

## 2.10 Object-Oriented Programming with PHP

The concept of object-oriented programming had its roots in SIMULA 67, a general purpose programming language developed at the Norwegian Computing Center. In 1980s, however, Smalltalk was developed and some consider Smalltalk to be the base model for a purely object-oriented programming language.

**What is an object?** “An object is an instance of a class.” An object is often created by instantiating a class with the **new** keyword and a constructor invocation.

The following enumeration summarizes the properties of objects:

- ... are structures with *state and behavior*.
- ... *cooperate* to perform complex tasks.
- ... can communicate with each other by means of *messages*.
- ... have precise *interfaces* specifying which messages they accept.
- ... have *hidden state*.

**What is a class?** “A class is a collection of data fields that hold values and methods that operate on those values. A class defines a new reference type.”

A class definition consists of a signature and a body. The class signature defines the name of the class. The body of the class is a set of members, such as fields and methods, and may include constructors, initializers, and nested types. Members can be static or nonstatic. A static member belongs to the class itself, whereby a nonstatic member is associated with the instance of a class (an object).

Consider the following class named **Point**, providing two attributes **x**, **y** and three methods **draw()**, **move()**, and **remove()**:

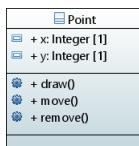


Figure 2.1: Class definition in UML notation

The signature of a class may declare that the class **extends** another class (see Inheritance). The extended class is known as superclass and the extension as the subclass. A subclass inherits the members of its superclass and may declare new members or override inherited methods with new implementations (see Polymorphism). The members of a class may have access modifiers **public**, **protected** or **private**. These modifiers specify their visibility and accessibility to clients and subclasses.

**What are the principles of the object-oriented approach?** The principles of the object-oriented approach are the following:

- **Abstraction.** “An **abstraction** is a view or representation of an entity that includes only the most significant attributes. In a general sense, abstraction allows one to collect instances of entities into groups in which their common attributes need not be considered.” In the world of programming languages, abstraction reduces the complexity of programming; its purpose is to simplify the programming process.
- **Encapsulation and Information Hiding.** **Encapsulation** means that “different components of a software system should not reveal (dt. zeigen, enthüllen) the internal details of their respective implementations.” An advantage of encapsulation is that one programmer can implement the details of a component without explaining these internals to other programmers. In contrast to **information hiding**, which is the process of hiding implementation details that are likely to change. So, information hiding is concerned with how an item is hidden. Abstraction is only concerned about which item should be hidden.

- Inheritance. “In object-oriented programming, the mechanism for a modular and hierarchical organization is a technique known as **inheritance**.”
- Polymorphism. According to the Greek words “poly” (many) and “morphos” (form), the term **polymorphism** “is the ability of an entity (e.g. variable, class, method, object, code, parameter, ...) to take on different meanings in different contexts.” The entity that takes on different meanings is known as a polymorphic entity. “In the context of object-oriented design, it refers to the ability of a reference variable to take different forms.”

### PHP enables - as object-oriented language - programming classes and instantiating objects.

You can declare classes as follows:

```
class ClassName {  
    var $attribut _a1;  
    var $attribut _x;  
    function __construct() {}  
    function fun_a() {}  
    function fun_x() {}  
}
```

You extend a base class with sub classes as follows.

```
class BaseClass {  
    function __construct() {  
        echo ("In BaseClass constructor.<br/>");  
    }  
}  
class SubClass extends BaseClass {  
    function __construct() {  
        parent::__construct();  
        echo ("In SubClass constructor.<br/>");  
    }  
}
```

You get access to a class by creating a new object:

```
$object = new ClassName;  
$object -> fun_x();
```

Let us start with a simple example: we write a class with the name *Circle*. Classnames start with an uppercase letter. First of all, we formulate the constructor with the keyword `__construct`. Then, all others methods follow. The sourcecode is within the class declaration.

```

class Circle {
    var $rad;
    var $con;
    var $sur;
    var $name;
    const PI = 3.14159;
    function __construct ($r, $n) {
        $this->rad = $r;
        $this->name = $n;
        $this->con = $this->contour();
        $this->sur = $this->surface();
    }
    function contour () {
        return (2 * $this->rad * Circle::PI);
    }
    function surface () {
        return (($this->rad*$this->rad) * Circle::PI);
    }
    function display () {
        echo ('Name = ' . $this->name . '<br/>');
        echo ('Radius = ' . $this->rad . '<br/>');
        echo ('Contour = ' . $this->con . '<br/>');
        echo ('Surface = ' . $this->sur . '<br/>');
    }
}

```

Now, we are going to create objects or instantiating the class *Circle*.

```

$small = new Circle(1.0, "Small circle");
$big = new Circle(130, "Big circle");
$big->display();
$small->display();

```

You can get access to the constant value of the class as well.

```

echo ('Constant PI: ' . Circle::PI . '<br/>');

```

You can modify the access to attributes, methods and functions of the class as follows:

<i>keyword</i>	<i>description</i>
public	Variables are visible and changeable outside the class.
private	Variables are protected from external access.
protected	Variables can only be used in the class or in subclasses.
final	The keyword protects from being overridden from methods or classes.
static	The keywords allows to access variables without having an instance of the class.
const	The keyword is similar to static.

**Exercise.** Get used to work with classes and objects.

1. Create a PHP program with a class (Student) with the following properties: student\_id, student\_name, student\_address, student\_state, student\_zip, and student\_age. The program includes get and set methods for each property. Validate the proper type and size of data passed into each property. The program includes the ability for each property to use the constructor to set values. Create an instance of the class passing properties through the constructor. Change two of the properties using set methods. Display the properties using get methods.
2. Implement a base class called *Geometry* and corresponding sub classes as presented in the following table.

square (Quadrat)	rectangle (Rechteck)	triangle (Dreieck)	circle (Kreis)	annulus (Kreisring)
$A = l^2$	$A = l \cdot b$	$A = \frac{l \cdot b}{2}$	$A = \frac{\pi \cdot d^2}{4}$	$A = \frac{\pi}{4}(D^2 - d^2)$
$e = \sqrt{2} \cdot l$ (Diagonale)	$e = \sqrt{l^2 + b^2}$	$u = \text{sum of side lengths}$	$A = \pi \cdot r^2$	$d_m = \frac{D+d}{2}$
$u = 4 \cdot l$	$u = 2 \cdot (l + b)$		$u = \pi \cdot d = \pi \cdot 2 \cdot r$	$b = \frac{D-d}{2}$ (Breite bzw. Höhe)

Following features have to be implemented:

- (a) Calculate surface, contour, diameter, diagonal as requested for the particular geometric form.
- (b) Display the geometric form with all available - calculated - attributes and a representative picture.
- (c) Test your classes by instantiating different objects. Present them on an HTML web site.

### 3 Task "Ars Electronica Center"

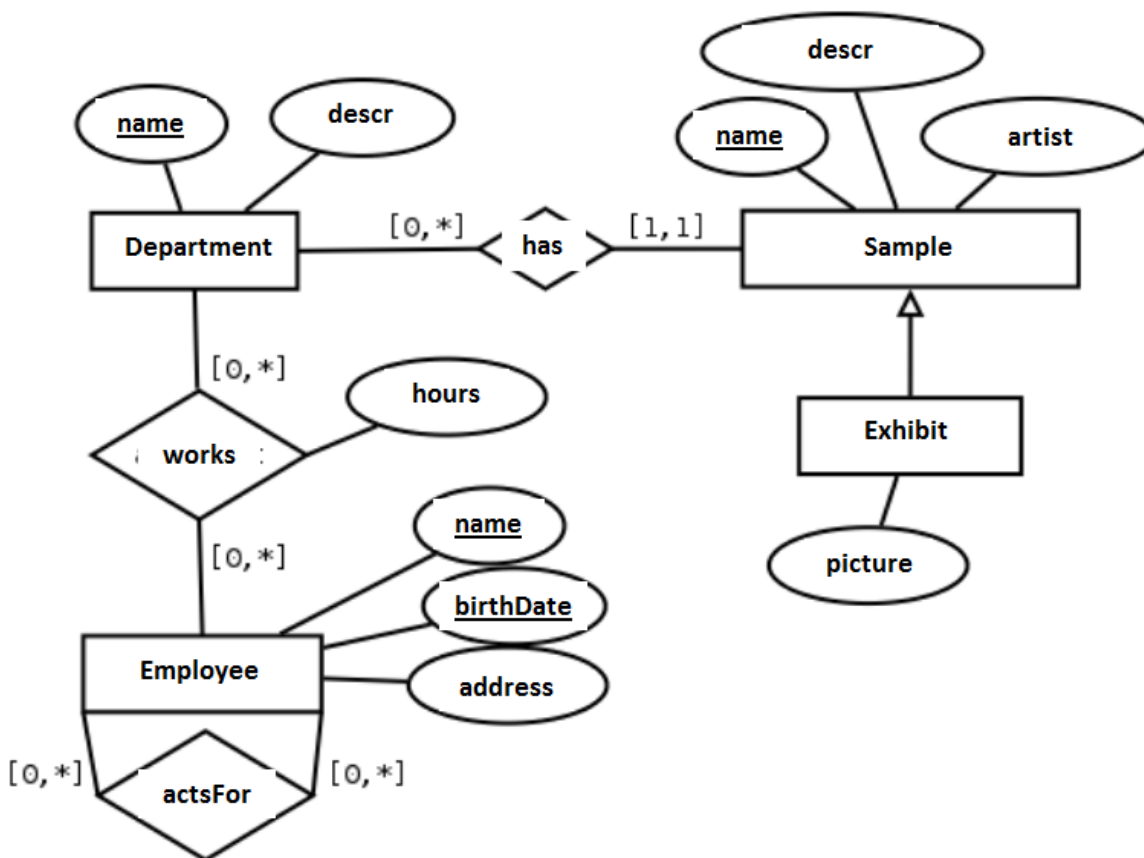
#### Learning objectives

- Repetition of Entity-Relationship Diagrams.
- Repetition of object-oriented programming techniques.
- Implementation of relationships in classes.
- Preparation for database integration in PHP.

---

#### Exercise. Relationships in Classes.

In this exercise you are requested to implement the following ER diagramm by using object-oriented programming techniques.



As real-world example you can use information available about the ARS ELECTRONICA CENTER Linz  
<http://www.aec.at>(<http://www.aec.at/center/de/ausstellungen/>).



## 4 MySQL with XAMPP

### 4.1 Installation

#### 4.1.1 Website

<http://www.xampp.org>

#### 4.1.2 XAMPP Distribution

e.g. XAMPP for Windows, Download XAMPP USB Lite (7zip), Method B: "Installation" without the Installer

- Extract it to C:\ and rename it to e.g. xampp\_s
- Start the installation with C:\xampp\_s\setup\_xampp.bat via shell
- Start Xampp with the statement C:\xampp\_s\xampp\_start.exe
- Start MySQL via second shell with the statement C:\xampp\_s\mysql\bin\mysql.exe
- Stop Xampp via statement C:\xampp\_s\xampp\_stop.exe

### 4.2 First Statements

#### Show current network

```
shell> netstat - nao (used ports of mysql: 80, 443, 3306 - mysql is running)
```

#### Show mysql-statements

```
shell> mysql --help
```

#### Show username

```
shell> mysql -u myusername
```

#### Show password

```
shell> mysql -p mypwd
```

#### Show port

```
shell> mysql -P myPort
```

#### Show serverhost

```
shell> mysql -h myHost
```

#### Connect to the Server

```
shell> mysql -u root -p -h 127.0.0.1 -P 3306 (without password!)
```

### Close the connection to the Server

```
mysql> quit
```

## 5 PHP and MySQL

Combining dynamic web scripting such as PHP and database systems such as MySQL enables implementing powerful Web applications. When implementing dynamic web sites using databases in background, following actions have to be performed:

1. Connect to the database and choose the database.
2. Execute an SQL statement.
3. Display the results in HTML or process data in any other ways.
4. Free the resultset.
5. Close the opened database connection.

The PHP statements with MySQL PDOs are as follows (is shown by using a simple example):

```
try {
    /* build up database connection */
    $dsn = 'mysql:dbname=mini_school;host=127.0.0.1';
    $user = 'root';
    $password = '';
    $dbh = new PDO($dsn, $user, $password);

    /* student (firstname varchar(32), lastname varchar(32)) */
    echo "<h3>Simple query by using the statement query </h3>";
    $sql = "SELECT * FROM student";
    $users = $dbh->query($sql);
    foreach ($users as $row) {
        print $row["firstname"] . "-" . $row["lastname"] . "<br />";
    }

    /* $users->execute();
    foreach ($users as $row) {
        print $row["firstname"] . "-" . $row["lastname"] . "<br />";
    }
    */

    echo "<h3>Prepared Statement with unnamed parameters </h3>";
    $statement = $dbh->prepare("SELECT * FROM
        student WHERE firstname = ? AND lastname = ?");
    $statement->execute(array('Max', 'Mustermann'));
    while($row = $statement->fetch()) {
        echo $row['firstname'] . " " . $row['lastname'] . "<br />";
    }

    echo "<h3>Prepared Statement with named parameters </h3>";
    $statement = $dbh->prepare("SELECT * FROM student
        WHERE firstname = :vorname AND lastname = :nachname");
    $statement->execute(array(':vorname' => 'Max', ':nachname' => 'Mustermann'));
    while($row = $statement->fetch()) {
        echo $row['firstname'] . " " . $row['lastname'] . "<br />";
    }
}
```

```
echo "<h3>Amount of rows in the result set</h3>";
$statement = $dbh->prepare("SELECT * FROM student
    WHERE firstname = ?"); $statement->execute(array('Max'));
$anzahl_user = $statement->rowCount();
echo "We found $anzahl_user student(s).";

echo "<h3>Insert new tuple</h3>";
$statement = $dbh->prepare("INSERT INTO student (firstname, lastname)
    VALUES (?, ?)"); $statement->execute(array('Simone', 'Neuper'));
/* in case of AUTO_INCREMENT id */
$neue_id = $dbh->lastInsertId();
echo "Neuer Nutzer mit id $neue_id angelegt";
} catch (PDOException $e) {
    echo 'Connection failed: ' . $e->getMessage();
}
```