college CDI

# Server side technologies (TCS)

Lesson plan

# Class 01

## PHP language

- Client side
- Static vs dynamic documents
- How HTML documents are generated by PHP
- PHP module needs to be installed and running on server
- to work offline = virtual PHP server needed :
  EasyPHP (Windows), wampserver (Windows), mamp (Apple), BigApache, etc.

## PHP syntax

- **<?php** some PHP coding **?>**
- **<?php**
  **print (**"Hello World"**);** *// print can take only one parameter*
  **?>**
- **<?php**
  **echo** "Hello World !"**;** *// each statement ends with a semicolon*
  **?>**
- **<?php**
  **echo** "Hello"**,**" World!"**;** *// echo can take multiple parameters*
  **?>**

## Special characters

| **Code** | **Result** |
|---|---|
| \n | New line |
| \r | Return |
| \t | Tab |
| \\ | Backslash |
| \$ | $ |
| \" | Double quotes |

-

## Comments

-
  ```
  <?php
  // This is a single line comment

  /* This is
  a multiple lines
  comment */
  ?>
  ```

# Variables

- Letters and numbers + symbols (e.g.: underscore)
- Never starts with a number
- Never includes spaces
- Case sensitive

## Declaring variables

-
  ```
  <?php
  $myVariable = "Hello world!";    // string
  $x = 5;                          // integer
  $y = 7.5;                        // float
  ?>
  ```

## Output data to screen

-
  ```
  <?php
  $name = "John";
  echo "Hello ";          // outputs the word « Hello »
  echo $name;             // outputs the value of the variable (John)
  echo " !";              // outputs the character « ! »
  ?>
  ```

  **OR**

  ```
  <?php
  $name = "John";
  print "<p>Hello ";      // both commands can output HTML tags
  var_dump($name);        // string(4) "John"
  print " !</p>";
  ?>
  ```

# Concatenation

- **Concatenating variables**
  ```
  <?php
  $a = "Hello ";
  $b = "world!";
  echo $a . $b;
  ?>
  ```

- **Concatenating variables and strings**
  ```
  <?php
  $firstName = "John";
  $lastName = "Smith";
  echo "Hello " . $firstName . " " . $lastName. " !";
  ?>
  ```

- **Adding text to a string using « .= »**
  ```
  <?php
  $a = "John";
  $a .= " Smith";
  echo $a;
  ?>
  ```

- Using single and/or double quotes

## Useful string functions

| | |
|---|---|
| **strtolower :** | Converts string characters to lower case. |
| **strtoupper :** | Converts string characters to upper case. |
| | *echo strtoupper( "hello world!");* |
| **strlen :** | Counts and returns a string's number of characters (including spaces). |
| **str_word_count :** | Counts and returns the number of words of a string. |
| **substr :** | Return part of the string using three parameters : the string to be shortened, the position of the starting point, the number of characters to be returned. |
| | *echo substr($myVariable,5,16);* |
| **str_replace :** | Replaces a specified string values in a given string using three parameters : the text to be replaced, the replacement text and the text that is analyzed. |
| | *echo str_replace ("This", "Here", "This is an example.");* |
| **strpos :** | Locates and return the position of a suite of character(s) within a string. |
| | *echo strpos($myVariable,"ex");* |
| **ucfirst :** | Makes the first character of a string an upper case. |

| | |
|---|---|
| **lcfirst:** | Makes the first character of a string an upper case. |
| **wordwrap:** | Wraps a string to a given number of characters. |

*nl2br();*
*$a = "An example: I wonder how it will look like using word wrap function on this text...";*
*echo wordwrap($a,15,"<br>\n");*

## Other functions

| | |
|---|---|
| **chop()** | Removes whitespace or other characters from the right end of a string. |
| **chr()** | Returns a character from a specified ASCII value. |
| **ltrim()** | Removes whitespace or other characters from the left side of a string. |
| **rtrim()** | Removes whitespace or other characters from the right side of a string. |
| **str_shuffle()** | Randomly shuffles all characters in a string. |
| **strcasecmp()** | Compares two strings (case-insensitive). |
| **strcmp()** | Compares two strings (case-sensitive). |
| **strrev()** | Reverses a string. |
| **trim()** | Removes whitespace or other characters from both sides of a string. |

## Heredoc syntax

```
<?php
$myVariable = <<<EOD                          // any suite of characters you want
<h1>Using heredoc syntax is pretty cool</h1>
<p>
It layouts things the way you want!
</p>
EOD;                                          // same suite needs to be used both places
?>
```

### Assignment 1 : Your first PHP script

In a HTML document, use PHP to obtain the following:

The user will write a sentence as a value of a predefined variable.

The output should look like this:

**The sentence you entered is :**
*This is my sentence*

**It is composed of :**

- 4 different words
- 19 characters (including spaces)

**Note:** *Your program should make sure the final output is in lower case with the first letter in upper case.*

# Class 02

# Variables (suite)

## Scope of variables

- Global / local variables

```
<?php
$a=1;                  // global (default)
function hello() {
        echo $a;       // local to function (no data to output)
}
?>
```

- Global keyword

```
<?php
$a=1;                  // global (default)
$b=2;                  // global (default)
function plus() {
        global $a,$b;      // retrieves global variables
        global $result;    // declares new global variable
        $result=$a+$b;     // assigns value to new variable
}
plus();                // launches function
echo $result;          // outputs final data to screen : 3
?>
```

- Pre-defined $GLOBALS associative array

```
$a = 1;
$b = 2;
function plus() {
        $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
}
plus();
echo $b;               //b now equals 3
?>
```

## Constants

- Like variables
- Global
- Value never changed by program
- Written in UPPERCASE by convention

```
<?php
$price = 9.99;              // variable
define("TAX", 1.15);        // constant's value representing 15% tax to apply
echo $price * TAX;
?>
```

## Some predefined constants

| | |
|---|---|
| **__FILE__ :** | Returns the name of the file including its full path. |
| **__DIR__ :** | Returns the name of the directory of the file including its path. |
| **__LINE__ :** | Returns the line number in the source file. |
| | *echo "<b>Full path :</b> " . __FILE__ . "<br />";* |

## Booleans

- true (1) / false (0)

- **Boolean is *false* when the value is :**

    - 0 (zero) integer
    - 0.0 (zero) float
    - Empty or zero string (e.g. : " " or "0")
    - Empty array
    - Empty object
    - NULL special constant

    - Bolean is *true* in all other circumstances

# Numbers

## Integers

```
<?php
$a = 1234;          // decimal
$a = -123;          // a negative
$a = 0123;          // octal (equivalent to 83 decimal)
$a = 0x1A;          // hexadecimal (equivalent to 26 decimal)
$a = 0b11111111;    // binary (equivalent to 255 decimal)
?>
```

## Operators

```
<?php
echo 10 + 2;
echo 10 - 2;
echo 10 * 2;
echo 10 / 2;
?>
```

## Float

- Numbers with decimals
- Operation on integers generating decimals : result = float

  **Example :**
  ```
  <?php
  var_dump(41/8);
  ?>
  ```

  **Result :**
  ```
  float(5.125)
  ```

## Converting a value into an integer

- ***(int)***, ***(intval)*** or ***(integer)***
  *Converting a float to an integer rounds the value to the lowest integer (rounds down)*

  ```
  <?php
  var_dump(41/8);                  // result : float(5.125)
  var_dump((int) (41/8));          // result : int(5)
  var_dump((integer) (41/8));      // result : int(5)
  var_dump((intval) (41/8));       // result : int(5)
  ?>
  ```

## Numerical strings

- Numbers in string do not always behave as strings but as integers

```
<?php
echo 123;               // result : 123
echo 0123;              // result : 83 (octal numbers start with 0)
echo "0123";            // result : 0123 (string)
echo "0123" + "10";     // result : 133 (123 + 10)
echo "0123" + 10;       // result : 133 (123 + 10)
?>
```

## Comparing values

- PHP have problems comparing a number value with the result from operations = Need to use the ***round()*** function.

```
<?php
$x = 8 - 6.4;                           // equals 1.6
$y = 1.6;
var_dump($x == $y);                     // boolean = false
?>
```

```
<?php
$x = 8 - 6.4;                           // equals 1.6
$y = 1.6;
var_dump(round($x) == round($y));       // boolean = true
?>
```

## Booleans and numbers

- **Provide the same result :**
  if($var==true) echo "ok";
  if($var) echo "ok";

- A negative value (e.g. : -2) is considered true (it isn't 0).

- **Booleans converted into integers are :**
  False = 0
  True = 1

- **Booleans converted into floats are rounded down :**
  1.9 = 1
  0.8= 0

-

## Math functions

| | |
|---|---|
| **ceil()** | Rounds a number up to the nearest integer |
| **exp()** | Calculates the exponent of e |
| **floor()** | Rounds a number down to the nearest integer |
| **lcg_value()** | Returns a pseudo random number in a range between 0 and 1 |
| **is_float()** | Returns true if a value is a float |
| **is_int()** | Returns true if a value is an integer |
| **is_nan()** | Checks whether a value is 'not-a-number' |
| **is_numeric()** | Returns true if a value is numerical |
| **max()** | Returns the highest value in an array, or the highest value of several specified values |
| **min()** | Returns the lowest value in an array, or the lowest value of several specified values |
| **mt_rand()** | Generates a random integer using Mersenne Twister algorithm |
| **number_format()** | Formats a number with groups of thousands |
| **pi()** | Returns the value of PI |
| **pow()** | Returns x raised to the power of y |
| **rand()** | Generates a random integer |
| **round()** | Rounds a floating-point number |
| **sqrt()** | Returns the square root of a number |

## Math constants

| | |
|---|---|
| **M_PI** | Returns Pi |
| **PHP_ROUND_HALF_UP** | Round halves up |
| **PHP_ROUND_HALF_DOWN** | Round halves down |

# Date and time

```
<?php
echo "<b>Today is :</b> " . date("l M d, Y") . "<br>";
echo "<b>Today is :</b> " . date("Y/m/d") . "<br>";
echo "<b>Today is :</b> " . date("Y.m.d") . "<br>";
echo "<b>Today is :</b> " . date("Y-m-d") . "<br>";
echo "<b>Today is :</b> " . date("l");                    // Lower case «L»
?>
```

```
<?php
date_default_timezone_set("America/New_York");
echo "<b>The time is :</b> " . date("H:i:s");
?>
```

**Result :**
**The time is :** 21:34:28

```
<?php
echo "<b>The time is :</b> " . date("h:i:sa");
?>
```

**Result :**
**The time is :** 09:35:33pm

## Assignment 2 : Your first PHP script

In a HTML document, use PHP to obtain the following :

The user will enter different values for the main global variables (table diameter and side a and be of a triangle) and the program will output a message such as the following :

**Date :**
Saturday, April 6th 2019

**Time :**
10:25 am

**Your table details :**

| | |
|---|---|
| Diameter : | *User enters diameter* |
| Circumference : | *Program output* |
| Surface area : | *Program output* |

**The length of your triangle hypotenuse :**

| | |
|---|---|
| Side a : | *User enters length of side a* |
| Side b : | *User enters length of side b* |
| Hypotenuse : | *Program output* |

# Class 03

# Arrays

## Creating an array

- Array = variable containing several values
- Values = indexed with numbers starting with zero
- Many ways to create an array using ***array()***
  OR directly OR automatically assigning indexes to variables

```
<?php
$myInstruments = array("Piano", "Drum", "Guitar");
?>
```

***OR***

```
$myInstruments = array();
$Instruments[0] = "Piano";
$Instruments[1] = "Drum";
$Instruments[2] = "Guitar";
```

***OR***

```
$myInstruments = array();
$Instruments[] = "Piano";
$Instruments[] = "Drum";
$Instruments[] = "Guitar";
```

**Output to screen :**

```
<?php
echo $Instruments[0];                    // result : Piano
?>
```

***OR***

```
<?php
echo "I like hearing a " . $Instruments[1] . "and a " . $Instruments[2]
// result : I like hearing a Drum and a Guitar
?>
```

## Replacing index numbers with strings

```
<?php
$myInstruments = array();
$Instruments[Chopin_favorite] = "Piano";
$Instruments[Boom_Boom] = "Drum";
$Instruments[Hendrix_thing] = "Guitar";

echo $Instruments[Hendrix_thing];          // would show : Guitar
?>
```

## Creating an array using keys

**Syntax :**

```
<?php
$myArray = array("a"=>"Piano", "name"=>"John", 29, "Karen");
?>
```

***Outputs to screen :***

```
echo $myArray["a"];             // would show : Piano
echo $myArray["name"];          // would show : John
echo $myArray[0];               // would show : 29
echo $myArray[1];               // would show : Karen
```

Index are attributed to «unkeyed» values the regular way :

```
<?php
$myArray = array("a"=>"Piano", 29, "name"=>"John", "Karen");
                               [0]                 [1]
?>
```

# Conditions

## Control operators

| | |
|---|---|
| **==** | Equal (type and value) |
| **!=** | Different of |
| **<** | Lower than |
| **>** | Higher than |
| **<=** | Lower or equal to |
| **>=** | Higer or equal to |
| **and** OR **&&** | AND / AND (type and value) |
| **or** OR **ll** | OR / OR (type and value) |

## If, else, elseif

```
<?php
$myNumber = 11;

if ($myNumber >= 0 && $myNumber < 10) {           // if the value of the variable is between 0 and 9 :
        echo $myNumber . " is between 0 and 9";
}

elseif ($myNumber >= 10 && $myNumber < 20) {      // if not, if the value of the variable between 10 and 19 :
        echo $myNumber . " is between 10 and 19";
}

else {                                            // if none of the two preceding conditions apply :
        echo $myNumber . " is higher than 19";
}
?>
```

**Explanation**

- FIRST : $myNumber is tested (with the condition between parenthesis)
  Equal or higher than zero AND lower than 10
  TRUE : echo « 11 is between 0 and 9 »

- NOW 2 possibilities :
  ***else*** = directly echo an appropriate message OR second condition to test if the first one fails (***elseif***)
  $myNumber tested : equals or is higher than 10 OR lower than 20
  TRUE : echo « 11 is between 10 and 19 »

- FINALLY : if both conditions false
  **else** is executed : echo « 11 is higher than 19 ».

## Switch

Equivalent of ***if*** followed by several ***elseif***

```
<?php
$name = "Morpheus";

switch ($name) {
        case "John" :                    // if John
        echo "Your name is John.";       // output John
        break;

        case "Karen" :
        echo "Your name is Karen.";
        break;

        case "Morpheus" :
        echo "Your name is Morpheus.";
        break;

        default :                        // if none of the possibilities
        echo "I don't know who you are.";  // output default message
}
?>
```

**Comparison with *if* structure:**

```
<?php
$name = "Morpheus";

if ($name == "John") {                   // if John
        echo "Your name is John.";       // output John
}

elseif ($name == "Karen") {              // not John but Karen
        echo "Your name is Karen.";      // output Karen
}

elseif ($name == "Morpheus") {           // not Karen but Morpheus
        echo "Your name is Morpheus.";
}

else {                                   // otherwise
        echo "I don't know who you are.";  // output this message
}
?>
```

## While

The while condition structure allows us to create a loop repeating itself as long as a condition is true.

**Syntax :**

```
<?php
$number = 5;
$i = 0;
while ($i < $number) {
        echo "Our number isn't " . $i. "<br />";
        $i = $i + 1;                              // OR $i++;
}
echo "Our number is ". $i;
?>
```

**Result :**

```
Our number isn't 0
Our number isn't 1
Our number isn't 2
Our number isn't 3
Our number isn't 4
Our number is 5
```

**Explanation :**

- Value of ***$number*** = 5
  iteration variable (***$i***) = 0
- *While* creates a loop (it repeats itself) as long as the condition is ***true***
  (***as long as $i is lower than 5***)
- When the condition becomes ***false*** (***when $i is equal or higher than 5***)
  the loop is stopped and the program continues being executed (next commands)

## for

- Allows to create loops repeating itself as long as a condition is true
- Useful to count or to limit number of tries, for instance
- Important structure which can replace ***if*** in many cases.

**Syntax :**

```
for ($i=0; $i < $number; $i++) {
        Some commands...
}
```

**Explanation :**

- FIRST : iteration variable (***$i***) = 0

- SECOND :  condition tests if ***$i*** is lower than ***$number***
  As long as the condition is ***true***, $i is increased by 1 AND the command lines are executed

- FINALLY : When the condition is ***false***, the loop stops
  AND the rest of the program is then executed

**Example :**

```
<?php
$number = 5;
for ($i=1; $i < $number; $i++) {
        echo "The number is " . $number . "<br />";
}
echo "And it has been output " . $i . " times";
?>
```

**Result :**

```
The number is 10
The number is 10
The number is 10
The number is 10
The number is 10
And it has been output 5 times
```

**Explanation :**

- ***$number*** = *5* AND **$i** = *0*

- Condition : *as long as **$i is lower than 10***) commands are executed

- When the condition becomes *false* (when ***$i is equal or higher than 10***), loop stops and program continues

## Assignment 3 : Guessing the secret number

In a HTML document, use PHP to obtain the following :

The first variable ($number) to be declared is the one representing the number chosen by the user.

The second variable declared ($secret) is the secret number. You need for the program to generate a random number between 0 and 10.

You need messages saying whether the secret number is higher, lower or equal to the chosen number.

# Class 04

# Retrieving contents from external files

## file_get_contents()

- Most effective way to import content to a PHP document as a string
  = bring the content of an external file into a web document.
  Used for recurrent information : welcome word / main navigation

```
<h1>Welcome</h1>
<p>
<?php echo file_get_contents("welcome_word.txt"); ?>
</p>

OR

<header>
<img src="my-logo.svg" />
<nav>
<?php echo file_get_contents("main-navigation.txt"); ?>
</nav>
</header>
```

## include() & require()

- Can be used just like file_get_contents()
  BUT = Importing any code needing to be executed such as variables or plugin file
  file_get_contents() = only allows you to import static data such as text, HTML tags and CSS
  Can't be used to import code that needs to be executed.

- Attention, on failure :
  **require()** : produces a fatal error and stops the script.
  **include()** : produces a warning and the script continues.

  **Example :**

  ```
  <?php include "myFile.php"; ?>
  ```
  ***OR***
  ```
  <?php require "myFile.php"; ?>
  ```

  **MyFile.php could include the following code and much more :**

  ```
  <?php echo "<p>This is an example</p>"; ?>
  ```

## Using variables list

- include() and require() = allows to import programing and programing elements :

```
<h1>I want a new car!</h1>
<?php include "myVariables.php";
echo "I want to buy a $color $model.";
?>
```

**MyVariables.php could include the following code :**

```
<?php
$color="black";
$model="Mercedes";
?>
```

# File handling and manipulating

- Handling files = very common tasks with web applications.
- Needed = open and process files
  Using PHP functions allowing you to create, read, upload and edit files
- Careful when handling file = mistakes can be catastrophic

## readfile()

- Reads a file and writes it to the output buffer
- Using *echo* = output the content followed by the number of bytes read
  (line breaks are not taken into accounts)

**Example :**

mydata.txt = containsg a simple short sentence.

```
<?php
echo readfile("mydata.txt");
?>
```

**Result :**

This is the content of mydata.txt output to screen
« The file's content is short. It is displayed on three lines.94 »

# fopen(), fread() and fclose()

```
<?php
$myFile = fopen("mydata.txt", "r") or die("Unable to open file!");      // opens file OR error message
echo fread($myFile,filesize("mydata.txt"));                             // reads content of the file
fclose($myFile);                                                        // closes the file
?>
```

## fopen()

fopen() function is also used to read external files (offers different opening modes) :

| | |
|---|---|
| **r** | (read only mode)<br>File pointer is positioned at the beginning of the file. |
| **w** | (write only mode)<br>Deletes the contents of the file or creates a new file if it doesn't exist.<br>File pointer positioned at the beginning of the file. |
| **a** | (write only mode - append mode)<br>Preserves existing data of the file or creates a new file if the file doesn't exist.<br>File pointer is positioned at the end of the file. |
| **x** | Creates a new file (write only mode)<br>Returns FALSE and an error if the file already exists. |
| **r+** | Opens a file (read and write mode)<br>File pointer is positioned at the beginning of the file. |
| **w+** | Opens a file (read and write mode)<br>Deletes the contents of the file or creates a new file if it doesn't exist.<br>File pointer positioned at the beginning of the file. |
| **a+** | Open a file (read and write mode - append mode)<br>Preserves existing data of the file or creates a new file if the file doesn't exist.<br>File pointer positioned at the end of the file. |
| **x+** | Creates a new file (read and write mode)<br>Returns FALSE and an error if the file already exists. |

## fread()

Reads from an open file.

**fread**($myFile,filesize("mydata.txt"));

First parameter = file to be read (*$myFile*)
The second parameter = maximum number of bytes to read (*filesize* = the entire content of the file)

## fclose()

Closes file. Having several unused files opened would be taking resources from the server.

**fclose**($myFile);

## fgets() and fgetc()

Reads a single line from a file and *fgetc()* reads a single character.

```
<?php
$myfile = fopen("mydata.txt", "r") or die("Unable to open file!");
echo fgets($myfile);                                          // could be fgetc
fclose($myfile);
?>
```

**Result :**

The first line (or character) of the file is read
and the pointer is positioned to the beginning of the following line.

## feof()

Checks if the "end-of-file" has been reached.
Can be used in a loop in order to read all lines of a file :

```
<?php
$myFile = fopen("mydata.txt", "r") or die("Unable to open file!");
while(!feof($myFile)) {
        echo fgets($myFile) . "<br>";
        }
fclose($myFile);
?>
```

# Creating and modifying external files

## Creating a file with fopen()

- Can also be used to create a file (If used to open an inexisting file) (write or append modes must be chosen)

```
<?php
$myFile = fopen("mycontent.txt", "w")
?>
```

## Writing into a file using fwrite()

- First parameter = name of the file to write in
- Second parameter = string to be written

```
<?php
$myFile = fopen("mydata.txt", "w") or die("Unable to open file!");

$txt = "Let's invite John\n";
fwrite($myFile, $txt);

$txt = "But let's not forget to invite Mary\n";
fwrite($myFile, $txt);

fclose($myFile);
?>
```

**Overwriting a file using *fwrite()* :**

Go through the same process of using *fwrite()*
BUT with different strings (existing content will be overwritten)

## fseek()

- Moves the file pointer from its current position to a new one (forward or backward), specified by the number of bytes

```
fseek($myFile, 0);          // brings the pointer to the beginning of the file
```

## file_put_contents()

- Same thing as using fopen(), fwrite() and fclose() (twice slower than the usual structure)

  **Syntax :**
  file_put_contents(*file,data,mode,context*)

  **file :** Name of the file to write in.

  **data :** variable containing the string to write.

  **mode :** How to write the file (default = overwrite)
  FILE_APPEND (to append to existing data

  **context :** For data stream

  **Example :**
  <?php
  $myData = "This is a test! ";
  **file_put_contents**("mycontent.php",$myData,FILE_APPEND);
  ?>

### Assignment 4 : Create a visits counter

In a HTML document, use PHP to show the number of time the page has been visited like in the following example :

« This page has received 6 visits up to now. »

You will need to create a text file which will be used as a counter and you will need to read and write in this text file each time the page is visited.

## Class 05

### Revision

### Workshop

## Class 06

### Mid-term exam

# Class 07

# The superglobals

## $_SERVER

- Array containing different predefined indexes = used to retrieve informations about the server

### Syntax:

$_SERVER["PREDEFINED_INDEX"]

### Indexes (selection):

**PHP_SELF**
Returns the name of the script currently being executed.

**SERVER_NAME**
Returns the name of the server on which the script is being executed.

**SERVER_SOFTWARE**
Returns the server's ID string.

**SERVER_PROTOCOL**
Returns the name and the revision of the information protocol.

**REQUEST_METHOD**
Returns the request method used to access a page *(GET, HEAD, POST, PUT).*

**HTTP_REFERER**
Returns the URL of the page from which the current page was called.

**HTTP_USER_AGENT**
Returns the User_Agent of the current request, if available.

**REMOTE_ADDR**
Returns the user's IP address.

**REMOTE_PORT**
Returns the user's computer port used to make a request to the web server

**SCRIPT_NAME**
Returns the path of the current script.

# Handling forms

## Writing in a file using a form with « method="post" »

- HERE : Form field will be used to write into *greeting.php*
- Superglobal ***$_POST*** uses ***name=***"  " values to transfer contents

**index.php**

```
<DOCTYPE html>
<html lang="fr">
<head>
<title>Writing into a file using forms</title>
</head>
<body>

<form action="greeting.php" method="post">
First name: <input type="text" name="first_name"><br>
Last name: <input type="text" name="last_name"><br>
<button type="submit">Submit</button>
</form>

</body>
</html>
```

**greeting.php**

```
<html>
<body>

Welcome
<?php echo $_POST["first_name"] . " " . $_POST["last_name"]; ?>

// Result : Welcome John Smith

</body>
</html>
```

## Writing in a file using a form with «method="get"»

**index.php**

```
<DOCTYPE html>
<html lang="fr">
<head>
<title>Writing into a file using forms</title>
</head>
<body>

<form action="greeting.php" method="get">
First name: <input type="text" name="first_name"><br>
Last name: <input type="text" name="last_name"><br>
<button type="submit">Submit</button>
</form>

</body>
</html>
```

**greeting.php**

```
<html>
<body>

Welcome
<?php echo $_GET["first_name"] . " " . $_GET["last_name"]; ?>

// Result : Welcome John Smith

</body>
</html>
```

## Difference between «post» and «get»

***$_POST***

- Variables passed to the script using an *http request* (they are invisible = embedded in the HTTP request)
- Well adapted to sensitive data
- Unlimited amount of data to be sent (+ supports many advanced options)

**$_GET**

- Variables passed to the script using the URL (information actually added to the URL = visible to everyone)
- To be used for non-sentitive data (URL can be bookmarked = sometimes be useful)
- Limited to transfers of approximately 2000 characters

**Warning :**

Never use forms without proper validation in order to protect yourself from spammers and hackers.

# Basic form validation

*See course's notes TCS-SERVER_SIDE_TECHNOLOGIES_07-ENG, page 4.*

- **Variables declaration :**
  $name = $email = $gender = $comment = $website = "";

- **Check if form has been submitted :**

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

### Assignment 5 : Using form to update an external file

In a HTML document, use PHP to create a form that will allow users to input their full name, city, and comment.

Upon submit, the data will be added (append) to existing data in another PHP page.

Finally, the external page will show its results underneath the form, most recent first, in the first page, like it would happen in a blog, for instance.

- **Cleaning the strings :**
  - Function test_input()
  - trim()
  - stripslashes()
  - htmlspecialchars()

- **Required fields**

- **Outputting the data :**
  - echo // to itself
  - $_POST[] // to external file

## Programing your own conditions and error messages

```
<?php
$nameError = $emailError = $genderError = $websiteError = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
        if (empty($_POST["name"])) {
```

```
                    $nameError = "Name is required";
          } else {
                    $name = test_input($_POST["name"]);                    // existing command
  }
?>
...
<label>Name:</label>
<input type="text" name="name" required>
<span style="color:red">* <?php echo $nameErr;?></span>
...
```

## Validating name input and programing an error message

```
<?php
$name = test_input($_POST["name"]);
if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
        $nameError = "Please use letters and white spaces only";
}
```

**preg_match()**

- Inspects a string for pattern.
- Returns ***true*** if a given pattern is found and ***false*** if not

## Validating email input and programing an error message

```
<?php
$email = test_input($_POST["email"]);
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        $emailError = "Please use a valid email format";
}
```

**filter_var()**

- This one verifies that the email format is respected
- Many PHP filter functions and constants available
- Selected list to come

## Validating url input and programing an error message

```
<?php
$website = test_input($_POST["website"]);
if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
        $websiteError = "Please provide a valid URL";
}
```

## filter_var() and constants

```
<?php
$string = "<strong>Hello World!</strong>";
$finalString = filter_var($string, FILTER_SANITIZE_STRING);
echo $finalString;
?>
```

## Constants of filter_var()

| | |
|---|---|
| FILTER_VALIDATE_EMAIL | Validates an e-mail address |
| FILTER_VALIDATE_FLOAT | Validates a float |
| FILTER_VALIDATE_INT | Validates an integer |

| | |
|---|---|
| FILTER_VALIDATE_IP | Validates an IP address |
| FILTER_VALIDATE_URL | Validates a URL |
| FILTER_SANITIZE_EMAIL | Removes all illegal characters from an e-mail address |
| FILTER_SANITIZE_ENCODED | Removes/Encodes special characters |
| FILTER_SANITIZE_MAGIC_QUOTES | Apply addslashes() |
| FILTER_SANITIZE_NUMBER_FLOAT | Remove all characters, except digits, +- and optionally .,eE |
| FILTER_SANITIZE_NUMBER_INT | Removes all characters except digits and + - |
| FILTER_SANITIZE_SPECIAL_CHARS | Removes special characters |
| FILTER_SANITIZE_STRING | Removes tags/special characters from a string |
| FILTER_SANITIZE_URL | Removes all illegal character from s URL |

# Class 08

## Uploading files using PHP

```
<form action="#" method="post" enctype="multipart/form-data">
        <h3>Select your image :</h3>
        <input type="file" name="userImage">
        <input type="submit" value="Upload Image" name="submit">
</form>
<?php
$target_dir = "uploads/";                                        // directory on server
$target_file = $target_dir . basename($_FILES["userImage"]["name"]);
$uploadOk = 1;
$imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));

if(isset($_POST["submit"])) {                                    // checks if the file is really an image
                $check = getimagesize($_FILES["userImage"]["tmp_name"]);
                if($check !== false) {
                        echo "This file is an image - " . $check["mime"] . ".";
                        $uploadOk = 1;
                } else {
                        echo "This file is not an image.";
                        $uploadOk = 0;
                }
}
if (file_exists($target_file)) {                                 // Checks if file already exists
        echo "Sorry, file already exists.";
        $uploadOk = 0;
}
if ($_FILES["userImage"]["size"] > 500000) {                     // Checks file size
        echo "Sorry, your file is too large.";
        $uploadOk = 0;
}
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg" ) {        // Allows formats
        echo "Sorry, only JPG, JPEG and PNG are allowed.";
        $uploadOk = 0;
}
if ($uploadOk == 0) {                                            // checks if the upload was OK
        echo "Sorry, your file was not uploaded.";
        // if everything is ok, try to upload file
        } else {
                if (move_uploaded_file($_FILES["userImage"]["tmp_name"], $target_file)) {
                        echo "The file ". basename( $_FILES["userImage"]["name"]) . " has been uploaded.";
                } else {
                echo "Sorry, there was an error uploading your file.";
        }
}
?>
```

# Sending mail with PHP

## mail()

- Sends emails directly from a script
- Needs working email system on server (program defined by the configuration settings of the php.ini file) (usually installed and available on all servers)
- Local : might have to change the configuration (application such as Mamp)

**Syntax:**
mail(*to,subject,message,headers,parameters*);

| | |
|---|---|
| **to** | (Required) Email address of the receiver. |
| **subject** | (Required) Subject line of the message. |
| **message** | (Required) Message to be sent.<br>Line break must be made using **\*n***.<br>Maximum line length of 70 characters. |
| **headers** | (Optional) Additional headers, (e.g.: From, Cc, Bcc).<br>Additional headers must be separated using **\*r*\*n***. |
| **parameters** | (Optional) Additional parameters to the *sendmail* program. |

**Example:**

```
<?php
$msg = "First line of the message\nSecond line of the message";
$msg = wordwrap($msg,70);                          // wordwrap max. 70 characters
mail("someone@somewhere.com","My subject line",$msg);
?>
```

## Extra headers:

```
<?php
$to = "someone@somewhere.com";
$subject = "This is my subject";
$headers = "From: me@server.com" . "\r\n";
$headers .= "CC: someoneelse@server.com";
$msg = "First line of the message\nSecond line of the message.";
$msg = wordwrap($msg,70);

mail($to,$subject,$msg,$headers);
?>
```

## HTML email

```
<?php
$to = "someone@somewhere.com";
$subject = "My HTML email";

$headers = "From: me@server.com" . "\r\n";
$headers .= "CC: someoneelse@server.com" . "\r\n";
$headers .= "MIME-Version: 1.0" . "\r\n";
$headers .= "Content-type:text/html;charset=UTF-8" . "\r\n";

$msg = "
        <html>
        <head>
        <title>My HTML email</title>
        </head>
        <body>
        <h1>Hello world!</h1>
        <p>This is my message</p>
        </body>
        </html>
";

mail($to,$subject,$msg,$headers);
?>
```

### Final project :

### Create a web page that will be updated using a form in an admin control pannel

**index.php**
Is the one-pager web site to update.

This page must contain a header and a footer, as well as a main an aside sections with different contents. The main section will have a title and a welcome text and an aside section showing a picture and a short description of the picture.

Use CSS to make the overall aspect as nice as possible.

**admin.php**
Is the form used to update the index page (admin control panel)

Fields will be used to change the title and the welcome word of the main section as well as the picture and the picture description of the aside section.

# Class 9

Following topic optional.
Teacher may use this class to complete previous topic or for practice purposes

# Functions

- Really helpful = subprograms within the main program performing a single task
  Shortens the code

```
function helloWorld() {                      // Creates the function
        echo "Hello world!";
}
helloWorld();                                // Calls the function + outputs to screen
```

## Function arguments

- Information can be passed from arguments to functions
- Argument acts just like a variable + defined within the function's parenthesis
- Several arguments can be added (separated by a comma)

```
function myFunction($myArgument) {                      // argument between parenthesis
        echo "My argument is: $myArgument<br>";         // instructions
}
myFunction("Hello World");                              // value passed to argument
```

### Example :

```
<?php
function carsList($cars) {
        echo "$cars is a nice car<br>";
}
carsList("BMW");
carsList("Ferrari");
carsList("Mercedes");
carsList("Audi");
?>
```

### Result :

```
BMW is a nice car
Ferrari is a nice car
Mercedes is a nice car
Audi is a nice car
```

### Example with multiple arguments:

```
<?php
function carsList($car, $year) {
        echo "My car is a $year $car<br>";
}
carsList("BMW", "1984");
carsList("Ferrari", "1997");
carsList("Mercedes", "2010");
carsList("Lada", "1980");
?>
```

### Result:

```
My car is a BMW 1984
My car is a Ferrari 1997
My car is a Mercedes 2010
My car is a Lada 1980
```

## Using default arguments in functions

In the example, a default parameter is used ($unknown) to be used if no argument is passed

### Example:

```
<?php
function carsList($unknown="Unspecified car") {
        echo "My car is a: $unknown<br>";
}
carsList("BMW");
carsList("");
carsList("Mercedes");
carsList("Audi");
?>
```

### Result:

```
BMW is a: BMW
Ferrari is a: Unspecified car
Mercedes is a: Mercedes
Audi is a: Audi
```

## Returning values in functions

Just like with JavaScript, the statement ***return*** is used to return values.

**Example :**

```
<?php
function addition($a, $b) {
            $c = $a + $b;
            return $c;
}
echo "1 + 2 = " . addition(1, 2) . "<br>";
echo "2 + 2 = " . addition(2, 2) . "<br>";
echo "3 + 2 = " . addition(3, 2) .  "<br>";
?>
```

**Result :**

```
1 + 2 = 3
2 + 2 = 4
3 + 2 = 5
```

# Classes and objects

- Classes and functions may seem alike BUT many differences
- Class contains both variables (***properties***) and functions (***methods***) = ***object***.
- Class = representation (template) of an object
- Several objects (***instances***) can be made of real data from a class

## Creating a class

```
class Person {                      // Capital used for name
            $age;                   // properties = variables
            $gender;
            $city = "Montreal";     // Declared and assigned (default value)
}
```

## Creating an object

```
$person1 = new Person;              // prefix new + class name
```

## Calling an object's method

- A method (function) needs to have been created
- Call a method in relation to a specific object using **->**

### Example :

```
<?php
class Person {                          // Creates new class
        public $age;                    // Object's properties
        private $_city = "Montreal";
        private $_experience = 0;

        public function speak() {       // Creates function
                echo "I am a person!";
        }
}
$person1 = new Person;                  // Creates new object
$person1->speak();                      // Runs function for the object
```

## Properties and methods visibility :

- Visibility attribute = before properties
- ***private*** = can't be modified/not accessible out of the objects created by the class
- ***public*** = make it possible to modify/to use the property outside of the objects created by the class
- SAME AS global/local scopes of variables
- Visibility is applied the same way to functions

### Note
In respect of *PEAR coding standards*, classes' name starts with a capital letter and private properties with an underscore.
https://pear.php.net/

## Accessing objects' properties

```
class Person {
        private $_age;
        private $_city = "Montreal";
        private $_experience = 0;
}
$person1 = new Person;
$perso1->_experience = $perso1->_experience + 1;          // Fatal error = private property
```

### Pseudo-variable $this

- Used to refer to the current object
- You can invoke a private property usign ***$this*** instead of property name

```
class Person {
        private $_experience = 0;
        public function showExperience() {
                echo $this->_experience;
        }
        public function addExperience() {
                $this->_experience = $this->_experience + 1;
        }
}
$person1 = new Person;
$person1->addExperience();
$person1->showExperience();
}
```

# Class 10

## Revision

## Workshop

# Class 11

## Final exam