



# WEB APPLICATION ENGINEERING II

Lecture #2

Umar Ibrahim Enesi



## Objectives

- Gain understanding on PHP operating modes
- Learn how to declare and initialize variable
- Understand the various datatypes supported by PHP
- Understand the basic operators supported by PHP



# Introduction to PHP

- PHP is a recursive acronym for **PHP: Hypertext Preprocessor**
- Originally called **Personal Home Page** by its creator Rasmus Lerdorf in 1994
- PHP code is usually processed by a PHP interpreter
- PHP code is stored in a file with .php extension
- PHP is especially suited for web development
- PHP can also be used for:
  - Command Line Scripting
  - Writing desktop applications
- Versions of PHP: 1.0, 2.0, 3.0, 4.0 – 4.4, 5.0-**5.6**, 6.x, 7.0 and 7.1



# Advantages of PHP

- Its Free
- Open Source
- Cross Platform (Linux, Windows, Mac, ...)
- Easy to learn
- Its Efficient
- Supports a wide range of database (Mysql, Oracle DB, ...)
- Compatible with almost all web servers (Apache, IIS, ...)
- Support and Documentation



## Basic Syntax: PHP Tags

- PHP interpreter recognizes code placed between PHP opening and closing tags
  - `<?php ... ?>` (*recommended*)
  - `<? ... ?>` short tag (*not encouraged*)
  - `<% ... %>` ASP tag
  - `<?= ... ?>` special tag
  - `<script language="php">...</script>` script tag
- PHP tags tell PHP to interpret the code between them

} *Removed as from 7.0*

```
<?php
    echo "Welcome to Web Application Development II (COSC405)";
?>
```



## Basic Syntax: Instruction Separation

- Instructions are separated with semi-colon ";"
- Optional if it is the last statement immediately before PHP closing tag.

```
<?php
    echo "Welcome to Web Application Engineering II";
    echo "Course code COSC405"
?>
```



## Basic Syntax: PHP Operating Modes

- A typical PHP file (with .php extension) can contain mixed content.
- For example HTML mark-up and PHP code.

```

<html>
  <head><title>HTML + PHP</title></head>
  <body>
    <?php
      $x = "Welcome to COSC405";
      echo "<h1>$x</h1>";
    ?>
  </body>
</html>

```

Diagram illustrating the structure of a PHP file with mixed content:

- HTML mark-up (enclosed in curly braces on the right) includes the `<html>`, `<head>`, `<title>`, `</title>`, `</head>`, `<body>`, `</body>`, and `</html>` tags.
- PHP code (enclosed in curly braces on the right) includes the `<?php` opening tag, the variable assignment `$x = "Welcome to COSC405";`, and the `echo` statement `echo "<h1>$x</h1>";`.



## Basic Syntax: PHP Operating Modes (cont.)

- PHP interpreter has two operating modes:
  - Copy Mode
  - Interpret (Process) Mode
- In copy mode, PHP interpreter outputs what ever it finds.
- In interpret mode, PHP interpreter executes each statement it finds.
- Jumping in and out of copy/interpret mode offers better performance than outputting huge amounts of HTML through PHP.
- Ultimately the final output from PHP interpreter is always a plain (e.g. HTML) file



## Basic Syntax: PHP Operating Modes (cont.)

<pre>&lt;html&gt;   &lt;head&gt;&lt;title&gt;HTML + PHP&lt;/title&gt;&lt;/head&gt;   &lt;body&gt;     &lt;?php       \$x = "Welcome to COSC405";       echo "&lt;h1&gt;\$x&lt;/h1&gt;";     ?&gt;     \$x = "Welcome to COSC405";     echo "&lt;h1&gt;\$x&lt;/h1&gt;";   &lt;/body&gt; &lt;/html&gt;</pre>	<div style="font-size: 3em; line-height: 1;">}</div> <div style="font-size: 3em; line-height: 1;">}</div> <div style="font-size: 3em; line-height: 1;">}</div>	<p>Copy Mode</p> <p>Interpret Mode</p> <p>Copy Mode</p>
--	--	---



## Basic Syntax: PHP Operating Modes (cont.)

- Output from PHP interpreter:

```
<html>
  <head><title>HTML + PHP</title></head>
  <body>
    <h1>Welcome to COSC405</h1>
    echo "<h1>$x</h1>";
  </body>
</html>
```

### The Result of Interpreting:

```
$x = "Welcome to COSC405";
echo "<h1>$x</h1>";
```

```
$x = "Welcome to COSC405";
```



## Basic Syntax: PHP Operating Modes (cont.)

- Output from web browser:



## Basic Syntax: PHP Comments

- Comments are not interpreted/executed by the PHP interpreter.
- Comments are used to make code easy to understand.
- PHP supports three (3) styles of comment

### One-Line Comment (C++ or shell style):

- any character after `"/"` or `"#"` is considered a comment.
- effective up to the end of the line or PHP closing tag.
- cannot span multiple lines – each line of comment must start with `"/"` or `"#"`.

```
<?php
    //This is a one-line(C++ style) comment
    #Yet this is another one-line(shell style) comment
    echo "Welcome to Web Application Engineering II";
?>
```



## Basic Syntax: PHP Comments (cont.)

### Multi-Line Comment (C Style):

- characters between “/\*” and **first occurrence** of “\*/” are considered as a comment
- can span multiple lines
- can be placed in between expressions
- does not support nesting

```
<?php
    /*
        This is a multi-line (C style) comment
        It will not be outputted by the interpreter
    */
    echo "Welcome to Web Application Engineering II";
?>
```



## Basic Syntax: Datatypes

- PHP supports eight (8) primitive datatypes:
  - Integer
  - Float (double)
  - Boolean
  - String
  - Array
  - Object
  - Resource
  - Null
- Datatype determines how data is handled/manipulated by PHP interpreter

## Basic Syntax: Datatypes (Integer)

Dept of Computer Science  
Ahmadu Bello University



- In PHP, integers are signed whole numbers (positive or negative)
- The size of an integer is interpreter/platform-dependent. But common range is between -2,147,483,648 and +2,147,483,647

$$\mathbb{Z} = \{-2147483648, \dots, -2, -1, 0, 1, 2, \dots, 2147483647\}$$

- Integer overflow occurs when the number assigned to a variable is beyond the bounds of integer type.
- Some operations on integer variables result in integer overflow
- Integer overflow causes PHP to automatically convert the result type to float.

```
<?php
    $x = 2147483647;
    var_dump($x);
    $x = $x+1;
    var_dump($x);

?>
```

06-Nov-16

Umar Ibrahim Enesi

15

## Basic Syntax: Datatypes (Integer)

Dept of Computer Science  
Ahmadu Bello University



- Integer size (in bytes) can be determined using the constant PHP\_INT\_SIZE.

```
<?php echo PHP_INT_SIZE; ?>
```

- Maximum value using the constant PHP\_INT\_MAX.

```
<?php echo PHP_INT_MAX; ?>
```

- From PHP 7.0.0, minimum value using the constant PHP\_INT\_MIN

```
<?php echo PHP_INT_MIN; ?>
```

**Note:** Integers can also be represented in binary, decimal, octal and hexadecimal form.

06-Nov-16

Umar Ibrahim Enesi

16





## Basic Syntax: Datatypes (Float)

- Float type are recognized by the presence of decimal point (.)  
e.g. 3.50, -2.0...
- Just like integer, the range of float value is interpreter/platform-dependent. But a maximum of  $\sim 1.8e308$  is a common value on 64-bit systems
- Syntaxes:

DNUM	[+-]?((( [0-9]*[\.] [0-9]+)   ([0-9]+[\.] [0-9]*))
EXPONENTIAL	([0-9]+ DNUM) [eE] [+-]? [0-9]+



## Basic Syntax: Datatypes (Boolean)

- Boolean represent truth values
- Boolean type can only be of two possible values: TRUE or FALSE
- Boolean values are case-insensitive (True, true, TRUe, trUe...)
- Some PHP operators (eg ==, && ...) always return values of type Boolean



## Basic Syntax: Datatypes (String)

- String is a sequence of characters.
- A character is internally represented by 1 byte. What's the limitation?
- Strings can be declared literarily using
  - single quote
  - double quote
  - heredoc
  - nowdoc



## Basic Syntax: Datatypes (String)

- Single quoted string:
  - Improves performance than other styles of creating strings
  - Supports escape sequence for single quote and backslash character only

```
<?php
    $coursecode='COSC405';
    $coursetitle='Web Application Engineering II';
    $remark = 'Students\' favourite course';
?>
```



## Basic Syntax: Datatypes (String)

- Double quoted string:
  - Variables are expanded
  - Supports more escape sequence for special characters than single quoted string

```
<?php
    $coursecode="COSC405";
    $coursetitle="Web Application Engineering II";
    $remark = "Students' favourite course";
?>
```



## Basic Syntax: Datatypes (String)

- Strings in double quotes can be parsed:
  - **Simple syntax:** If a dollar sign (\$) is encountered, the parser will try to form valid variable
  - **Complex syntax:** Within a string, wrap an expression in curly braces (**{ and }**)
- Every character within a string can be identified by a zero-based offset.
- This makes it possible to access or modify a character in the form of an array. (i.e. [] or {} )
- Two or more strings may be joined (concatenated) together using the dot (.) operator.



## Basic Syntax: Datatypes (String)

- Common String functions:

- `echo()` : output one or more strings
- `print()` : output a string
- `strlen()` : get the length of a string
- `str_word_count()` : get the number of words in a string
- `strtolower()` : converts all characters in a string to its lowercase equivalent
- `strtoupper()` : converts all characters in a string to its uppercase equivalent
- `substr()` : get part of a string specified by an index and length
- `str_replace()` : replace text within a string
- `str_pos()` : find the position of a part of a string, within a larger string
- `trim()` : remove trailing spaces (or characters) from both end of a string



## Basic Syntax: Datatypes (Array)

- An array is a variable that stores many values
- Array is analogous to list
- Array size is limited by the amount of memory allocated to the script containing the array
- Each value is associated with a key
- Keys can be string or integer
- Values can be of any type (including arrays too)



## Basic Syntax: Datatypes (Array)

### Creating Array:

```
array( key1 => value1, key2 => value2, key3 => value3, ... )
```

**Note:** Keys are optional

### Example:

```
array("COSC400", "COSC401", "COSC405")

array(
    "COSC400"=>"Project",
    "COSC401"=>"Algorithm and Complexity Analysis",
    "COSC405"=>"Web Application engineering II",
)
```



## Basic Syntax: Datatypes (Array)

### Indexed Arrays

- Keys are **all** integers
- If the key is not specified, PHP will use a higher integer value next to the largest previously used integer key (if any otherwise 0).
- When declaring or adding elements to an indexed array, the following key type conversion takes place:
  - Strings containing valid integers will be cast to the integer type
  - Floats are cast to integers (fractional part truncated)
  - Booleans are cast to integers (true to 1 and false to 0)

```
array("COSC400", "COSC401", "COSC405")
//is the same as...
array(0=>"COSC400", 1=>"COSC401", 2=>"COSC405")
```



# Basic Syntax: Datatypes (Array)

## Associative Arrays

- Keys are all strings
- Empty string ("" ) can be used as a key
- When declaring or adding elements to an indexed array, the following key type conversion takes place:
  - Null will be cast to the empty string

Example:

```
array(
    "COSC400"=>"Project",
    "COSC401"=>"Algorithm and Complexity Analysis",
    "COSC405"=>"Web Application engineering II",
)
```



# Basic Syntax: Datatypes (Array)

## Accessing Arrays

Syntax:

```
array[key]
```

Example:

```
$courses = array(
    "COSC400"=>"Project",
    "COSC401"=>"Algorithm and Complexity Analysis",
    "COSC405"=>"Web Application engineering II",
);
echo $courses[1]; //outputs Error (Undefined offset)
echo $courses["COSC405"]; //Outputs Web Application engineering II
```



# Basic Syntax: Datatypes (Array)

## Modifying/Creating Arrays

### Syntax:

```
array[key] = value
```

### Example:

```
$courses = array(
    "COSC400"=>"Project",
    "COSC401"=>"Algorithm and Complexity Analysis",
    "COSC405"=>"Web Application engineering II",
);
$courses['COSC400']= "Final Project";
echo $courses["COSC400"];
```

*Note: This syntax can also be used to create array*



# Basic Syntax: Datatypes (Array)

## • Some common Array Functions:

- `count()` : Get the number of elements in an array
- `implode()` : make a string out of an array
- `explode()` : make an array out of a string
- `in_array()` : check the existence of a value in an array
- `array_unique()` : return a set of values
- `sort()` : re-organize the values of an array
- `array_diff()` : compute the difference between two arrays
- `list()` : assign values to variables as if they were array



## Basic Syntax: Datatypes (Object)

- An object is a container that contains data (properties) and functions
- Use `new` operator to declare an object
- Classes are defined by `class` keyword (similar to java language)

```
<?php
    class Student{
        private $regNum="U13CS9999";
        public function getRegNum(){ return $this->regnum; }
    }
    $std = new Student();
    echo $std->getRegNum();

?>
```

06-Nov-16

Umar Ibrahim Enesi

31



## Basic Syntax: Datatypes (Resource and Null)

- A resource holds reference to external resources(e.g. database connection, file handle, etc.)

```
$con = mysql_connect($host, $username, $password);
```

- Null represent no value;
- A variable is considered to be `null` if:
  - It is declared but not assigned to any value
  - It is explicitly set to null by calling `unset()`
  - It is explicitly assigned to `null`
- Such variable is said to be *Undefined* and will issue a Notice when certain operations are performed on them.
- Undefined variables can be confirmed by calling `isset()` or `is_null()`

06-Nov-16

Umar Ibrahim Enesi

32





## Basic Syntax: Type Conversion

- Remember PHP is loosely typed – type are not specified when variables are declared
- The context in which a variable is used determines the type of the variable

```
<?php
    $x = "404"; //String context $x is of type string
    $y = $x+1;  //Arithmetic context $x is of type integer
?>
```



## Basic Syntax: Type Conversion

- The type of a variable can be explicitly converted (casting)
- The name of the desired type is written in parentheses before the variable which is to be cast

```
<?php
    $x = 405; //$x is integer
    $y = (string) $x; //$y will hold "405", $x will still hold 405
?>
```

- Explicit type conversion can also be achieved using `settype()` function

```
<?php
    $x = 405; //$x is integer
    settype($x, "string"); //$x becomes string
?>
```

# Type Conversion Example

Complete the table with the appropriate values (see examples)

	CONTEXT			
	Integer	Float	Boolean	String
0	0			
"false"		0.00		
"true"			true	
"cosc405"				"cosc405"
3.84			true	
true		1.00		
"400 Level"				
345	345			
-1.7E-4				"0.00017"

# Basic Syntax: Variables

- Variables are containers for temporarily storing data
- Variables are represented by a name preceded by a dollar sign (\$)
- Naming Rule:
  - Must begin with a letter or underscore
  - Subsequent characters can be a combination of letters, underscores and numbers.
  - Eg. \$Courses, \$student\_name, \$courseTitle ...
- Variable names are case-sensitive
  - \$Courses is not the same as \$courses
- Variables are loosely typed (type is determined by containing value)



## Basic Syntax: Variable Variables

- PHP allows variable to be referenced using other variable's value.

```
<?php
    $Cosc400 = "Course";
    $Course = "Project";
    echo $$Cosc400; //what will be the output?
?>
```



## Basic Syntax: Variable Variables

- Care has to be taken when using variable variables with arrays

```
<?php
    $coursecode = array("COSC400", "COSC405", "COSC408");
    $CODE= "coursecode";
    $COSC400="Project";
    $COSC405="Web Application Engineering II";
    $COSC408="Compiler Construction";
    $o = "DEVELOPER CONFUSION :(";
    echo $$coursecode[1]; //what will be the output?
    echo ${$CODE}[1]; //what will be the output?
    echo $$CODE[1]; //what will be the output?
?>
```



# Basic Syntax: Predefined Variables

- PHP makes available, some predefined variables when a script is running:
  - \$GLOBALS
  - \$\_POST
  - \$\_GET
  - \$\_REQUEST
  - \$\_FILES
  - \$\_COOKIE
  - \$\_SESSION
  - \$\_ENV
  - \$\_SERVER
  - \$php\_errormsg
  - \$HTTP\_RAW\_POST\_DATA
  - \$http\_response\_header
  - \$argc
  - \$argv

*The information stored in these variables are extracted from HTTP request message and various server configuration files*



# Basic Syntax: Variable Scope

- Variable scope determines where a variable is defined (recognised)
- Two type of scope exists:
  - global
  - local
- **superglobals** are predefined variables that are available in all scopes

\$GLOBALS	\$_FILES
\$_POST	\$_COOKIE
\$_GET	\$_SESSION
\$_REQUEST	\$_ENV
\$_SERVER	



## Basic Syntax: Global Scope

- A variable is said to be in a global scope if it is declared in a script but not in any function definition
- Variables with global scope are NOT recognised in structures like functions.
- To access such variables use *global* keyword or *\$GLOBALS* array
- All predefined variables (including superglobals) have global scope



## Basic Syntax: Local Scope

- A variable is said to be in a local scope if it is declared within a function.
- Such variables are only useful in the function and cannot be accessed from outside.
- The life cycle of local variables end when the function finish executing.
- Use *static* keyword to cause a variable retain its value after the function finished executing.



## Basic Syntax: Variable Assignment

- By default variables are assigned by value (copy)

```
<?php
    $x = 2;
    $y = $x; //value of $x is copied into $y
    $x = 5;
    echo $y; //outputs 2
?>
```



## Basic Syntax: Variable Assignment

- Variables can be assigned by reference

```
<?php
    $x = 2;
    $y = &$x; //$y will also point to the value of $x
    $x = 5;
    echo $y; //outputs 5
    $y = 10;
    echo $x; //outputs 10
?>
```



## Basic Syntax: Variable Handling Functions

- `gettype()` – To get the datatype of a variable
- `print_r()` – output a human readable string representation of a variable
- `var_dump()` – output information about a variable
- `is_*()` - finds whether a variable is of a type (where \* can be int, string, bool, null, object, real, resource, scalar)



## Basic Syntax: Constants

- As the name implies, constants are meant to hold values that never change
- Constants follow the same naming rule as variables except that it is not preceded by \$
- By convention constant identifiers are always uppercase
- Constants cannot be redefined or unset once created
- Constants are accessible in all scopes
- Constants are case sensitive



## Basic Syntax: Constants

- To declare a constant, use *define()* function

```
<?php
    define("LEVEL", 400);
?>
```

- Constants can also be defined using *const* keyword

```
<?php
    const LEVEL = 400;
?>
```



## Basic Syntax: Expressions

- An expression is anything that has value.
- Almost everything you do in PHP is an expression
- Most basic form of expression are Constants and Variables

```
$course, LEVEL
```

- More complex form of expressions can be functions.

```
function getName(){ return $this->name;}
```

- Constructs involving operators are also expressions

```
$course= "Web Application Engineering II";
$sum = $x +$y;
```





# Basic Syntax: Operators

- PHP operators takes one or more values and yields another value
  - **unary operators** – takes on one value
  - **binary operators** – takes on two values
  - **ternary operators** – takes on three values
- Operators are classified into 11 groups. Check PHP manual for a full list of operators



# Basic Syntax: Arithmetic Operators

- Arithmetic operators are used to do basic arithmetic on values.

Operator	Meaning	Example
-	Negation	-\$x
+	Addition	\$x + \$y
-	Subtraction	\$x - \$y
*	Multiplication	\$x * \$y
/	Division	\$x \ \$y
%	Modulus	\$x % \$y
**	Exponentiation	\$x ** \$y



# Basic Syntax: Comparison Operators

- Comparison operators are used to compare two values.
- Most comparison operators return value of type boolean

Operator	Meaning	Example
==	Equals (after implicit conversion)	\$x == \$y
===	Equals (in value and type)	\$x === \$y
!= or <>	Not equals (after implicit conversion)	\$x != \$y
!==	No equals (in value or type)	\$x !== \$y
<	Less than	\$x < \$y
<=	Less than or equals	\$x <= \$y
>	Greater than	\$x > \$y
>=	Greater than or equals	\$x >= \$y



# Basic Syntax: Ternary Operator

- Ternary operator is a special comparison operator that selects between two expressions based on the truth value of its condition
- Syntax:  
`condition ? true_expr : false_expr`
- If the truth value of `condition` is TRUE, `true_expr` is evaluated otherwise `false_expr` is evaluated



## Basic Syntax: Logical Operators

- Logical operators are used to evaluate truth values
- The result is always boolean TRUE or FALSE

Operator	Meaning	Example
and	AND	\$x and \$y
&&	AND	\$x && \$y
or	OR	\$x or \$y
	OR	\$x    \$y
!	NOT	!\$y
xor	XOR	\$x xor \$y

06-Nov-16

Umar Ibrahim Enesi

53



## Basic Syntax: Increment Operators

- Increment operators are unary operators that increment or decrement values by one.

Operator	Meaning	Pre-	Post-
++	increment	++\$x	\$x++
--	decrement	--\$y	\$y--

- When used alone, pre-increment and post-increment yield the same result

06-Nov-16

Umar Ibrahim Enesi

54



# Basic Syntax: Other Operators

Operator class	Operator (s)
String operator	.
Assignment operators	=, +=, -=, *=, **=, /=, .=, %=, &=,  =, ^=, <<=, >>=, =>
Error control operator	@
Type operator	instanceof

*Check out PHP manual for more operators*



# Basic Syntax: Operator Precedence and Associativity

- PHP has a way of resolving ambiguity when two or more operators are used in an expression
- Associativity is applicable where two or more operators within an expression have equal precedence
- Associativity determines in what direction operators with equal precedence are to be considered.
- Operators can either be left associative, right associative or non-associative
- Check out PHP manual for full description



## Key Points

- PHP code is stored in a file with .php extension
- `<?php ... ?>` - Recommended php tag
- PHP has two operating modes (copy and interpret)
- Variable names are case-sensitive and loosely typed
- PHP has two scopes – global and local
- PHP has eight (8) primitive types
- Type conversion is automatic except when done explicitly by casting or using `settype()` function
- Operators can be unary, binary or ternary
- Ambiguities orchestrated by the presence or two or more operators are resolved by operator precedence and associativity



## References

- PHP Manual
- Murach PHP and MySQL
- <http://www.w3schools.com>
- <http://www.php5-tutorial.com>
- <http://www.tutorialspoint.com>