Information Technology Institute

Day01

Prepared by:
Noha Shehab

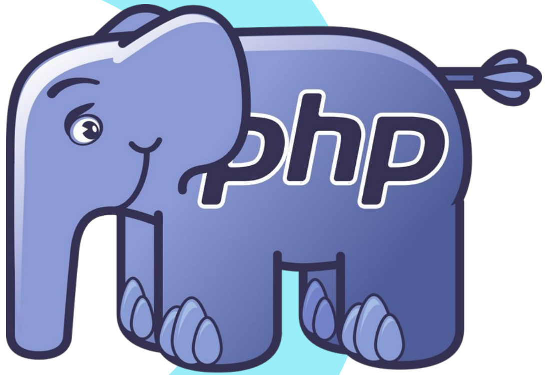
Updated and presented by:
Kareem Saeed

Teaching Assistant
Information Technology Institute (ITI)

Agenda

- History of PHP.
- Why PHP?
- What do we need? (LAMP Overview)
- Installing LAMP
- PHP Overview (Variables, Constants, Flow control, ...)

History of



History

1994

PHP originally stood for "personal home page". PHP development began by the Danish/Greenlandic programmer **Rasmus Lerdorf**

1997

- Zeev Suraski and Andi Gutmans , two Israeli developers at the Technion IIT, rewrote the parser and formed the base of PHP 3 , changed the name to PHP: Hypertext Preprocessor

1999

They started a new rewrite of PHP's core, producing the Zend Engine, They also founded Zend Technologies

- PHP 8.0 is a major update of the PHP language.
- It contains many new features and optimizations,, JIT, and improvements in the type system, error handling, and consistency.

WHY PHP?

- Ease of Learning PHP.
- Object-Oriented Support
- Availability of Support and Documentation
- 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 an open-source scripting language.
  www.php.net

Some facts about PHP

- PHP page is a file with a .php extension can contain a combination of HTML Tags and PHP scripts.

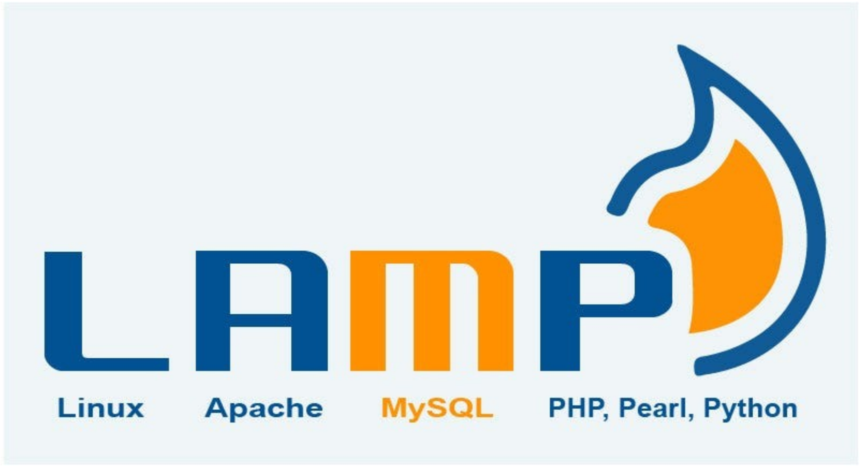
- PHP recursive acronym for PHP(Hypertext Preprocessor): Hyper
HTML to the web browser. A client cannot see the PHP source c

Some facts about PHP

- PHP is Server-side scripting language: Server-side scripting means that the PHP code is processed on the web server rather than the client machine.
- PHP supports many databases (MySQL and PHP combination is widely used).

WHAT DO WE NEED?

- Linux (Operating system).
- Installing Apache and Updating the Firewall.
- Installing MySQL.
- Installing PHP.



WHAT DO WE NEED?

- XAMPP is the most popular PHP development environment.

- XAMPP is a completely free, easy to install Apache distribution containing MariaDB, PHP, and Perl. The XAMPP open source package has been set up to be incredibly easy to install and to use.

- Lamp --> linux apachi mysql php
- Wamp --> windows apachi mysql php
- Mamp --> mac apachi mysql php

WHAT DO WE NEED?

- LAMP is an acronym for a solution stack of free, open-source software.
- Originally coined from the first letters of

  #**L**inux (operating system),

  #**A**pache HTTP Server,

  #**M**ySQL (database software)

  #**P**rogramming language like Perl/PHP/Python

principal components to build available general purpose web server.

Installation

- LAMP
  https://bitnami.com/stack/lamp/installer
- XAMPP
  https://www.apachefriends.org/index.html
- PHP –version (7.4)
- Test the running of server by http://localhost/

What is 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

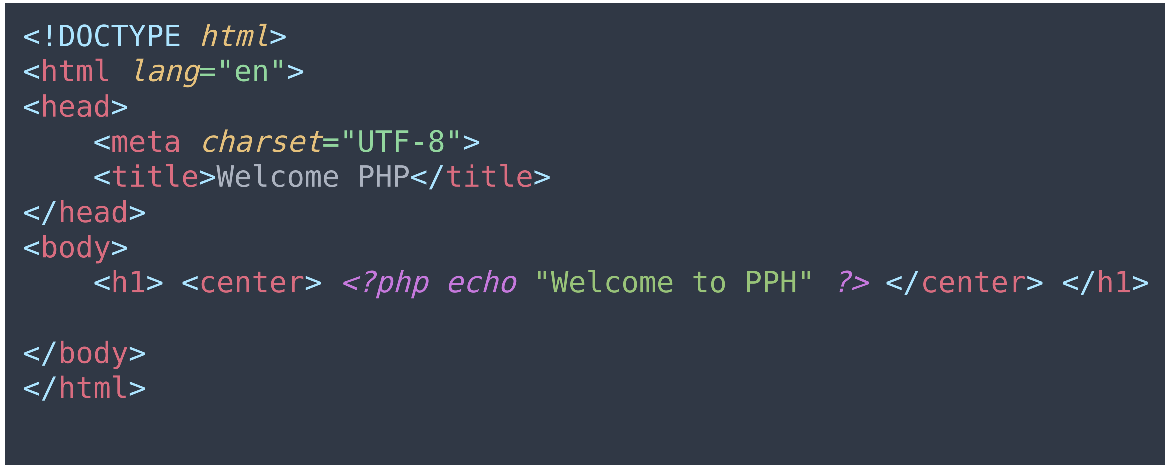
- PHP files have a default file extension of ".php"

What PHP can 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.

EMBEDDING PHP IN HTML

- Simply you can PHP in HTML page by Adding the php tag as the following



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Welcome PHP</title>
</head>
<body>
    <h1> <center> <?php echo "Welcome to PPH" ?> </center> </h1>

</body>
</html>
```

- The PHP interpreter will run through the script and replace it with the output from the script

PHP IS A SERVER SIDE

- The PHP has been interpreted and executed on the web server, as distinct from JavaScript and other client-side technologies interpreted and executed within a web browser on a user's machine

- The code that you now have in this file consists of five types of text
  #HTML
  #PHP tags
  #PHP scripts
  #White spaces
  #Comments

PHP TAGS

- XML style

  ```
  <?php echo ‘<p>Hello!.</p>’; ?>
  ```

- Short style

  ```
  <? echo ‘<p>Hello!</p>’; ?>
  ```

- SCRIPT style

  ```
  <script language=’php’> echo
       ‘<p>Hello!.</p>’; </script>
  ```

- ASP style

  ```
  <% echo ‘<p>Hello!.</p>’; %>
  ```

PHP TAGS

- XML style

  Is recommended because it can’t be closed off by the administrator beside it’s portable through systems.

- Short style

  Is the simplest and follows the style of a Standard Generaliz Markup Language (SGML) processing instruction.

  To use this type you need to enable the short_open_tag setting in your config file.

- SCRIPT style

  This tag style is the longest and will be familiar if you’ve used JavaScript or VBScript.

PHP TAGS

- ASP.Net style

  Is the same as used in Active Server Pages (ASP) or ASP.NET.

  You can use it if you have enabled the asp_tags configuration setting in php.ini.

PHP STATEMENTS & WHITESPACES

- **Echo** : reserved word to display content in browser,

  echo ‘<p>Hello, World!.</p>’;

- each line ends with (;)
- Spacing characters such as newlines (carriage returns), spaces, and tabs are known as whitespace.
- Browsers ignore whitespace in HTML. So does the PHP engine.

- echo ‘Hello ‘;
- echo ‘World’;
- echo **‘hello ‘;echo ‘world’;** are the same, but the first version is easier to read

COMMENTS

- C-style, multiline comment that might appear at the start of a PHP script:

```
/* PHP Day01, Insturctor:Noha Shehab, Wish a fruitful Journey with PHP ! ^^ */
```

- You can also use single-line comments, either in the C++ like

```
// This is a comment
```

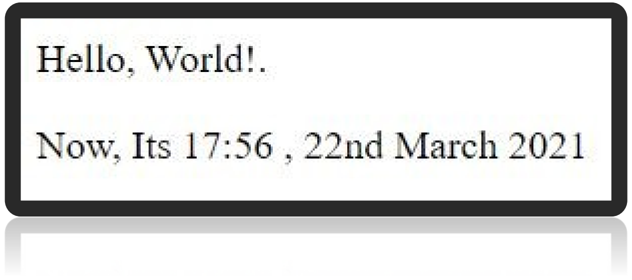
- Or like bash script

```
#this is another comment
```

Go Dynamic …

- Date function will print the current date and time as following

```php
echo "<p>Now, Its ";
echo date('H:i , jS F Y');
echo "</p>";
```



```
Hello, World!.

Now, Its 17:56 , 22nd March 2021
```

Form Variables

- You may be able to access the contents of the field in the following ways:

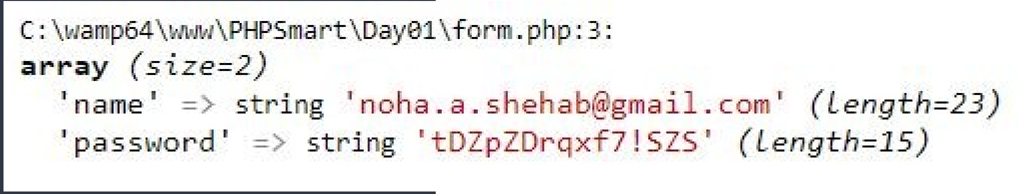
$field_name: Short style ($ field_name) is convenient but requires the register_globals configuration setting be turned on.

Medium style involves retrieving form variables from one of the arrays $_POST,$_GET or $_REQUEST

ACCESSING FORM VARIABLES

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <form action="form.php" method="GET" >
        <input type="text" name="name">
        <input type="password" name="password">
        <input type="submit" >
    </form>
</body>
</html>
```

```php
var_dump($_GET);
```



```
C:\wamp64\www\PHPSmart\Day01\form.php:3:
array (size=2)
  'name' => string 'noha.a.shehab@gmail.com' (length=23)
  'password' => string 'tDZpZDrqxf7!SZS' (length=15)
```

VARIABLES AND LITERALS

- Value itself is a literal.

- There are two kinds of strings:
  #Double quotation
  #Single quotation.

- PHP tries to evaluate strings in double quotation marks, resulting in the behavior shown earlier.

- Single-quoted strings are treated as true literals.

VARIABLES AND LITERALS

- There is also a third way of specifying strings using the heredoc syntax.

```
echo <<<theEnd
Line 1
Line 2
Line 3
theEnd;
```

UNDERSTANDING IDENTIFIERS

- Identifiers are the names of variables .

- You need to be aware of the simple rules defining valid identifiers:

  #Identifiers can be of any length and can consist of letters, numbers, and under-scores.

  #Identifiers cannot begin with a digit.

  #In PHP, identifiers are case sensitive.

A variable can have the same name as a function. This usage is confusing.

PHP Variables

- Variable can have short names (like x and y) or more descriptive names (age, carname, totalvolume).

- **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
  #A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
  #A variable name should not contain spaces
  #Variable names are case sensitive ($y and $Y are two different variables)

Declare your first variable

- PHP has no command for declaring a variable.

- A variable is created the moment you first assign a value to it:

```
$txt="Hello world!";
$x=5;
```

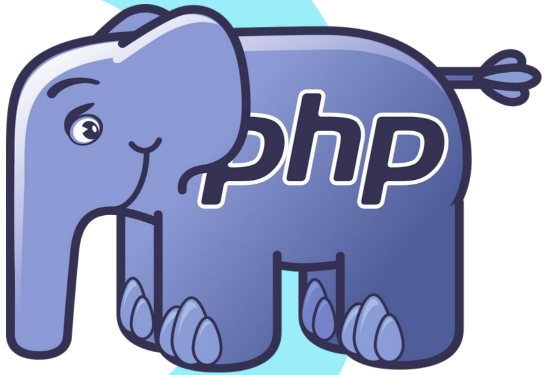
- After the execution of the statements above, the variable txt will hold the value Hello world!, and the variable x will hold the value 5.

- Note: When you assign a text value to a variable, put quotes around the value.

PHP is a Loosely Typed Language

- In the example above, notice that 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.
- In a strongly typed programming language, we will have to declare (define) the type and name of the variable before using it.

Variable Scope



Variable Scope

- 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
  Parameter
  Constants
  SuperGlobal

Local Scope

- A variable declared within a PHP function is local and can only be accessed within that function:

```
$x=5; // global scope
function myTest(){
    $y=5
    echo $y; // local scope
}
myTest();
```

Global Scope

- A variable that is 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.

Global Scope

```
$x=5; // global scope
function myTest(){
    $y=5;
    echo $y; // local scope
     global $x;
     $x= 15;
    var_dump($x);  // 15
}
myTest();

var_dump($x);  //15
```

Static Scope

- When a function is completed, all of its variables are normally deleted. However, sometimes you want a local variable to not be deleted.

- To do this, use the static keyword when you first declare the variable:

```
function testStaticFunction(){
    static $m;
    $m ++;
    var_dump($m);
}
testStaticFunction(); #1
testStaticFunction(); #2
testStaticFunction(); #3
```

Constants

- You can define these constants using the define function, or Const keyword.
- One important difference between constants and variables is
- that when you refer to a constant, it does not have a dollar sign
- in front of it. If you want to use the value of a constant, use its
- name only.

```
define("CONSTANT","Hello world from PHP");
const TEST="Welcome";
echo CONSTANT; // Hello world from PHP
echo TEST; // Welcome
```

Parameter scope

- 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:

```php
function parameterScope($var){
    echo $var;
}
parameterScope(5); //5
$name= "Noha";
parameterScope($name);// Noha
```

Super global

Super global or auto global and can be seen everywhere, both inside and outside functions.
Check the following:

$_GET: An array of variables passed to the script via the GET method.
$_POST: An array of variables passed to the script via the POST method.
$_REQUEST: An array of all user input including the contents of input including $_GET, $_POST & $_COOKIE
$_COOKIE: An array of cookie variables
$_FILES: An array of variables related to file uploads
$_SESSION: An array of session variables

Variables scoping summary

The six basic scope rules in PHP are as follows:

- Global variables declared in a script are visible throughout that script, but not inside functions.

- Global Variables inside functions refer to the global variables of the same name.

- Static variables created inside functions are invisible from outside the function but keep their value between one execution of the function and the next.

- Variables created inside functions are local to the function and cease to exist when the function terminates.

Variables scoping summary

- Built-in super global variables are visible everywhere within a script.

- Constants, once declared, are always visible globally; that is, they can be used inside and outside functions.

Echo & Print

In PHP there is two basic ways to get output: echo and print.
There are some differences between echo and print:

- echo - can output one or more strings
- print - can only output one string, and returns always 1

```
echo "<h2>PHP is fun!</h2>";
echo "Hello world!<br>";
echo "This", " string", " was", " made", " with multiple
parameters.";
$var="PHP Day01";
print ($var) ; #"PHP Day01“
```

Tip: echo is marginally faster compared to print as echo does not return any value.

Variables’ datatypes

Variables’ data types

- A variable’s type refers to the kind of data stored in it
- PHP supports the following basic data types:
  - Integer—Used for whole numbers
  - Float (also called double)—Used for real numbers
  - String—Used for strings of characters
  - Boolean—Used for true or false values
  - Array—Used to store multiple data items
  - Object—Used for storing instances of classes
  - NULL
  - Resource--Reference to external source of data

Variable casting

- You can pretend that a variable or value is of a different type by using a type cast.
- You simply put the temporary type in parentheses in front of the variable you want to cast.
- For example, you could have declared the two variables from the preceding section using a cast:

```
$var1 = 0;
$var2 = (float)$var1;
```

Variable of variable

- PHP provides one other type of variable: **the variable of variable.**
- Variable variables enable you to change the name of a variable dynamically.
- For example, you could set
  `$varname= ‘age’;`
- You can replace $$varname with $age. For example, you can set the value of $age as follows:
  `$$varname= 5;`
- This is exactly equivalent to
  `$age= 5;`

Arithmetic operators

- Arithmetic operators are straightforward; they are just the normal mathematical operators.

- With each of these operators, you can store the result of the operation, as in this example:

```
$result = $a + $b;
```

Arithmetic operators

Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power

Operators

- You can use the string concatenation operator to add two strings and to generate and store a result much as you would use the addition operator to add two numbers:

```
$a = "Hello, ";
$b = "World!";
$result = $a.$b; // Hello, World
```

- The $result variable now contains the string “Hello, World!”

Comparison operators

Expression	Meaning	Example	Illustrate
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7

Combined operators

Combined assignment operators exist for each of the arithmetic operators and for the string concatenation operator.

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	Concatination

Pre/Post-increment.

- The pre-and post-increment (++) and decrement (--) operators are similar to the +=and -= operators, but with a couple of twists.
- Example:

```
$a=4;
echo ++$a;//echo 5 , value of $a = 5
$a=4;

echo $a++;//echo 4 , value of $a = 5
echo --$a;//
```

Reference operator

- The reference operator (& an ampersand) can be used in conjunction with assignment.

```
$a = 5;
$b = $a;
```

- These code lines make a second copy of the value in $a and store it in $b. If you subsequently change the value of $a, $b will not change:

```
$a = 7; // $b will still be 5
```

- You can avoid making a copy by using the reference operator. For example,

```
$a = 5;
$b = &$a;
$a = 7; // $a and $b are now both 7
```

Reference tip.

References can be a bit tricky.

Remember that a reference is like an alias rather than like a pointer.

Both $a and $b point to the same piece of memory. You can change this by unsetting one of them.

Logical operators

The logical operators combine the results of logical conditions. $a, is between 0 and 100. using the AND operator, as follows: **$a >= 0 && $a <=100**

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

Error suppression operator @

- The error suppression operator (@) can be used in front of any expression that is, any thing that generates or has a value.

```
$a = @(25/0);
var_dump($a);  // INF

$b= 44/0;
var_dump($b);
//  Warning: Division by zero in on line 77
```

- Without the @ operator, this line generates a divide by zero warning. With the operator included, the error is suppressed

The execution operator ``

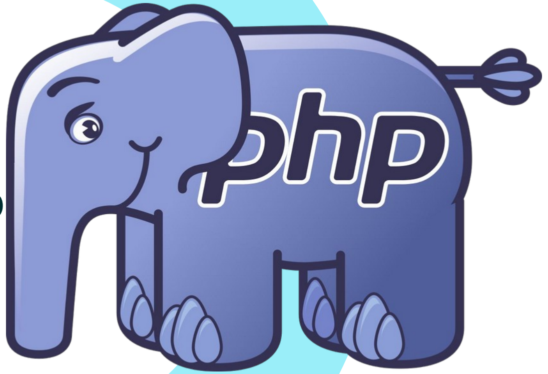
- The execution operator is really a pair of operators a pair of backticks (``) in fact.

- The backtick is not a single quotation mark;

- It is usually located on the same key as the ~ (tilde) symbol on your keyboard.

```
$out = `ls -la`;
echo $out;
```

Variables’ functions



Variable functions

- **gettype($var)**

  It determines the type and returns a string containing the **type name**: bool, int, double (for floats), string, array, object, resource, or   NULL.

  It returns **unknown type** if it is not one of thestandard types.

- **Settype($var,”datatype”)**

  you pass it a variable for which you want to change the type and a string containing the new type for that variable from the previous list.

```
$num="10";
settype($num,"int");
echo gettype($num); // Integer
```

Common variables functions

- is_array(): Checks whether the variable is an array.
- is_double(), is_float(), is_real() (All the same function): Checks whether the variable is a float.
- is_long(), is_int(), is_integer() (All the same function): Checks whether the variable is an integer.
- is_string(): Checks whether the variable is a string.
- is_bool(): Checks whether the variable is a boolean.

Common variables functions

- is_object(): Checks whether the variable is an object.
- is_resource(): Checks whether the variable is a resource.
- is_null(): Checks whether the variable is null.
- is_scalar(): Checks whether the variable is a scalar, that is, an integer, boolean, string, or float.
- is_numeric(): Checks whether the variable is any kind of number or a numericstring.

Common variables functions

- **isset():** function takes a variable name as an argument and returns true if it exists and false otherwise.

- You can also pass in a comma-separated list of variables, and isset() will return true if all the variables are set.

- You can wipe a variable out of existence by using its companion function, **unset().**

Common variables functions

empty(): function checks to see whether a variable exists and has a nonempty, nonzero value; it returns true or false accordingly.

is_callable(): Checks whether the variable is the name of a valid function.

Flow control & Looping

Flow Control

- If condition
- Switch case
- for
- Foreach
- Break, continue, exit.
- While
- Do while

If condition

```
if (condition) {
  code to be executed if this condition is true;
} elseif(condition) {
  code to be executed if first condition is false and this
condition is true;
} else {
  code to be executed if all conditions are false;
}
```

Switch case

```
switch (n) {
  case label1:
    code to be executed if n=label1;
    break;
  case label2:
    code to be executed if n=label2;
     break;
  case label3:
    code to be executed if n=label3;
   break;
    ...
  default:
    code to be executed if n is different from all labels;
}
```

For, Foreach

```
for( expression1; condition; expression2){
expression3;
}


foreach ($array as $value) {
  echo $value;
}
```

While, do-while

- The while loop executes a block of code as long as the specified condition is true.

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

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

```
$x= 1000;
do{
    print("welcome to do while looping");
}while($x<10);
```

Break, continue, Exit

- If you want to jump to the next loop iteration, you can instead use the continue statement.

```
for($i=0;$i<10; $i++){
    echo "We need the break!"
    if($i==4) break;
}
```

```
for($i=0;$i<10; $i++){
    echo "We need the break! ";
    if($i==4) continue;
}
```

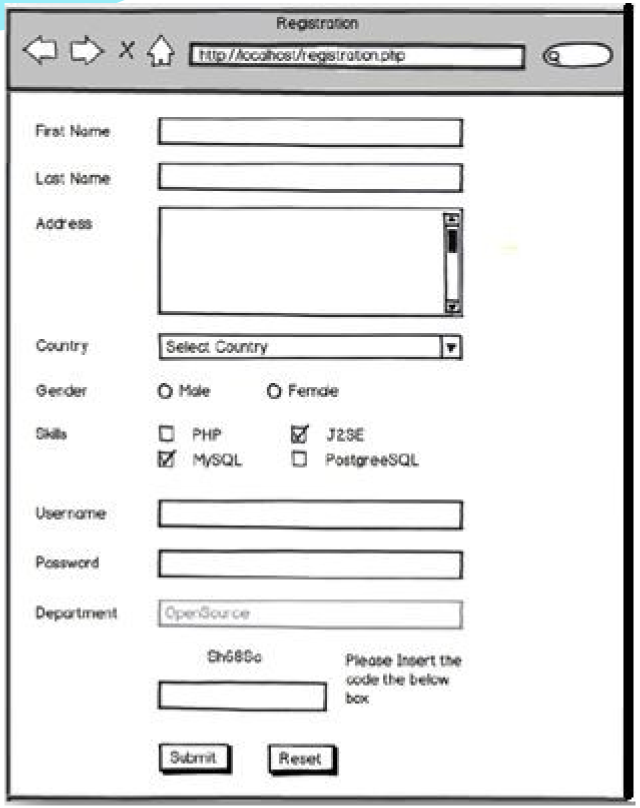
- Exit:

  #exit; or exit();

  #If you want to finish executing the entire PHP script, you can use exit. This approach is typically useful when youare performing error checking

Lab 01

Construct this form in html,
send the data to the PHP Server



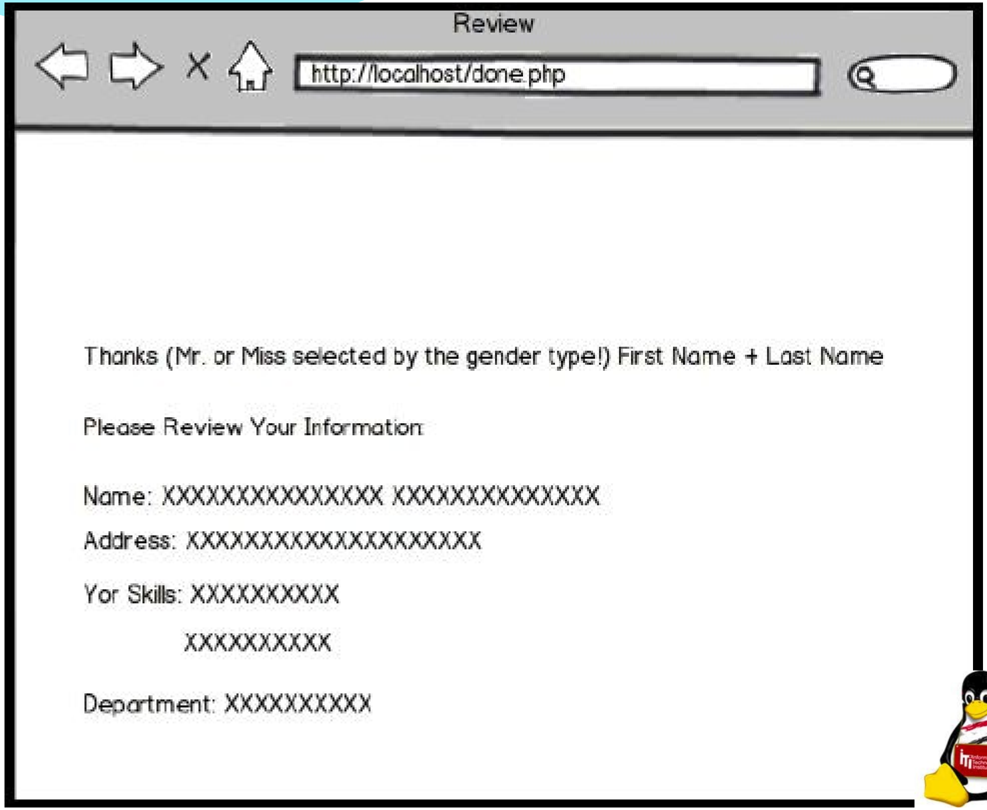
Registration

http://localhost/registration.php

First Name	
Last Name	
Address	
Country	Select Country ▾
Gender	○ Male ○ Female
Skills	☐ PHP ☑ J2SE
	☑ MySQL ☐ PostgreeSQL
Username	
Password	
Department	OpenSource
	Sh68Sa
	Please Insert the code the below box
	Submit Reset

Lab 01

Construct and send a mail with the provided data



Thanks (Mr. or Miss selected by the gender type!) First Name + Last Name

Please Review Your Information:

Name: XXXXXXXXXXXXXXX XXXXXXXXXXXXXX

Address: XXXXXXXXXXXXXXXXXXXX

Yor Skills: XXXXXXXXXX

XXXXXXXXXX

Department: XXXXXXXXXX

Thanks ^^

Kareem Saeed
kareemmorsy30@gmail.com