Class :- SYBBA(CA)(III Semester) (2019 Pattern)

Sub:-PHP

Prof. Yogesh. P. Deshmukh

Unit 1

PHP Basics

- **PHP is a server scripting language, and 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.(Active server page)...**
- **PHP -7 is the latest version of PHP.**
- **Php is a object oriented programming language , it is used to create the object…**
- **It means any items, which have property and method..**

- The default file extension for PHP files is ".php".
- html =.html,css=.css java script=.js or html ,,,,php=.php
- A PHP file normally contains HTML tags, and some PHP scripting code.
- built-in PHP function "echo" to output the text "Hello World!" on a web page:
- Echo is used to show the out put …or give message..same as printf in c

- **What is PHP?**
- PHP is an acronym for "PHP: **Hypertext Preprocessor"**
- PHP is a widely-used, open source scripting language.
- PHP scripts are executed on the server for ex wamp...
- PHP is free to download and use.
- **What is a PHP File?**
- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code is executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"

- **PHP Data Types**
- Variables can store data of different types, and different data types can do different things.
- PHP supports the following data types:
- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL

PHP Array

An array stores multiple values in one single variable.

- In the following example $cars is an array. The PHP var_dump() function returns the data type and value:
- Example
- <?php
  $cars = array("Volvo","BMW","Toyota");
  var_dump($cars);
  ?>
- <!DOCTYPE html>
- <html>
- <body>
-
- <?php
- $cars = array("Volvo","BMW","Toyota");
- var_dump($cars);
- ?>
- </body>
- </html>

PHP Operators

- PHP divides the operators in the following groups:

- **Arithmetic operators**
- **Assignment operators**
- **Comparison operators**
- **Increment/Decrement operators**
- **Logical operators**
- **String operators**
- **Array operators**
- **Conditional assignment operators**

PHP Operators

- PHP divides the operators in the following groups:

- **Arithmetic operators**
- **Assignment operators**
- **Comparison operators**
- **Increment/Decrement operators**
- **Logical operators**
- **String operators**
- **Array operators**
- **Conditional assignment operators**

Unit 2
Control Structures and Loops

PHP Conditional Statements

A. if statement - executes some code if one condition is true
B. if...else statement - executes some code if a condition is true and another code if that condition is false
C. if...elseif...else statement - executes different codes for more than two conditions
D. switch statement - selects one of many blocks of code to be executed

PHP Loops

In PHP, we have the following loop types:

- while - loops through a block of code as long as the specified condition is true
- do...while - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- for - loops through a block of code a specified number of times
- foreach - loops through a block of code for each element in an array

- **The PHP foreach Loop**

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

Syntax

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

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

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

<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
  echo "$value <br>";
}
?>

</body>
</html>
```

Unit 3
Functions, Objects and Errors

PHP Built-in Functions

strrev(): reverse the string..
strlen() function returns the length of a string.
strpos() function searches for a specific text within a string. If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.
str_replace() function replaces some characters with some other characters in a string.
pi() function returns the value of PI:
The min() and max() functions can be used to find the lowest or highest value in a list of arguments:
abs() function returns the absolute (positive) value of a number:
sqrt() function returns the square root of a number:
round() function rounds a floating-point number to its nearest integer:
rand() function generates a random number

- Create a PHP Constant

To create a constant, use the define() function.
Syntax
define(*name*, *value*, *case-insensitive*)

```
<!DOCTYPE html>
<html>
<body>
<?php
// case-sensitive constant name
define("GREETING", "Welcome to W3Schools.com!");
echo GREETING;
?>

</body>
</html>
```

- PHP Constant Arrays

In PHP7, you can create an Array constant using the define() function.

```
<!DOCTYPE html>
<html>
<body>
<?php
define("cars", [
  "Alfa Romeo",
  "BMW",
  "Toyota"
]);
echo cars[0];
?>
</body>
</html>
```

- **PHP User Defined Functions**

```
<head>
<title>
Another php example
</title>
</head>
<body>
<h1> Login Page </h1>
<form method="post" action="greate2.php">
Enter the mark  of one sub:-
   <input type="text" name="n1" />
Enter the marks  of 2sub:-
            <input type="text" name="n2" />
   <input type="submit" value="click me"
onclick="greate2()" />
</form>
</body>
</html>
```

- **PHP User Defined Functions**

```
<html><head><title>Another php example</title></head><body>

<?php
    $x = $_POST["n1"];
  $y= $_POST["n2"];
            function   greate2($x,$y)
    {
      if($x>$y)
        {
          echo " gretest no is".$x;
          }
           else
                {
               echo "gretest no is".$y;
                }
 }
?>

<?php
        greate2($x,$y);
?>
</body>
</html>
```

Unit 4
working with Forms

- **PHP Form Handling**

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

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

</body>
</html>
```

- **PHP Form Handling**

When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method.
To display the submitted data you could simply echo all the variables. The "welcome.php" looks like this:

```
<html>
<body>
Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>
</body>
</html>
```

• GET vs. POST

Both GET and POST create an array (e.g. array(key1 => value1, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.

Both GET and POST are treated as $_GET and $_POST. These are superglobals, which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

$_GET is an array of variables passed to the current script via the URL parameters.

$_POST is an array of variables passed to the current script via the HTTP POST method.

- When to use GET?

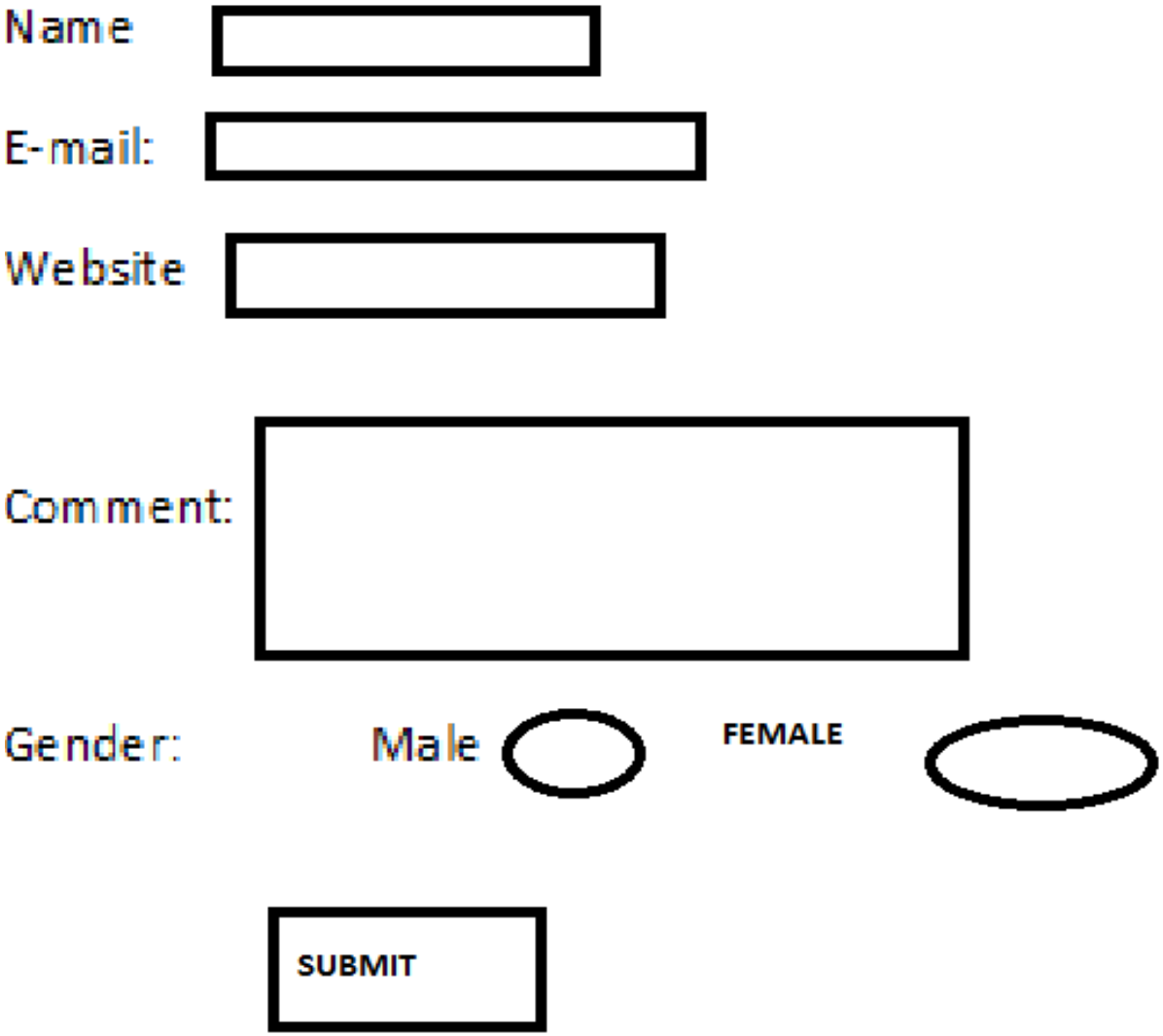
Information sent from a form with the GET method is **visible to everyone** (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.
GET may be used for sending non-sensitive data.
Note: GET should NEVER be used for sending passwords or other sensitive information!

- When to use POST?

Information sent from a form with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and has **no limits** on the amount of information to send.
Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.
However, because the variables are not displayed in the URL, it is not possible to bookmark the page.
Developers prefer POST for sending form data.

Unit 5
More with Forms

- **PHP Form Validation**



- **Text Fields**

The name, email, and website fields are text input elements, and the comment field is a textarea. The HTML code looks like this:
Name: <input type="text" name="name">
E-mail: <input type="text" name="email">
Website: <input type="text" name="website">
Comment: <textarea name="comment" rows="5" cols="40"></text area>

Radio Buttons

The gender fields are radio buttons and the HTML code looks like this:
Gender:
<input type="radio" name="gender" value="female">Female
<input type="radio" name="gender" value="male">Male
<input type="radio" name="gender" value="other">Other

PHP - Validate Name

The code below shows a simple way to check if the name field only contains letters, dashes, apostrophes and whitespaces. If the value of the name field is not valid, then store an error message:

```
$name = test_input($_POST["name"]);
if (!preg_match("/^[a-zA-Z-' ]*$/",$name)) {
  $nameErr = "Only letters and white space allowed";
}
```

comment field is a textarea. The HTML code looks like this:

```
Name: <input type="text" name="name">
E-mail: <input type="text" name="email">
Website: <input type="text" name="website">
Comment: <textarea name="comment" rows="5" cols="40"></text
area>
```

- **PHP - Validate E-mail**

The easiest and safest way to check whether an email address is well-formed is to use PHP's filter_var() function.
In the code below, if the e-mail address is not well-formed, then store an error message:

```
$email = test_input($_POST["email"]);
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
  $emailErr = "Invalid email format";
}
```

• PHP - Validate URL

The code below shows a way to check if a URL address syntax is valid (this regular expression also allows dashes in the URL). If the URL address syntax is not valid, then store an error message:

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

Unit 6
Storing and Protecting Data

What is a Cookie?

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP

A cookie is created with the setcookie() function.

Syntax

setcookie(*name, value, expire, path, domain, secure, httponly*);

Only the *name* parameter is required. All other parameters are optional.

What is a PHP Session?

When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.

So; Session variables hold information about one single user, and are available to all pages in one application.

• Start a PHP Session

A session is started with the session_start() function.
Session variables are set with the PHP global variable: $_SESSION.
Now, let's create a new page called "demo_session1.php". In this page, we start a new PHP session and set some session variables:<?php

• Start a PHP Session

```
 // Start the session
session_start();
?><!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```

Get PHP Session Variable Values

Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php").
Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session_start()).
Also notice that all session variable values are stored in the global $_SESSION variable:

Get PHP Session Variable Values

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>

</body>
</html>
```

Modify a PHP Session Variable

To change a session variable, just overwrite it:

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// to change a session variable, just overwrite it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>

</body>
</html>
```

Destroy a PHP Session

To remove all global session variables and destroy the session, use session_unset() and session_destroy():

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// remove all session variables
session_unset();
// destroy the session
session_destroy();

echo "All session variables are now removed, and the session is destroyed."
?>
</body>
</html>
```

Unit 7
MySQL Database

What is MySQL?

- ✓MySQL is a database system used on the web
- ✓MySQL is a database system that runs on a server
- ✓MySQL is ideal for both small and large applications
- ✓MySQL is very fast, reliable, and easy to use
- ✓MySQL uses standard SQL
- ✓MySQL compiles on a number of platforms
- ✓MySQL is free to download and use
- ✓MySQL is developed, distributed, and supported by Oracle Corporation

- **Create a MySQL Database Using MySQLi and PDO**

The CREATE DATABASE statement is used to create a database in MySQL.
The following examples create a database named "myDB":

Example (MySQLi Object-oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username,
$password);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
```

```
// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
  echo "Database created successfully";
} else {
  echo "Error creating database: " . $conn->error;
}

$conn->close();
?>
```