CS381
Web Application Development

PHP

These class notes are based on the material from our textbook,
Learning PHP, MySQL & JavaScript, 5th ed., by Robin Nixon

Software

WAMP (Windows, Apache, MySQL, and PHP)

is a fully functioning setup used for developing dynamic internet web pages.

AMPPS (Apache, Mysql, PHP, Perl, Python and Softaculous)

is one of the best free open source options of WAMP. **http://ampps.com**

The document root

is the directory that contains the main web documents for a domain. **C:\Program Files\Ampps\www**

PHP (Hypertext Preprocessors)

is originally derived from *Personal Home Page* Tools.

Variables

Comments /* somthing*/ or // something

Semicolons all php commands end with it.

$ placed before all variables; to make php parser faster, as it instantly knows whenever it comes across a variable.

PHP variables loosely typed; they can contain different types of data at different times.

Variable Naming rules:

- Start with $ then a letter of the alphabet or the _ (underscore) character.
- May contain only the characters a-z, A-Z, 0-9, and _ (underscore).
- Do not contain spaces.
- Are case-sensitive; $High_Score is not the same as $high_score.

Variables & Operators

```
$count    = 17;
$count    = 17.5;                                  var_dump( $count );          => float(17.5)

$team     = array('Bill', 'Mary', 'Mike', 'Chris', 'Anne');
$oxo      = array(    array('x', ' ', 'o'),        array('o', 'o', 'x'),        array('x', 'o', ' ')        )
```

Arithmetic Operators

Operator	Description	Example
+	Addition	$j + 1
-	Subtraction	$j - 6
*	Multiplication	$j * 11
/	Division	$j / 4
%	Modulus (the remainder after a division is performed)	$j % 9
++	Increment	++$j
--	Decrement	--$j
**	Exponentiation (or power)	$j**2

Assignment Operators

Operator	Example	Equivalent to
=	$j = 15	$j = 15
+=	$j += 5	$j = $j + 5
-=	$j -= 3	$j = $j - 3
*=	$j *= 8	$j = $j * 8
/=	$j /= 16	$j = $j / 16
.=	$j .= $k	$j = $j . $k
%=	$j %= 4	$j = $j % 4

Comparison & Logical Operators

Comparison Operators

Operator	Description	Example
==	Is *equal* to	$j == 4
!=	Is *not equal* to	$j != 21
>	Is *greater than*	$j > 3
<	Is *less than*	$j < 100
>=	Is *greater than or equal* to	$j >= 15
<=	Is *less than or equal* to	$j <= 8
<>	Is *not equal to* to	$j <> 23
===	Is *identical to* to	$j === "987"
!==	Is *not identical to* to	$j !== "1.2e3"

Logical Operators

Operator	Description	Example
&&	*And*	$j == 3 && $k == 2
and	Low-precedence *and*	$j == 3 and $k == 2
||	*Or*	$j < 5 || $j > 10
or	Low-precedence *or*	$j < 5 or $j > 10
!	*Not*	! ($j == $k)
xor	*Exclusive or*	$j xor $k

and, xor, or their precedencies are lower than =

!= is equivalent to <>

== true if the operands are equal of the same or different types.

=== true if the operands are equal having the same type.

Constants

Constants define("PI", 3.14); $w = $a * PI;

Predefined Constants:

__LINE__	the current line number.
__FILE__	the full path & filename.
__DIR__	the directory.
__FUNCTION__	the function name.
__CLASS__	the class name.
__METHOD__	the class method name.
__NAMESPACE__	the name of the current namespace.

```
echo "<br>This is line: " . __LINE__ . " of file: " . __FILE__;
```

Variable in Quotation

Single	its value will not be inserted; preserving the exact contents.
Double	its value is inserted into the string.

Scope of Variables

- **Variables**
  - **Global** created outside of all the functions can be accessed only by non-function code.
  - **Local** created within a function's body or parameter.
  - **Static** local variables declared with static keyword; like c. `static $int = 0;`
  - **Superglobal** predefined variables; provided by php environment and are global within the program.

- **Access global variables from a function.** `global $isFound;`
- **Execution Operator** between pair of backticks `` `` `$out = `dir c:`;`

echo Vs print

- Both are not actually real functions (they are language constructs) so parentheses are not required.
- echo is faster.
- print only accepts a single argument and always returns 1.

First.php

```php
<?php       // First.php

            $name = "Fred Smith";

            echo "Hi " . $name;
            echo "<br>Hi   $name";
            echo '<br>Hi   $name';

            echo "<br>This is line: " . __LINE__ . " of file: " . __FILE__;

            echo "  " . longDate(time());

            $out = `dir c:`;
            echo "<pre> $out </pre>";


            function longdate($timestamp)
            {           global $name;

                        echo "<br> Access global variables from a function  [ $name ]  from" . __FUNCTION__;
                        return date("D (d) - M (m) - Y", $timestamp);
            }

// ?>                  // not needed if the file contains only PHP code.
```

Superglobal Variables

Superglobal variables are arrays that contain data as the following:

$GLOBALS	global variables - all variables defined in the global scope of the script.
$_SERVER	server environment variables; created by the web server.
$_GET	variables passed to the script via the GET method.
$_POST	variables passed to the script via the POST method.
$_FILES	variables related to file uploads.
$_COOKIE	cookie variables.
$_SESSION	session variables that are available to the current script.
$_REQUEST	all user input including the contents of $_GET, $_POST, and $_COOKIE (but not including $_FILES).
$_ENV	environment variables under which the PHP parser is running.

- You should always sanitize superglobals and other variables before using them via the PHP **htmlentities** function.
- For example, (< and >) are transformed into the strings < and > so that they are rendered harmless.

Implicit and Explicit Casting

php is a loosely typed language that allows us to declare a variable and its type simply by using it.

Implicit Casting php is automatically converts values from one type to another whenever required.

Explicit Casting

- **(int) (integer)** Cast to an integer by dropping the decimal portion.
- **(bool) (boolean)** Cast to a Boolean.
- **(float) (double) (real)** Cast to a floating-point number; all are equivalents
- **(string)** Cast to a string.
- **(array)** Cast to an array.
- **(object)** Cast to an object

Functions

Defining a Function

```
function name( [parameter [, ...]] )
{
                    // Statements
}
```

Some built-in functions

```
<?php       // StrTest.php
            echo strrev(" dlrow olleH");
            echo str_repeat("Hip ", 2);
            echo strtoupper("hooray!");
            echo ucfirst(  strtolower("this iS MY tiME. iT.") );
            printf("<span style='color:#%X%X%X'>Hello</span>", 255, 50, 50);


?>          // output:  Hello world Hip Hip HOORAY!This is my time. it. Hello
```

Function names are case-insensitive, so all PRINT, Print, and PrInT refer to the print function.

Functions

~~Passing Arguments by Reference~~

not to be used any more

```php
<?php    // FixNames.php
$names = fix_names("WILLIAM", "henry", "gatES");
echo $names[0] . " " . $names[1] . " " . $names[2];

function fix_names($n1, $n2, $n3)
{
        $n1 = ucfirst(strtolower($n1));
        $n2 = ucfirst(strtolower($n2));
        $n3 = ucfirst(strtolower($n3));
        return array($n1, $n2, $n3);
}
?>
```

Include & include_once

include "file.php";

include_once "file.php"; to avoid including the file two time by another file.

require & require_once

require_once "file.php"; same as include_once; but stops executing and causing a fatal error if not found.

ucfirst() converts the first character of a string to uppercase.

lcfirst() converts the first character of a string to lowercase.

ucwords() converts the first character of each word in a string to uppercase.

Fix Names Global.php

```php
<?php   // FixNamesGlobal.php

        $a1 = "WILLIAM";
        $a2 = "henry";
        $a3 = "gatES";

        echo $a1 . " " . $a2 . " " . $a3 . "<br>";

        fix_names();

        echo $a1 . " " . $a2 . " " . $a3;

        function fix_names()
        {
                global $a1, $a2, $a3;

                $a1 = ucfirst(strtolower($a1));
                $a2 = ucfirst(strtolower($a2));
                $a3 = ucfirst(strtolower($a3));
        }

?>
```

WILLIAM henry gatES
William Henry Gates

Class

```php
<?php    // class.php

        $user1 = new User("Ahmed", "123");
        $user1->password = "567";                    // Syntax ( with no $)
        $user1->save();
        print_r($user1);                             // display info about a variable in human-readable form
        print(User::PI);

        class User
        {       public $name, $password;
                const PI = 3.14;

                public function __construct( $name, $password){
                        $this->name =$name;
                        $this->password = $password;
                }

                function save() {
                        echo "Save User code <br>";
                }

        }

?>
```

```
Save User code
User Object ( [name] => Ahmed [password] => 567 ) 3.14
```

public	can be accessed from everywhere. The default
protected	can be accessed within the class and by derived class
private	can only be accessed within the class

```php
private function save() {
        echo "Save User code goes here";
}
```

Arrays

```php
<?php   // Array1.php

    $paper1[] = "Copier";
    $paper1[] = "Inkjet";
    $paper1[] = "Laser";
    $paper1[] = "Photo";
    print_r($paper1);
    print "<br>";


    $paper2[0] = "Copier";    // Adding items to an array using explicit locations
    $paper2[1] = "Inkjet";
    $paper2[2] = "Laser";
    $paper2[3] = "Photo";
    print_r($paper2);
    print "<br>";

    for ($j = 0 ; $j < 4 ; ++$j)
        print "<br>  $j: $paper1[$j]";
?>
```

```
Array ( [0] => Copier [1] => Inkjet [2] => Laser [3] => Photo )
Array ( [0] => Copier [1] => Inkjet [2] => Laser [3] => Photo )

0: Copier
1: Inkjet
2: Laser
3: Photo
```

Associative Arrays

reference an item in an array by a name rather than by a number.

```
<?php   // Array2.php
        $paper['inkjet'] = "Inkjet Printer";
        $paper['laser'] = "Laser Printer";
        echo $paper['laser'];


        // using array function to create an array        with ' or "

        $p2 = array(  "copier" => "Copier & Multipurpose",
                      'inkjet' => "Inkjet Printer",
                      'laser' => "Laser Printer",
                      'photo' => "Photographic Paper");

        echo "p2 element: " . $p2['inkjet'] . "<br>";
?>
```

Laser Printer

p2 element: Inkjet Printer

- The names (copier, inkjet, …) are called **indexes** or **keys**
- The items assigned to them (such as Laser Printer) are called **values**.

foreach

```php
<?php  // Array3.php
    $p5 = array("Copier", "Inkjet", "Laser", "Photo");

    foreach( $p5 as $item )
        echo "$item <br>";



    $p6 = array(    'copier' => "Copier & Multipurpose",
                    'inkjet' => "Inkjet Printer",
                    'laser' => "Laser Printer",
                    'photo' => "Photographic Paper");
    print_r($p6);
    foreach($p6 as $item => $desc)
        echo "<br>$item: $desc";

?>
```

Copier
Inkjet
Laser
Photo

```php
<?php       // using list & each, deprecated in PHP 7.2.

    while (list($item, $description) = each($p6))
        echo "$item: $description<br>";
?>
```

Array ([copier] => Copier & Multipurpose [inkjet] => Inkjet Printer [laser] => Laser Printer [photo] => Photographic Paper)
copier: Copier & Multipurpose
inkjet: Inkjet Printer
laser: Laser Printer
photo: Photographic Paper

Multidimensional Array

```php
<?php // Array4.php

    $chessboard = array(
       array('r', 'n', 'b', 'q', 'k', 'b', 'n', 'r'),
       array('p', 'p', 'p', 'p', 'p', 'p', 'p', 'p'),
       array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
       array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
       array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
       array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
       array('P', 'P', 'P', 'P', 'P', 'P', 'P', 'P'),
       array('R', 'N', 'B', 'Q', 'K', 'B', 'N', 'R')
    );

    echo "<pre>";
    foreach( $chessboard as $row )
    {
        foreach( $row as $piece )
            echo "$piece   ";
        echo "<br>";
    }
    echo "</pre>";
?>
```

```
r  n  b  q  k  b  n  r
p  p  p  p  p  p  p  p




P  P  P  P  P  P  P  P
R  N  B  Q  K  B  N  R
```

Accessing one element echo $chessboard[7][3];

```php
<?php // Array5.php
    $array = array("Copier", "Inkjet",
                   "Laser", "Photo");

    list($a, $b, $c, $d) = $array;
    echo "$a - $b - $c - $d";

 // list:  assign array values to multiple variables.
?>
```

Copier - Inkjet - Laser - Photo

Array Functions

is_array	checks whether a variable is an array	`echo (is_array($fred)) ? "YES" : "NO";`
count	returns the number of elements in an array	`echo count($fred);`
sort	sort an array.	`sort($fred, SORT_NUMERIC);` `sort($fred, SORT_STRING);` `sort($fred);` `rsort($fred, SORT_NUMERIC);` `rsort($fred, SORT_STRING);` *returns T/F*
shuffle	randomly order an array.	`shuffle($cards);`
explode	returns an array of the substrings.	`$temp = explode( ' ' , "This is a sentence");`
extract	returns the key/value into php variables.	`extract($_GET);` `// not safe`

```
Array
(    [0] => This
     [1] => is
     [2] => a
     [3] => sentence
)
```

Note: if q is sent to a php script along with value Hi, a new variable called $q will be created and assigned Hi value.

To overcome the conflict with our own variables use **extra prefix**.

```
extract($_GET, EXTR_PREFIX_ALL, 'fg');
// instead of $q will be  $fg_q
```

Array Functions

compact: to create an array from variables and their values. (inverse of extract)

variable names to be supplied in quotes without $, because compact is looking for a list of variable names, not their values.

```php
<?php  // ArrayFunction1.php

    $fname  = "Doctor";
    $sname  = "Who";
    $planet = "Gallifrey";

    $contact = compact('fname', 'sname', 'planet');
    print_r($contact);

?>
```

```
Array
(    [fname] => Doctor
     [sname] => Who
     [planet] => Gallifrey
)
```

Array Functions

reset moves the internal array pointer to the **first** element and returns its value.

end moves the internal array pointer to the **last** element and returns its value.

current return the value of the element which the internal pointer is pointing to.

```
<?php  // ArrayFunction2.php

        $array = array("Copier", "Inkjet", "Laser", "Photo");
        echo current($array) . "<br>\n";

        next($array); next($array);          echo current($array) . "<br>";

        reset($array);                       echo current($array) . "<br>\n";

        end($array);                         echo current($array) . "<br>";
?>
```

```
Copier
Laser
Copier
Photo
```

echo <<<_END is a **heredoc**; a multiline sequence using the <<< operator.

anyting a way of **specifying a string literal, preserving the line breaks and other whitespace**.

_END; This code tells PHP to output everything between the two _END tags.

File Handling

```php
<?php     // FileWrite.php   fh: file handle
    $fh = fopen("testfile.txt", 'w') or die("Failed to create file");
    $text = <<<_END
    Line 1
    Line 2
    _END;

    fwrite($fh, $text) or die("Could not write to file");
    fclose($fh);
    echo "File 'testfile.txt' written successfully";
?>
```

die() prints a message and exit from the current php script. It is equivalent to **exit()**

```php
<?php     // FileRead.php
    $fh = fopen("testfile.txt", 'r') or die("File does not exist or you lack permission to open it");

    while(!feof($fh))
    {     $line = fgets($fh);
          echo "<br>$line";
    }

    fclose($fh)
?>
```

File Handling

```php
<?php    // FileCopy.php

    copy('testfile.txt', 'testfile2.txt') or die("Could not copy file");

    echo "File copied to 'testfile2.txt'";
?>
```

```php
<?php    // FileMove.php or rename

    if (!rename('testfile2.txt', 'testfile2.new'))      echo "Could not rename file";
    else                                                echo "File renamed to 'testfile2.new'";
?>
```

```php
<?php    // FileDelete.php

    if (!unlink('testfile2.new'))   echo "<br>Could not delete file";
    else                            echo "<br>File 'testfile2.new' deleted";
?>
```

File Handling

```php
<?php // FileUpdate.php

    $fh = fopen("testfile.txt", 'r+') or die("Failed to open file");
    $text = fgets($fh);
    fseek($fh, 0, SEEK_END);          // Set position to EOF plus offset 0     // Set position to the start:  fseek($fh, 0);

    fwrite($fh, $text) or die("Could not write to file");
    fclose($fh);
    echo "File 'testfile.txt' updated";
?>
```

```php
<?php   // FileUpdateWithLocking.php          for Multiple Accesses

    $fh = fopen("testfile.txt", 'r+') or die("Failed to open file");
    $text = fgets($fh);
    if (flock($fh, LOCK_EX)) {                          // sets an exclusive file lock
        fseek($fh, 0, SEEK_END);
        fwrite($fh, $text) or die("Could not write to file");
        fflush($fh);                                    // flush the output to the file before releasing the lock
        flock($fh, LOCK_UN);
    }
    fclose($fh);
    echo "File 'testfile.txt' successfully updated";
?>
```

File Handling

```php
<?php   // FileGetEntire.php
    echo "<pre>";                              // Enables display of line feeds
    echo file_get_contents("testfile.txt");    // Reads entire file into a string
    echo "</pre>";
?>
```

```php
<?php      // FileGetHomePage.php
     echo "<br><br><h1> Grabbing a Page <h1>";
     echo file_get_contents("https://en.wikipedia.org/");        // Get the source of a page
?>
```

The contents of the $_FILES array

Array element	Contents
$_FILES['*file*']['*name*']	The name of the uploaded file (e.g., *smiley.jpg*)
$_FILES['*file*']['*type*']	The content type of the file (e.g., *image/jpeg*)
$_FILES['*file*']['*size*']	The file's size in bytes
$_FILES['*file*']['*tmp_name*']	The name of the temporary file stored on the server
$_FILES['*file*']['*error*']	The error code resulting from the file upload

```
print_r($_FILES);

Array ( [filename]  => Array (
   [name] => Schedule 201.png
   [type] => image/png
   [tmp_name] => C:\Program Files\Ampps\tmp\php7748.tmp
   [error] => 0
   [size] => 38674 )
)
```

Uploading Files

```php
<?php // upload.php    enctype='multipart/form-data  is an encoding type that allows files to be sent through POST.
    echo <<<_END
            <html><head><title>PHP Form Upload</title></head><body>
            <form method='post' action='upload.php' enctype='multipart/form-data'>
            Select File: <input type='file' name='filename' size='10'>
            <input type='submit' value='Upload'>
            </form>
    _END;

    if ($_FILES)
    {
        $name = $_FILES['filename']['name'];                          // get the name of the file
        move_uploaded_file($_FILES['filename']['tmp_name'], $name);   // move file from, to
        echo "Uploaded image '$name'<br><img src='$name'>";
    }
    echo "</body></html>";
?>
```

<form enctype="*value*">	Description	from (https://wwww3schools.com/)
application/x-www-form-urlencoded	Default. All characters are encoded before sent (spaces are converted to "+" symbols, and special characters are converted to ASCII HEX values)	
multipart/form-data	No characters are encoded. This value is required when you are using forms that have a file upload control	
text/plain	Spaces are converted to "+" symbols, without encoding.	

Validate Uploaded Files

```php
<?php // FileUploadValidation.php
  echo <<<_END
    <html><head><title>PHP Form Upload</title></head><body>
    <form method='post' action='FileUploadValidation.php' enctype='multipart/form-data'>
    Select a JPG, GIF, PNG or TIF File:
    <input type='file' name='filename' size='10'>
    <input type='submit' value='Upload'></form>
   _END;

  if ($_FILES)
  {       $name = $_FILES['filename']['name'];
          switch($_FILES['filename']['type'])
          {          case 'image/jpeg': $ext = 'jpg'; break;
                     case 'image/gif':  $ext = 'gif'; break;
                     case 'image/png':  $ext = 'png'; break;
                     case 'image/tiff': $ext = 'tif'; break;
                     default:           $ext = '';    break;
          }
          if ($ext)
          {          $name = preg_replace("/[^A-Za-z0-9.]/", "", $name);     // replace using regular expression
                     move_uploaded_file($_FILES['filename']['tmp_name'], $name);
                     echo "Uploaded image $name <br>";
                     echo "<img src='$name'>";
          }
          else echo "'$name' is not an accepted image file";
  }
  echo "</body></html>";
?>
```

Exception.php

```php
<?php // Exception.php

    fix();

    function fix()
    {
        try
        {  throw new Exception('MY Exception');
        }
        catch(Exception $e)
        {  echo "XXX $e XXX";
        }
        finally
        {  // executed regardless of whether an exception has been thrown.
        }
    }

?>
```