PHP
Hypertext **P**re-processor

NOR ANITA FAIROS BINTI ISMAIL

Contents

- Lesson 1: Introduction
- Lesson 2: PHP Syntax
- Lesson 3: PHP Variables
- Lesson 4: PHP Operators
- Lesson 5: PHP Conditional Statements
- Lesson 6: PHP Loops
- Lesson 7: PHP Arrays
- Lesson 8: PHP Functions
- Lesson 9: PHP Form

Lesson 1: Introduction

- Lesson Outline
  - Introduction
  - What You Should Already Know
  - What is PHP?
  - What Can PHP Do?
  - Why PHP?
  - How does PHP work?
  - What is a PHP File?

Introduction

- PHP is a **server side scripting language**, and is a powerful tool for making dynamic and interactive Web pages.

- PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP

What You Should Already Know

- Before we continue, you should have a basic understanding of the following:
  - HTML
  - CSS
  - JavaScript

What is PHP?

- **PHP** was originally an acronym for **P**ersonal **H**ome **P**ages, but is now PHP stands for **P**HP: **H**ypertext **P**re-processor
- PHP is a widely-used, open source scripting language
- PHP scripts are **executed on the server**
- PHP is free to download and use

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can restrict users to access some pages on your website
- PHP can encrypt data

Why PHP?

- PHP runs on different platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP has support for a wide range of databases
- PHP is free. Download it from the official PHP resource: **www.php.net**
- PHP is easy to learn and runs efficiently on the server side

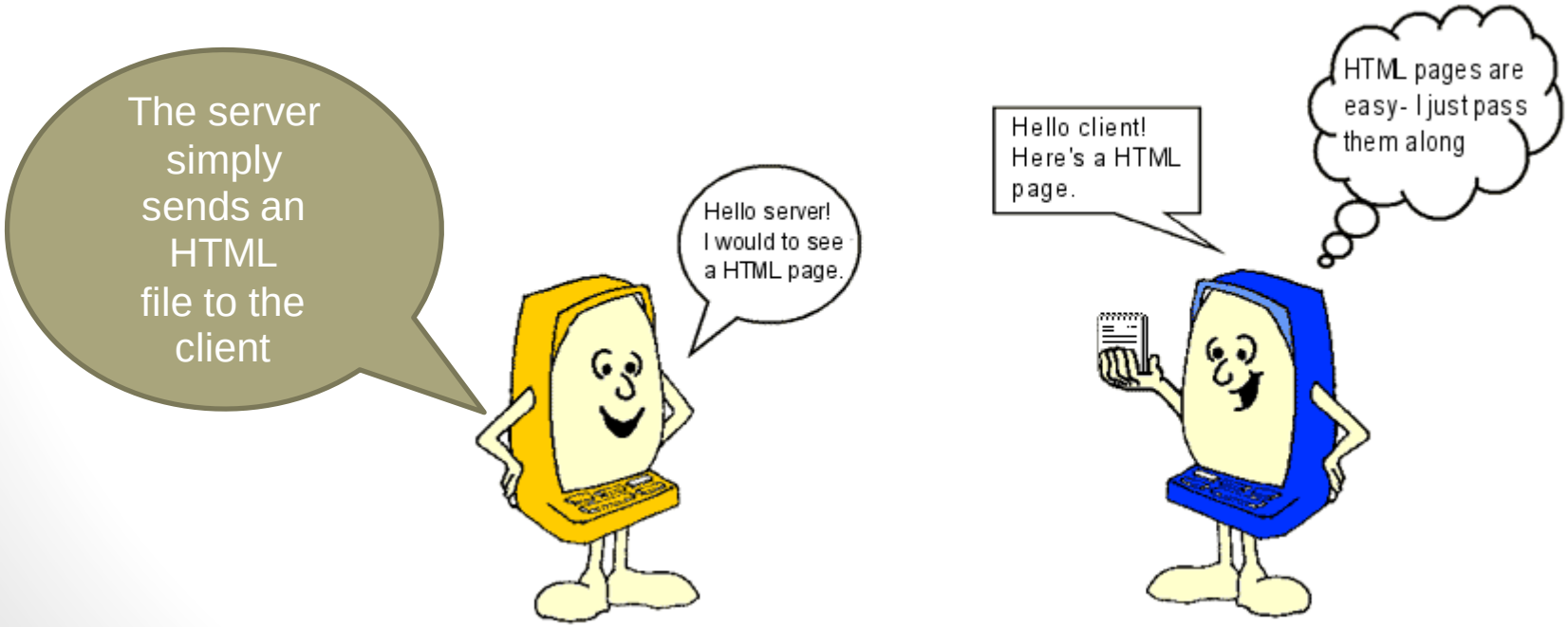
How does PHP work?

- PHP scripts are **executed on the server**, before the web page is displayed to the user (this is what mean by "server-side")
- The user only sees the **end result**, we which consists of client-side markup and scripts (i.e. HTML, JavaScript, CSS etc)

How does PHP work?

Requesting HTML page

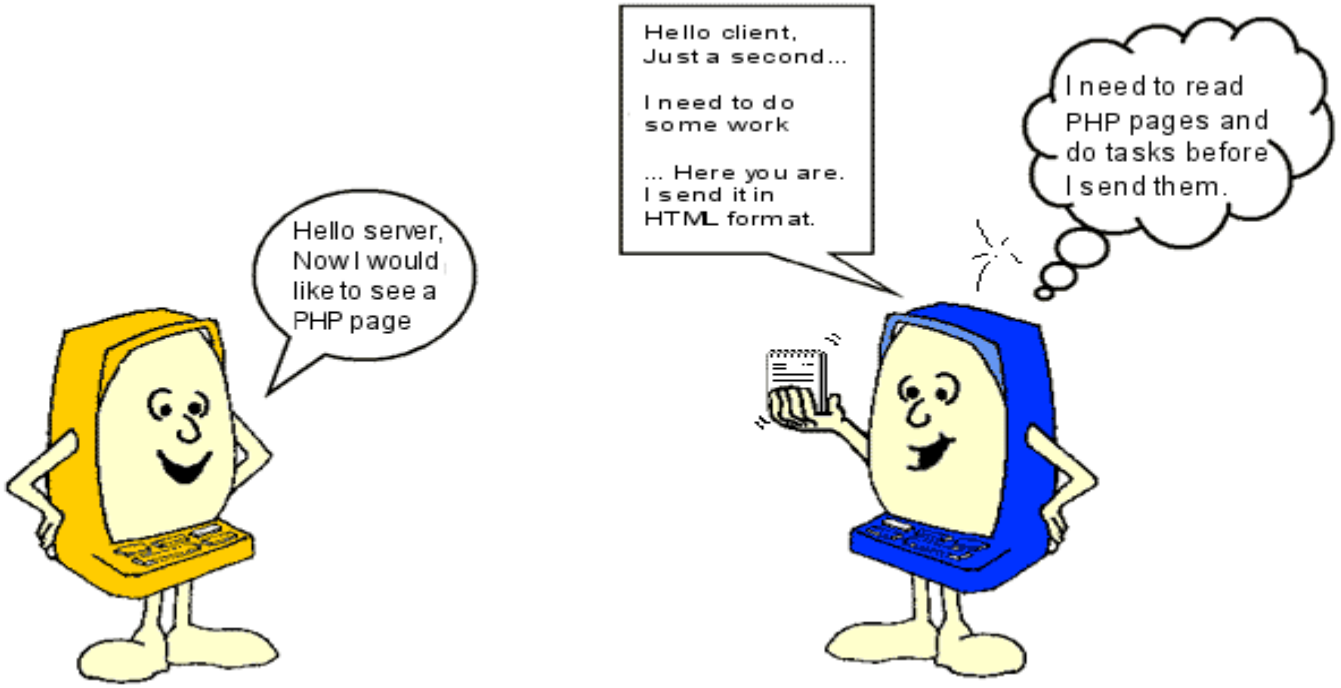
- Imagine you type the address of an HTML document in the browser. (e.g. **http://www.mysite.com/page.htm**) in the address



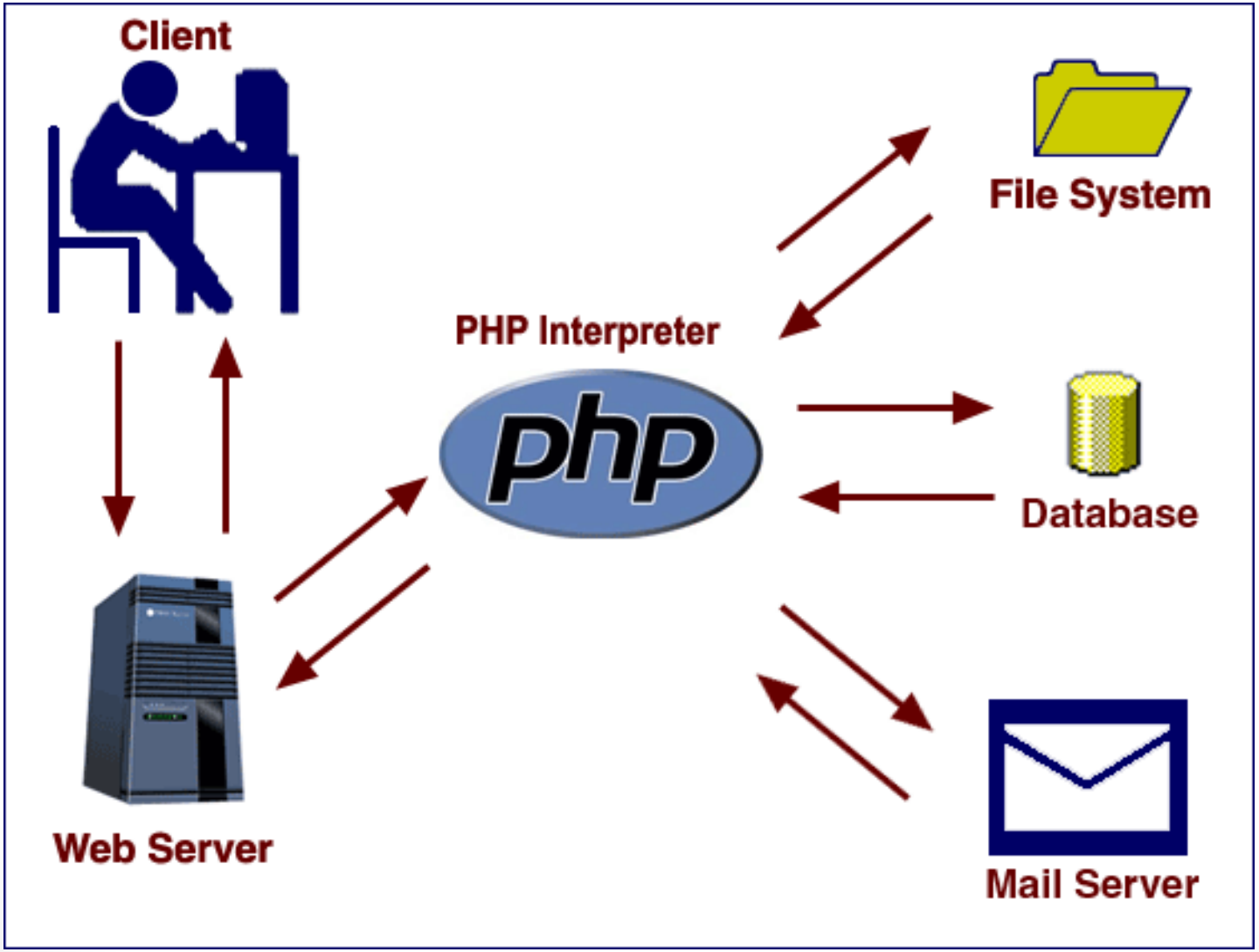
How does PHP work?

Requesting PHP page

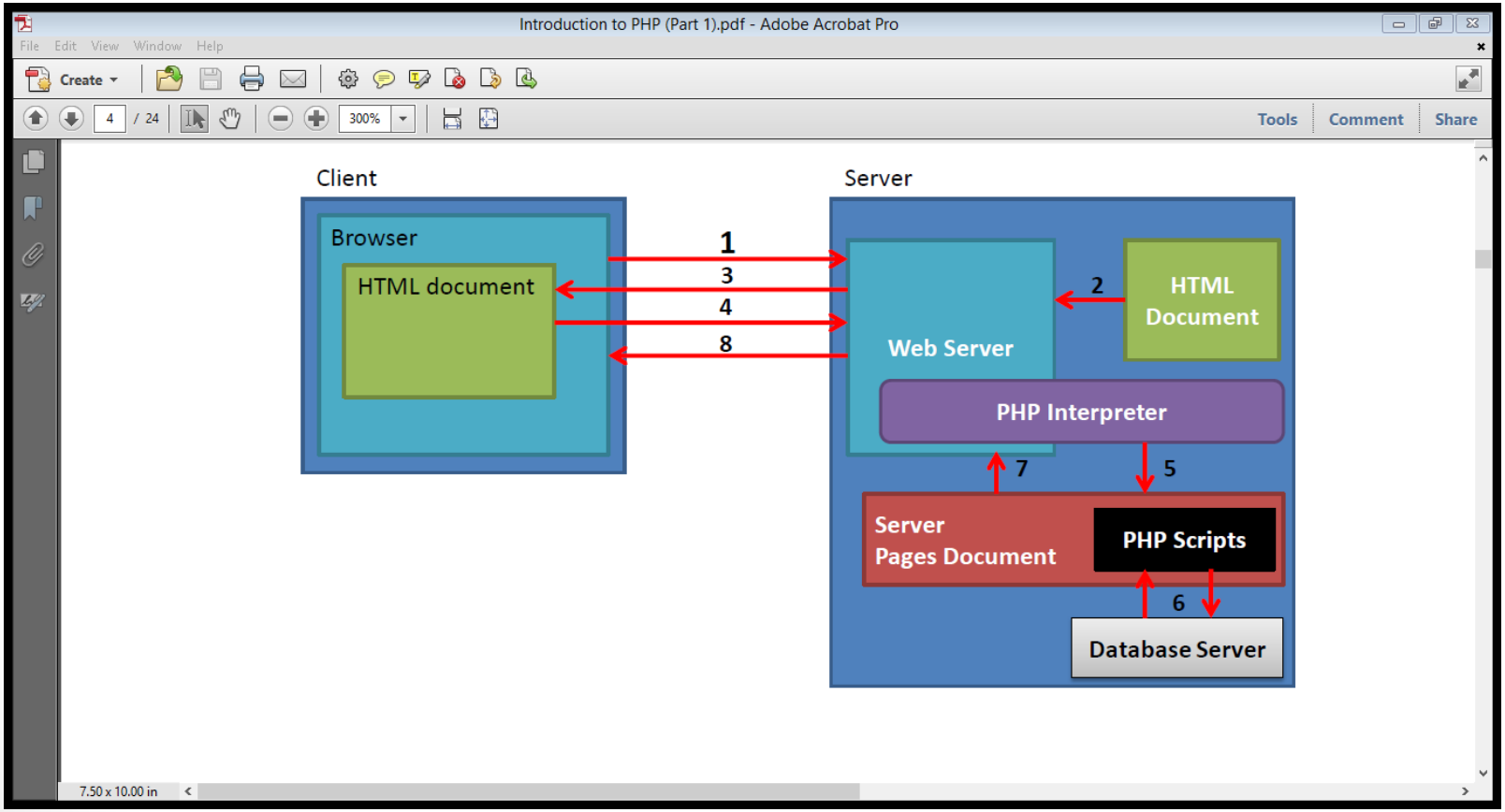
- If you type **http://www.mysite.com/page.php** and thus request an **PHP page** - the server is put to work:



How does PHP work?



How does PHP work?



How does PHP work?

- Whenever the server processes a file with the ***.php*** extension, it knows to look for PHP code
- When it encounters the PHP code, it processes it
- Generally, the same ***.php*** file will also have **client side code** such as HTML
- The server knows to process the PHP bits and output the client-side bits

What is a PHP File?

- PHP files can contain text, HTML, JavaScript code, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain HTML
- PHP files have a default file extension of **".php"**

What Do I Need?

- PHP is a *server-side* technology.
- Therefore, you need to have a **server** to run PHP.
- But it doesn't need to cost you anything to make this upgrade and there are several options for doing so.

  - Option 1: Website on a hosted server
  - Option 2: Install PHP on your computer
  - Option 3: Set Up PHP on Your Own PC
  - XAMPP, EasyPHP , Aprelium - Abyss **Web Server**, Zend Server Free, WampServer

What Do I Need?

Option 1: Website on a hosted server

- You can choose to have a website on a host that supports PHP.
  - If your server has activated support for PHP you do not need to do anything.
  - Just create some .php files and test whether your host supports PHP
  - If you don't already have a website on hosted server you can create a free account on 000webhost.com which supports PHP.
  - **You already have a GMM (SCSV) Student server.**

    http://gmm-student.fc.utm.my/~nafbti

What Do I Need?

Option 2: Install PHP on your computer

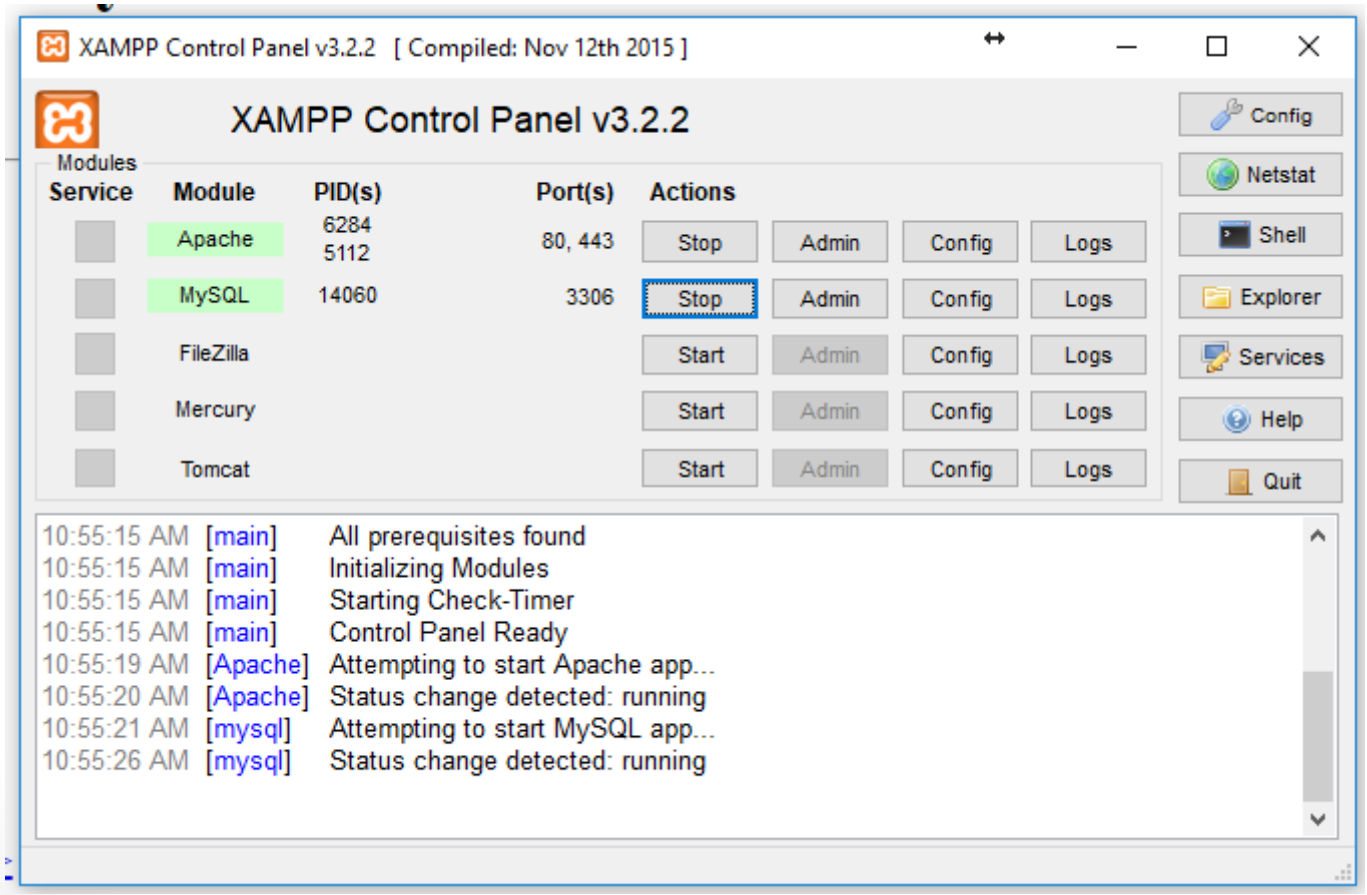
- However, if your server does not support PHP, you must:
  - Install a web server
  - Install PHP
  - Install a database, such as MySQL
- The official PHP website (PHP.net) has installation instructions for PHP: http://php.net/manual/en/install.php

MYSQL Database

- Login & password => same as web server account
- phpMyAdmin =>
  - http://gmm-student.fc.utm.my/phpMyAdmin/index.php
- Database name =>
  - db_nafbti

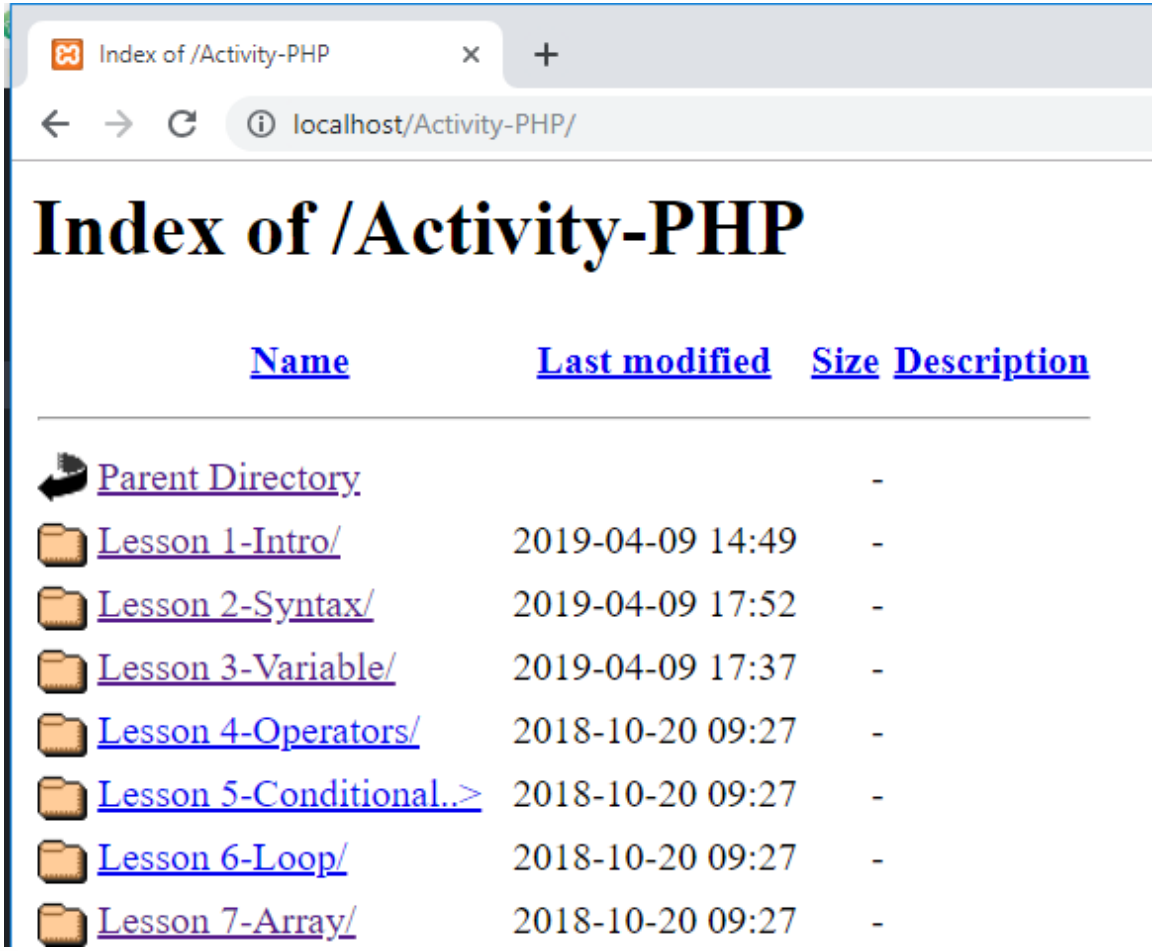
Start XAMPP Server

Start XAMPP Control Panel



Access Localhost Server

http://localhost/Activity-PHP/



Lesson 2: PHP Syntax

- Lesson Outline
  - Basic PHP Syntax
  - Comments in PHP

Basic PHP Syntax

- A PHP script can be placed **anywhere** in the document.
- A PHP script starts with **<?php** and ends with **?>**
- The default file extension for PHP files is **".php".**

```
<?php
// PHP code goes here
?>
```

- A PHP file normally contains **HTML tags**, and **some PHP** scripting code.
- Each code line in PHP must end with a **semicolon (;).**
- This tells the server that a particular statement has finished.

Basic PHP Syntax

Lesson 2a.php

- With PHP, there are two basic statements to output text in the browser: **echo** and **print**.
- Example: A simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page

```
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

Display the output:
Echo vs print

- We may use **"echo"** command instead of **"print",** to output string and the result will be identical
- echo() and print() are language constructs in PHP, both are used to output strings.
- The speed of both statements is almost the same.
- echo() can take **multiple expressions** whereas print **cannot take multiple expressions**.
- Print return true or false based on success or failure whereas echo doesn't return true or false.

Comments in PHP

Lesson 2b.php

- A comment in PHP code is a line that is **not read/executed** as part of the program.
- Its only purpose is to be read by someone who is editing the code!
- **Three** ways of commenting:

```
// This is a single line comment

# This is also a single line comment

/*
This is a multiple lines comment block
that spans over more than
one line
*/
```

Lesson 2b.php

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

#   This is also a single line comment

/*  This is a multiple lines comment block
    that spans over more than
    one line
*/

echo "Hello World!";
echo "<br>";
echo 'This is a test'; //this is a C++ style comment
echo "<br><p><b>"; //PHP can consist HTML
echo "This is a 2nd test </p></b>"; #this is a Shell-style comment

?>
```

PHP Case Sensitivity

Lesson 2c.php

- In PHP, all user-defined functions, classes, and keywords (e.g. if, else, while, echo, etc.) are **NOT case-sensitive**.
- However; all variables are **case-sensitive**.

Lesson 2c.php

```php
<?php
//all user-defined functions, classes, and keywords are NOT case-sensitive
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";

echo "<br>";

#However; all variables are case-sensitive.
/*In the example below, only the first statement will display
the value of the $color variable (this is because $color, $COLOR,
and $coLOR are treated as three different variables)
*/

$color="red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>
```

Lesson 3: PHP Variables

- Lesson Outline:
  - PHP Variables
  - Rules for PHP variables
  - PHP Variable Scopes
  - PHP String

PHP Variables and Declaring a variable

Lesson 3a.php

- Variables are **"containers"** for storing information.
- PHP has no command for declaring a variable.
- A variable is created the moment you first assign a value to it:

```
<?php

$txt="Hello world!";
$x=5;
$y=10.5;
$z=$x+$y;
echo $z;

?>
```

Output :
15.5

PHP Variables and Declaring a variable

- PHP is a Loosely Typed Language.
- We **did not have to tell** PHP which **data type** the variable is.
- PHP **automatically** converts the variable to the correct data type, depending on its value.

Rules for PHP variables

- A variable starts with the **$** sign, followed by the name of the variable

- A variable name :
  - must begin with a **letter** or the **underscore** character
  - only contain **alpha-numeric characters** and **underscores** (A-z, 0-9, and _)
  - but cannot use characters like **+ , - , % , (,) . & ,** etc
  - should **not contain spaces**
  - are **case sensitive** ($y and $Y are two different variables)

PHP Variable Scopes

- Scope can be defined as the range of availability a variable has to the program in which it is declared.
- The scope of a variable is the part of the script where the variable can be referenced/used.
- PHP has four different variable scopes:
  - local
  - global
  - static
  - Function parameter

PHP Variable: Local and Global Scope

Lesson 3b.php

Local Variable Scope

- declared within a PHP function is **local** and can only be **accessed within** that function

```
<?php
$x=5; // global scope
function myTest()
{
echo $x; // local scope.  Notice: Undefined variable: x
}
myTest();
?>
```

PHP Variable: Global Scope

Lesson 3c - 3d.php

Global Variable Scope

- defined **outside** of any function, has a global scope.
- Global variables can be accessed from **any part** of the script**, EXCEPT** from **within** a function.
- To access a global variable from within a function, use the **global** keyword.
- See Lesson 3c.php

- PHP also stores all global variables in an **array** called **$GLOBALS[index].**
- The **index** holds the **name** of the variable.
- This array is also accessible from **within** functions and can be used to **update global** variables directly.
- See Lesson 3d.php

Lesson 3e.php

PHP Variable: Static Scope

- When a function is completed/executed, all of its variables are normally **deleted**.
- However, sometimes you want a local variable **to not be deleted.**
- We need it for a **further job**.
- To do this,

  use the **static** keyword when you first declare the variable.

```
<?php

function myTest()
{
static $x=0;
echo $x;
$x++;
}
myTest(); // Output: 0
myTest(); // Output: 1
myTest(); // Output: 2
?>
```

PHP Variable: Functions Parameter Scope

Lesson 3f-a.php
Lesson 3f-b.php

- A **parameter** is a **local variable** whose value is passed to the function by the **calling code**.
- Parameters are declared in a parameter list as part of the function declaration.
- Function parameters are **declared after** the function name and inside parentheses.
- They are declared much like a typical variable would be:

Lesson 3g.php

PHP Variable: String

- String variables are used for values that contain **characters**.
- After we have created a string variable we can manipulate it.
- A string can be used directly in a function or it can be stored in a variable.
- Example:

```
<?php
$txt="Hello world!";
echo $txt;
?>
```

Output:
Hello World!

Lesson 3h.php

String Concatenation Operator

- There is only one string operator in PHP.
- The concatenation operator **(.)**,is used to **join two string values together**.
- To concatenate two string variables together, use the dot (.) operator.

```
<?php
$txt1="Hello world!";
$txt2="What a nice day!";
echo $txt1 . " " . $txt2;
?>
```

The PHP String function:
strlen() function

- The strlen() function is used to **find the length** of a string.
- Let's find the length of our string "Hello world!":

```
<?php
echo strlen("Hello world!");
?>
```

- The Output will print: 12

The PHP String function:
strpos() function

- The **strpos()** function is used t**o search for a character or a specific text within a string**.
- If a match is found, it will return the character position of the first match. If no match is found, it will return FALSE.
- Let's see if we **can find the string "world"** in our string:

```
<?php
echo strpos("Hello world!","world");
?>
```

Lesson 4: PHP Operators

- Lesson Outline:
  - Arithmetic Operators
  - Comparison Operators
  - Logical (or Relational) Operators
  - Assignment Operators
  - Conditional (or ternary) Operators

Type of PHP operators

- PHP operators are characters (or sets of characters) that perform a special operation within the PHP code.
- PHP language supports following type of operators:
  - Arithmetic Operators
  - Comparison Operators
  - Logical (or Relational) Operators
  - Assignment Operators
  - Conditional (or ternary) Operators

Lesson 4a.php

PHP Arithmetic Operators

Operator	Name	Description	Example	Result
x + y	Addition	Sum of x and y	2 + 2	4
x - y	Subtraction	Difference of x and y	5 - 2	3
x * y	Multiplication	Product of x and y	5 * 2	10
x / y	Division	Quotient of x and y	15 / 5	3
x % y	Modulus	Remainder of x divided by y	5 % 2 10 % 8 10 % 2	1 2 0
- x	Negation	Opposite of x	- 2	
a . b	Concatenation	Concatenate two strings	"Hi" . "Ha"	HiHa

Lesson 4b.php

Assignment Operators

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus
a .= b	a = a . b	Concatenate two strings

Lesson 4c.php

PHP String Operators

Operator	Name	Example	Result
.	Concatenation	$txt1 = "Hello" $txt2 = $txt1 . " world!"	Now $txt2 contains "Hello world!"
.=	Concatenation assignment	$txt1 = "Hello" $txt1 .= " world!"	Now $txt1 contains "Hello world!"

Lesson 4d.php

Comparison Operators

Operator	Name	Description	Example
x == y	Equal	True if x is equal to y	5==8 returns false
x === y	Identical	True if x is equal to y, and they are of same type	5=="5" returns false
x != y	Not equal	True if x is not equal to y	5!=8 returns true
x <> y	Not equal	True if x is not equal to y	5<>8 returns true
x !== y	Not identical	True if x is not equal to y, or they are not of same type	5!=="5" returns true
x > y	Greater than	True if x is greater than y	5>8 returns false
x < y	Less than	True if x is less than y	5<8 returns true
x >= y	Greater than or equal to	True if x is greater than or equal to y	5>=8 returns false
x <= y	Less than or equal to	True if x is less than or equal to y	5<=8 returns true

PHP Incrementing/ Decrementing Operators

Lesson 4e.php

Introduction to PHP (Part 1).pdf - Adobe Acrobat Pro

Operator	Name	Description
++ x	Pre-increment	Increments x by one, then returns x
x ++	Post-increment	Returns x, then increments x by one
-- x	Pre-decrement	Decrements x by one, then returns x
x --	Post-decrement	Returns x, then decrements x by one

Lesson 4f.php

PHP Logical Operators

Operator	Name	Description	Example
x and y	And	True if both x and y are true	x=6 y=3 (x < 10 and y > 1) returns true
x or y	Or	True if either or both x and y are true	x=6 y=3 (x==6 or y==5) returns true
x xor y	Xor	True if either x or y is true, but not both	x=6 y=3 (x==6 xor y==3) returns false
x && y	And	True if both x and y are true	x=6 y=3 (x < 10 && y > 1) returns true
x || y	Or	True if either or both x and y are true	x=6 y=3 (x==5 || y==5) returns false
! x	Not	True if x is not true	x=6 y=3 !(x==y) returns true

Lesson 4g.php

Conditional Operator

Operator	Description	Example
? :	Conditional Expression	If Condition is true ? Then value X : Otherwise value Y

Lesson 5: PHP Conditional Statements

- Lesson Outline:
  - if statement
  - if...else statement
  - if...else if....else statement
  - switch statement

PHP Conditional Statements

- Conditional statements are used to perform different actions based on different conditions.
- In PHP we have the following conditional statements:
  - **if statement** - executes some code only if a specified condition is true
  - **if...else statement** - executes some code if a condition is true and another code if the condition is false
  - **if...else if....else statement** - selects one of several blocks of code to be executed
  - **switch statement** - selects one of many blocks of code to be executed

Lesson 5a.php

The if Statement

- The if statement is used to execute some code **only if a specified condition is true**.
- Syntax

```
if (condition)
{
code to be executed if
condition is true;
}
```

```
$age = 20;
 if( $age > 18 )
  {
   echo "<b>Qualifies for
    driving</b>";
   }
```

Lesson 5b.php

The if...else Statement

- Use the if....else statement to execute some code **if a condition is true and another code if the condition is false**.
- Syntax

if (*condition*)
{ code to be executed if condition is true;
}
else
{ code to be executed if condition is false;
}

```php
<?php

$age = 15;
if( $age > 18 ){
    echo "<b>Qualifies
        for driving</b>";
}
else{
    echo "<b>Does not qualify
        for driving</b>";
}

?>
```

Lesson 5c.php

The if...else if....else Statement

- Use the if....else if...else statement to **select one of several blocks of code to be executed**.
- Syntax

```
if (condition)
{
 code to be executed if condition is true;
 }
 else if (condition)
{
 code to be executed if condition is true;
}
else
{
code to be executed if condition is false;
}
```

```php
<?php

$book = "maths";
if( $book == "history" )
{
    echo"<b>History Book</b>";
}
else if( $book == "maths" )
{
   echo"<b>Maths Book</b>";
}
else if( $book == "economics" )
{
   echo"<b>Economics Book</b>";
}
else
{
  echo"<b>Unknown Book</b>";
}

?>
```

The switch Statement

- In the previous lesson we used a PHP *if... Else if* statement in order to execute different block of code for each different condition.
- As mentioned, we could use as many "else if“ is as we like.
- If you have many conditions, PHP *switch* statements are a more efficient way of doing this.
- The server will execute a PHP switch statement quicker than multiple "else if"s.
- Also, there's actually less code for the programmer to write.

Lesson 5d.php

The switch Statement

- Use the switch statement to **select one of many blocks of code to be executed**.
- •Syntax

switch (*n*)
{
case *label1:*
code to be executed if n=label1;
break;
case *label2:*
code to be executed if n=label2;
break;
default:
code to be executed if n is different from both label1 and label2;
}

```
<?php
$grade= "A";
switch ($grade)
{
  case 'A': echo "Good job<br />";
            break;
//          case 'A':case 'a':  document.write("Good job<br />");
  case 'B': echo "Pretty good<br />";
            break;
  case 'C': echo "Passed<br />";
            break;
  case 'D': echo "Not so good<br />";
            break;
  case 'F': echo "Failed<br />";
            break;
  default:  echo "Unknown grade<br />";
}
?>
```

Lesson 6: PHP Loops

- Lesson Outline
- PHP Loops
  - The for Loop
  - The while Loop
  - The do...while Loop
  - Loop Control
  - foreach loop

PHP Loops

- Loops execute a block of code a specified number of times, or while a specified condition is true.
- In PHP, the looping statements:
  - **for** - loops through a block of code a specified number of times
  - **while** - loops through a block of code while a specified condition is true
  - **do...while** - loops through a block of code once, and then repeats the loop as long as a specified condition is true
  - **foreach** - loops through a block of code for each element in an array

Lesson 6a.php

The for Loop

- The for loop is used when you know in advance how many times the script should run.
- Syntax

for (*initialization*; *condition*; *increment*)

{

code to be executed;

}

```
<?php

for ($i = 0; $i <= 5; $i++)
{
echo "The number is " + $i;
echo "<br />";
}

?>
```

Lesson 6b.php

The while Loop

- The while loop executes a block of code while a condition is true.
- Syntax

```
while (condition)
{
code to be executed;
}
```

```
<?php
$i=0;
while ($i<=5)
{
echo "The number is " +
$i;
echo "<br />";
$i++;
}
?>
```

Lesson 6c.php

The do...while

- The do...while statement will always execute the block of **code once**, it will then check the condition, and repeat the loop while the condition is true.
- Syntax

```
Do
{
 code to be executed;
} while (condition);
```

```
<?php
$i = 0;
do
{
echo "The number is " +
$i;
echo "<br />";
$i++;
}
while ($i <= 5);
?>
```

Lesson 6d.php

The foreach Loop

- The foreach loop is used to loop through arrays.
- Syntax

```
foreach ($array as $value)
{
code to be executed;
}
```

```
<?php

// example to list out the values of an array.

$array = array( 1, 2, 3, 4, 5);
Foreach ( $array as $value )
{
  echo "Value is $value <br />";
}
?>
```

Lesson 7: PHP Arrays

- Lesson Outline
  - What is an Array?

What is an Array?

- An array is a special variable, which **can hold more than one value at a time**.
- If you have a list of items (a list of car names, for example), **storing the cars in single variables** could look like this:

```php
<?php
$cars1="Volvo";
$cars2="BMW";
$cars3="Toyota";
?>
```

```php
<?php
$cars=array("Volvo","BMW","Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . "
and " . $cars[2] . ".";
?>
```

- But if you want to store 100 cars then instead of defining 100 variables its easy to define an array of 100 length.

What is an Array?

- An array can hold many values under a **single name**, and you can access the values by referring to **an index number**.

Create an Array in PHP

- In PHP, the array() function is used to create an array:

  array();

- There are three types of arrays:
  - **Numeric /Indexed arrays** - Arrays with numeric index
  - **Associative arrays** - Arrays with named keys
  - **Multidimensional arrays** - Arrays containing one or more arrays **(PHP Advance)**

Lesson 7a.php

Numeric/Indexed Arrays

- An array with a numeric index.
- use a number as the "key".
- The key is the unique identifier, or ID, of each item within the array.
- There are two ways to create indexed arrays:

- The index can be assigned automatically (index always starts at 0):

```
$cars=array("Volvo","BMW","Toyota");
```

- or the index can be assigned manually:

```
$cars[0]="Volvo";
$cars[1]="BMW";
$cars[2]="Toyota";
```

Get The Length of an Array - The count()Function

Lesson 7b.php

- The **count()** function is used to return the length (the number of elements) of an array.

```
<?php
$cars=array("Volvo","BMW","Toyota");
echo count($cars);
?>
```

Output:

3

Loop Through an Indexed Array

Lesson 7c.php

- To loop through and print all the values of an indexed array, you could use a for loop.

```
<?php
$cars=array("Volvo","BMW","Toyota");
$arrlength=count($cars);
for($x=0;$x<$arrlength;$x++)
 {
 echo $cars[$x];
 echo "<br>";
 }
?>
```

Output:
Volvo
BMW
Toyota

Associative Arrays

- The associative arrays are **very similar** to numeric arrays in term of functionality but they are **different** in terms of their index.
- Associative array will have **their index as string "keyname"** so that you can establish a strong association between key and values.
- instead of using a number for the key, Associative arrays use a **value**.
- We then assign **another value** to the key

Lesson 7d.php

Creating Associative Arrays

- There are **two ways** to create an associative array:

```
$arrayName['keyName'] = "Value1";
$arrayName['keyName'] = "Value2";
$arrayName['keyName'] = "Value3“;
```

```
$age['Peter']="35";
$age['Ben']="37";
$age['Joe']="43";
```

```
$arrayName = array("keyName"=>"Value1",
                   "keyName"=>"Value2",
                   "keyName"=>"Value3");
```

```
$age=array("Peter"=>"35",
           "Ben"=>"37",
           "Joe"=>"43");
```

Loop Through an Associative Array

Lesson 7e.php

- To loop through and print all the values of an associative array, you could use a **foreach loop**.

```
<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>
"43");
foreach($age as $x=>$x_value)
 {
 echo "Key=" . $x . ", Value=" . $x_value;
 echo "<br>";
 }
?>
```

Output:

Key=Peter, Value=35
Key=Ben, Value=37
Key=Joe, Value=43

Lesson 7f.php

Multidimensional Arrays

- A multi-dimensional array each element in the main array can also be an array.
- And each element in the **sub-array** can be an array, and so on.
- Values in the multi-dimensional array are accessed using **multiple index**.

PHP Sorting Arrays

Lesson 7g.php
Lesson 7h.php

- The elements in an array can be sorted in alphabetical (A-Z) or numerical order(1-100), descending or ascending.
- PHP array sort functions:
  - **sort()** - sort arrays in ascending order
  - **rsort()** - sort arrays in descending order
  - **asort()** - sort associative arrays in ascending order, according to the value
  - **ksort()** - sort associative arrays in ascending order, according to the keyname
  - **arsort()** - sort associative arrays in descending order, according to the value
  - **krsort()** - sort associative arrays in descending order, according to the keyname

Lesson 8: PHP Functions

- Lesson Outline
  - PHP Functions
  - Creating and call a PHP function
  - PHP Functions with Parameters
  - PHP Functions with retruning value

PHP Functions

- PHP functions are similar to other programming languages.
- A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value.
- A function will be executed by a call to the function.
- You may call a function from anywhere within a page.
- In PHP, there are more than 700 built-in functions.
- There are **two parts** which should be clear to you:
  - Creating a PHP Function
  - Calling a PHP Function

Creating and call a PHP function

Lesson 8a.php

- Start with keyword **function** and
- all the PHP code should be put inside **{** and **}** braces.
- A function will be executed by a call to the function.
- Give the function a name that reflects what the function does
- The function name can start with a letter or underscore (not a number)

Syntax

```
/* Defining a PHP Function */
function
functionName()
{
 code to be executed;
}

/* Calling a PHP Function */
writeMessage();
```

PHP Functions with Parameters

Lesson 8b.php

- To add more functionality to a function, we can add parameters.
- A parameter is just like a variable.
- Parameters are specified after the function name, inside the parentheses.

```
function functionName (parameter1, parameter2,…)
{
 code to be executed;
}
```

PHP Functions with retruning value

Lesson 8c.php

- A function can return a value using the **return** statement in conjunction with a value or object.
- return stops the execution of the function and sends the value back to the calling code.

```
<?php
function addFunction($num1, $num2)
{
  $sum = $num1 + $num2;
  return $sum; //Function which returns value
}
echo "10 + 20 = " . addFunction(10,20);
?>
```

```
$return_value = addFunction(10, 20);
echo "Returned value from the function : $return_value";
```

Add this statement and see the result

Lesson 9: PHP Form and User Input

- Lesson Outline
  - PHP Form Handling
  - HTML Forms and PHP

PHP:
Passing Variables With Forms

- Interactive websites require **input from users**. One of the most common ways to get input is with forms.

```
<form method=" " action=" ">
…….
</form>
```

- When you code a form, there are two particular important attributes : **action** and **method**.

- **Action**
  - Is used to enter the **URL where the form is submitted**. It would be the PHP file that you want to handle the input.

- **Method**
  - Can either have the value **"post" or "get"**, which are two different methods **to pass data**, and to collect form-data.

PHP Form Handling :

Requesting form data with PHP

- The PHP **$_GET** and **$_POST** variables are used to **retrieve information** from forms, like user input. You can use:
  - **$_POST["fieldname"];**
    - returns the value (data) of a field in the form submitted by **post** method
  - **$_GET["fieldname"];**
    - returns the value (data) of a field in the form submitted by **get** method

- The most important thing to notice when dealing with HTML forms and PHP is that **any form element** in an HTML page will **automatically** be available to your PHP scripts.

form01.html

A simple HTML form

- Create a simple HTML form with two input fields and a submit button:
- The form data is sent with the HTTP **POST method**.

```
<html>
<body>

<form name="" action="welcome01.php" method="post">
Name: <input type="text" name="name"><br><br><br>
E-mail: <input type="text" name="email"><br><br><br>
<input type="Submit">
</form>

</body>
</html>
```

Output Result:

Name:

E-mail:

Submit Query

welcome01.php

PHP $_POST variables

- When the user **fills out** the form and **clicks** the submit button, the form data is **sent** for processing to a PHP file named **"welcome01.php"**.
- To display the submitted data you could simply echo all the variables.

```
1  <html>
2  <body>
3
4  Welcome <?php echo $_POST["name"]; ?><br>
5  Your email address is: <?php echo $_POST["email"];?>
6
7  </body>
8  </html>
```

Output Result:

Welcome anita
Your email address is: noranita@utm.my

$_POST["fieldname"];

Will returns the value of a field in the form text box **“name”**

```
<input type="text“  name="name">
```

Name: anita

E-mail: noranita@utm.my

Submit Query

Welcome ANITA brother
Your email address is: anita@utm.my

Rewrite the code

PHP $_POST variables

welcome01.php

action="**welcome01.php**"

- **OR** you also can write **welcome01.php** like this:

```
1   <html>
2   <body>
3
4   <?php
5     echo "Welcome".$_POST["name"];
6     echo "<br>";
7     echo "Your email address is:" .$_POST["email"];
8   ?>
9
10  </body>
11  </html>
```

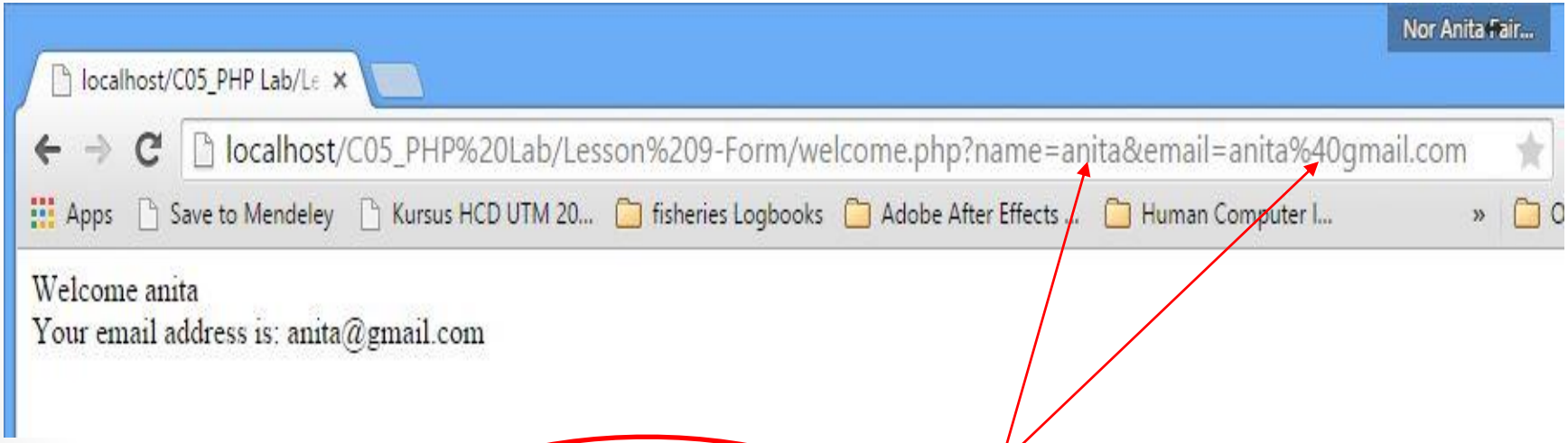
Output Result:

Welcome anita
Your email address is: noranita@utm.my

Then save as welcome02.php and change action="**welcome02.php**"

PHP **$_GET** variables

- Check out your address bar



Display the user input on address bar. No privacy!

Upload to your own web server

- **"http:// gmm-student.fc.utm.my /~nafbti/welcome01.php"**
- Using filezilla

PHP **$_POST** variables

$_POST Variable

- In PHP, the predefined $_POST variable is used to collect values in a form with method="post".
- Information sent from a form with the POST method is **invisible** to others and has no limits on the amount of information to send.

When to use method="post"?

- However, because the variables are not displayed in the URL, it is **not possible to bookmark** the page.

PHP **$_GET** variables

$_GET Variable

- In PHP, the predefined $_GET variable is used to collect values in a form with method="get".
- Information sent from a form with the GET method is **visible** to everyone (it will be displayed in the browser's address bar) and has **limits** on the amount of information to send.

When to use method="get"?

- When using method="get", all variable names and values are **displayed** in the URL.
- This method should not be used when sending **password** or other **sensitive** information!
- However, because the variables are displayed in the URL, it **is possible to bookmark** the page. This can be useful in some cases.

The $_REQUEST variable

See More

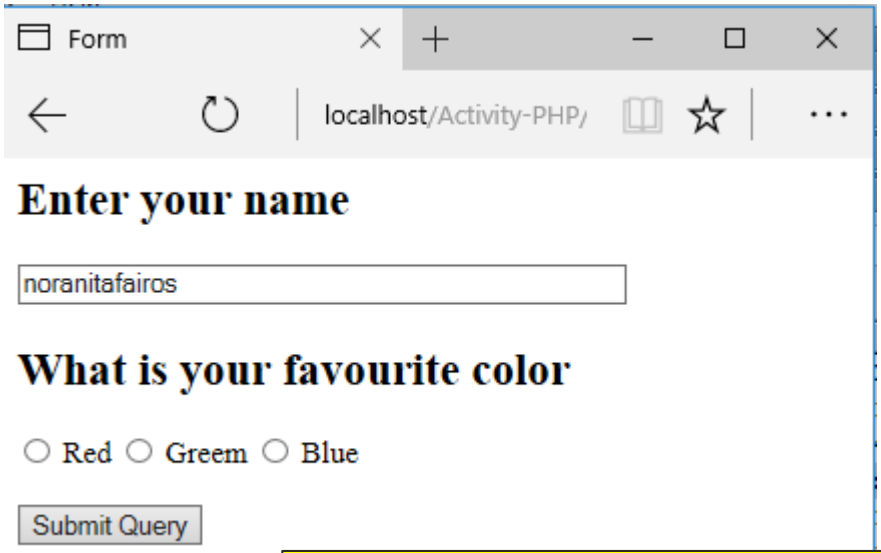
form_request.php

```php
<?php
    if( isset($_REQUEST["submit"])) {
        echo "Welcome ". $_REQUEST['name']. "<br />";
        echo "You are ". $_REQUEST['age']. " years old.";

        exit();
    }
?>
<html>
    <body>

        <form action = "<?php $_PHP_SELF ?>" method = "POST">
            Name: <input type = "text" name = "name" />
            Age: <input type = "text" name = "age" />
            <input type = "submit" name="submit" />
        </form>

    </body>
</html>
```

form02.html

User input and conditions

- In the next example, we will try to use **user input** to create conditions. First, we need a form look like this:

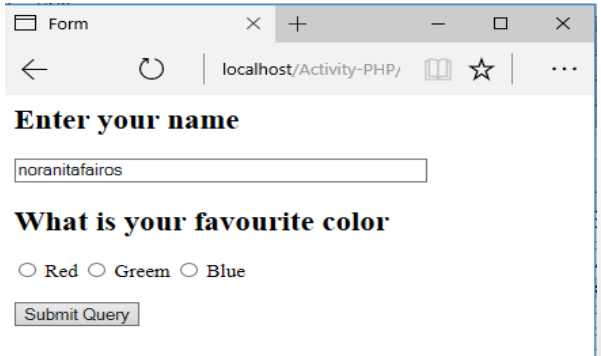



```
<input type="radio" name="color" value=”r">Red
<input type="radio" name="color" value=”g">Green
<input type="radio" name="color" value=”b">Blue
```

form02.html

User input and conditions

```
1  <html>
2  <head>
3  <title>Form</title>
4  </head>
5  <body>
6
7  <form method="post" action="handler02.php">
8
9  <h2>Enter your name</h2>
10 <input type="text" name="username" size="50">
11 <br>
12
13 <h2>What is your favourite color</h2>
14 <input type="radio" name="color" value="r"> Red
15 <input type="radio" name="color" value="g"> Greem
16 <input type="radio" name="color" value="b"> Blue
17 <br>
18 <br>
19 <input type="submit">
20 </form>
21 </body>
22 </html>
```



handler02.php

User input and conditions

- Now we will use these inputs to create a page that automatically **changes background color** according to what the **user's favorite color** is.
- We can do this by creating a condition that uses the data that the user has filled out in the form.

handler02.php

```
<html>
<head>
<title>Form</title>
</head>

<?php
$strHeading = "<h1>Hello " . $_POST["username"] . "</h1>";
$colortype = $_POST["color"];

    switch ($colortype) {
        case "r":
        $strBackgroundColor = "rgb(255,0,0)";
        break;

        case "g";
        $strBackgroundColor = "rgb(0,255,0)";
        break;

        case "b":
        $strBackgroundColor = "rgb(0,0,255)";
        break;

        default:
        $strBackgroundColor = "rgb(200,200,200)"; //gray color
        echo "<h3>Please select your color</h3>";
        break;
        }
?>

<body style="background: <?php echo $strBackgroundColor; ?>;">
<?php echo $strHeading; ?>

</body>
</html>
```

Enter your name

What is your favourite color

Red Greem Blue

Submit

```
<input type="text" name="username" size="50">
<br>

<h2>What is your favourite color</h2>
<input type="radio" name="color" value="r"> Red
<input type="radio" name="color" value="g"> Greem
<input type="radio" name="color" value="b"> Blue
<br>
<br>
<input type="submit">
```

User input and conditions

- The background color will be **white gray** if the user has **not chosen** any favorite color in the form.
- This is done by using **default** to specify what should happen if none of the above conditions are met.
- But what if the user **does not fill out his name**? Then it only says **"Hello Stranger!"** in the title.
- We will use an **extra** condition to change that.

handler02.php

User input and conditions

- Update handler02.php

- **Change this code:**

```
$strHeading = "<h1>Hello " . $_POST["username"] . "</h1>";
```

- **With this one!**

```
$strUsername = $_POST["username"];
if ($strUsername != "") {
$strHeading = "<h1>Hello " . $_POST["username"] . "</h1>";
} else {
$strHeading = "<h1>Hello stranger!</h1> ";
}
```

User input and conditions

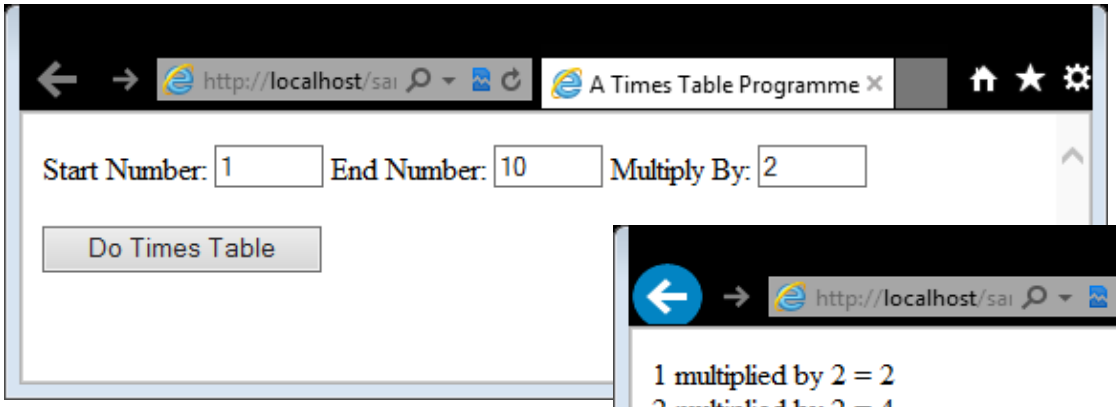
- In the example above, we use a condition to **validate** the information from the user.
- In this case, it might not be so important if the user did not write his name.
- But as you code more and more advanced stuff, it's vital that you take into account that the user may not always fill out forms the way you had imagined.

timesTableFull.php

HTML Forms and PHP

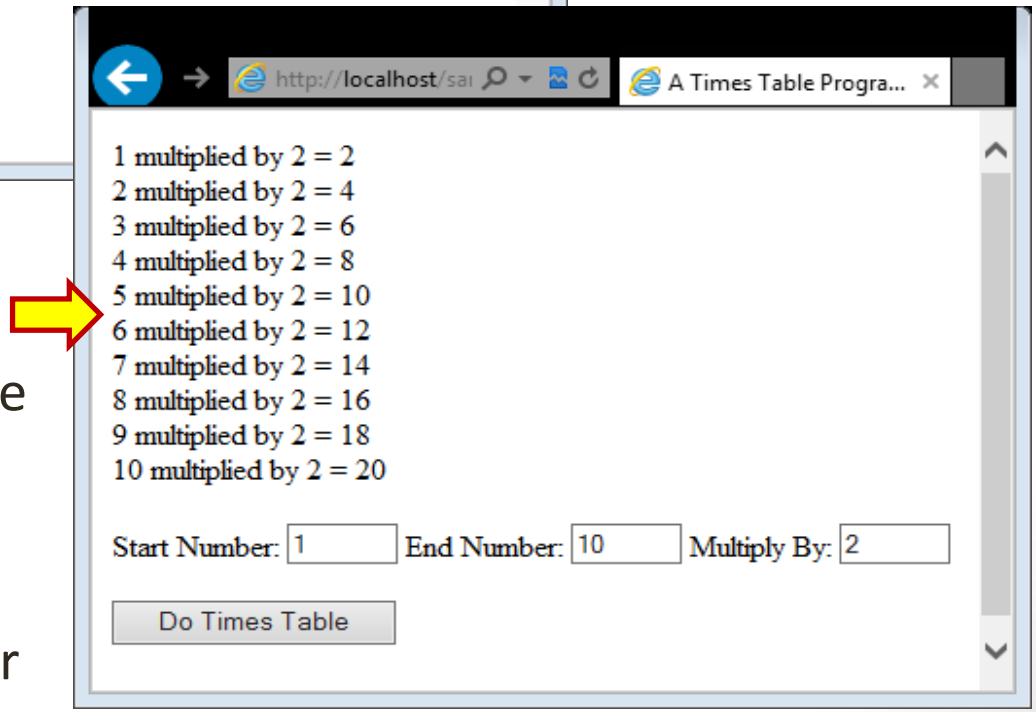
Example 1: Times Table Multiply

- Get the values from the textboxes and create a Times Table Multiply programmed.



timeTableFull.php

- When the button is clicked, the output will be something like this:



- Using **localhost** or **your own** web server

HTML Forms and PHP

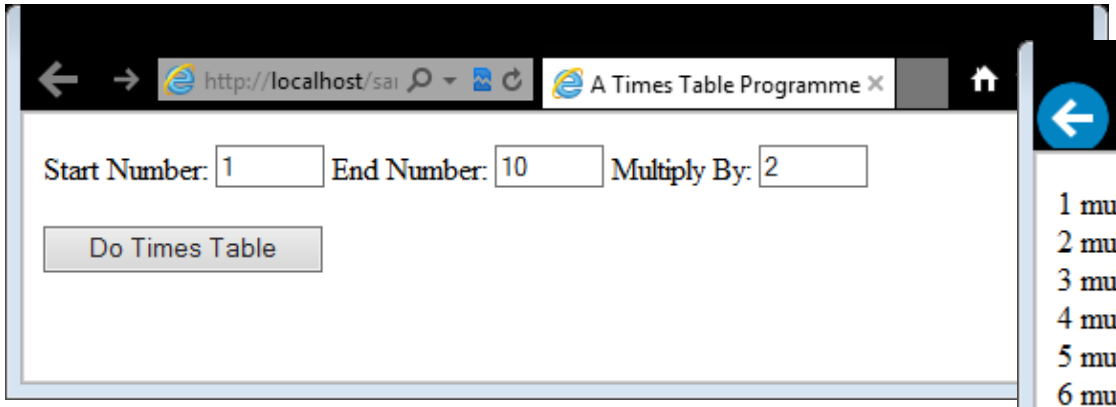
Example 1: Times Table Multiply

timesTable.html
timesTable.php

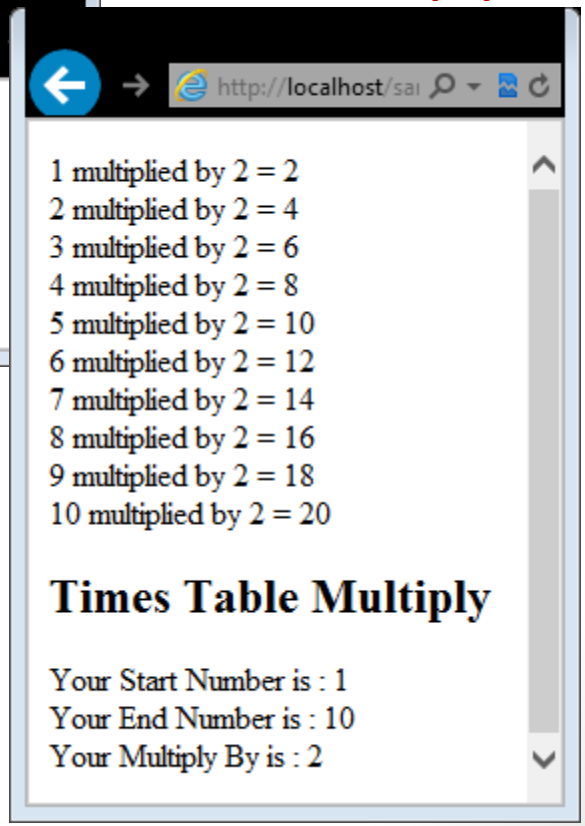
Do it..!
SUBMIT TODAY

- Get the values from the textboxes and create a Times Table Multiply programmed.

timetable.html



timetable.php



- When the button is clicked, the **timetable.php** process the data request in **timetable.html** and the output will be something like this:

- **Post to your webhosting & create it..!**