Introduction to PHP

PHP is an acronym for "PHP: Hypertext Preprocessor"

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.

First Practical:

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

<?php
echo "My first PHP script!";
?>

</body>
</html>
```

Variable declaration

A variable is declared using a **$ sign** followed by the variable name.

Syntax: $variablename=value;

Rules for declaring PHP variable:

- A variable must start with a dollar ($) sign, followed by the variable name.
- It can only contain alpha-numeric character and underscore (A-z, 0-9, _).
- A variable name must start with a letter or underscore (_) character.
- A PHP variable name cannot contain spaces.
- One thing to be kept in mind that the variable name cannot start with a number or special symbols.
- PHP variables are case-sensitive, so $name and $NAME both are treated as different variable.

Example:

```
<?php

    $str="hello string";

    $x=200;

    $y=44.6;

    echo "string is: $str <br/>";

    echo "integer is: $x <br/>";

    echo "float is: $y <br/>";

    ?>
```

Example 2:

```
    <?php
    $x=5;
    $y=6;
    $z=$x+$y;
    echo $z;
    ?>
```

Control statement

PHP If-else Statement

PHP if-else statement is executed whether condition is true or false.

If-else statement is slightly different from if statement. It executes one block of code if the specified condition is **true** and another block of code if the condition is **false**.

Syntax:

```
if(condition){

//code to be executed if true

}else{
```

```
//code to be executed if false

}
```

Example

```
<?php
$num=12;
if($num%2==0){
echo "$num is even number";
}else{
echo "$num is odd number";
}
?>
```

PHP For Loop

PHP for loop can be used to traverse set of code for the specified number of times.

It should be used if the number of iterations is known otherwise use while loop. This means for loop is used when you already know how many times you want to execute a block of code.

It allows users to put all the loop related statements in one place. See in the syntax given below:

Syntax

```
for(initialization; condition; increment/decrement){

//code to be executed

}
```

Example

```
<?php
```

```
for($n=1;$n<=10;$n++){

echo "$n<br/>";

}

?>
```

PHP Arrays

PHP array is an ordered map (contains value on the basis of key). It is used to hold multiple values of similar type in a single variable.

Advantage of PHP Array

Less Code: We don't need to define multiple variables.

Easy to traverse: By the help of single loop, we can traverse all the elements of an array.

Sorting: We can sort the elements of array.

PHP Array Types

There are 3 types of array in PHP.

1. Indexed Array
2. Associative Array

PHP Indexed Array

PHP index is represented by number which starts from 0. We can store number, string and object in the PHP array. All PHP array elements are assigned to an index number by default.

There are two ways to define indexed array:

1st way:

```
$season=array("summer","winter","spring","autumn");
```

2nd way:

```
$season[0]="summer";

$season[1]="winter";

$season[2]="spring";

$season[3]="autumn";
```

Example

```
<?php

$season=array("summer","winter","spring","autumn");

echo "Season are: $season[0], $season[1], $season[2] and $season[3]";

?>
```

PHP Associative Array

We can associate name with each array elements in PHP using => symbol.

There are two ways to define associative array:

1st way:

```
$salary=array("Sonoo"=>"350000","John"=>"450000","Kartik"=>"200000");
```

2nd way:

```
$salary["Sonoo"]="350000";

$salary["John"]="450000";
```

```
$salary["Kartik"]="200000";
```

Example

```
<?php

$salary=array("Sonoo"=>"350000","John"=>"450000","Kartik"=>"200000");

echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";

echo "John salary: ".$salary["John"]."<br/>";

echo "Kartik salary: ".$salary["Kartik"]."<br/>";

?>
```

PHP Form Handling

We can create and use forms in PHP. To get form data, we need to use PHP superglobals $_GET and $_POST.

The form request may be get or post. To retrieve data from get request, we need to use $_GET, for post request $_POST.

PHP Get Form

Get request is the default form request. The data passed through get request is visible on the URL browser so it is not secured. You can send limited amount of data through get request.

Let's see a simple example to receive data from get request in PHP.

File: form1.html

```
<form action="welcome.php" method="get">

Name: <input type="text" name="name"/>
```

```
<input type="submit" value="visit"/>

</form>
```

File: welcome.php

```
<?php

$name=$_GET["name"];//receiving name field value in $name variable

echo "Welcome, $name";

?>
```

PHP Post Form

Post request is widely used to submit form that have large amount of data such as file upload, image upload, login form, registration form etc.

The data passed through post request is not visible on the URL browser so it is secured. You can send large amount of data through post request.

Let's see a simple example to receive data from post request in PHP.

File: form1.html

```
<form action="login.php" method="post">

<table>

<tr><td>Name:</td><td> <input type="text" name="name"/></td></tr>

<tr><td>Password:</td><td> <input type="password" name="password"/></td></tr>

<tr><td colspan="2"><input type="submit" value="login"/>  </td></tr>

</table>

</form>
```

File: login.php

```
<?php

$name=$_POST["name"];//receiving name field value in $name variable

$password=$_POST["password"];//receiving password field value in $password variable



echo "Welcome: $name, your password is: $password";

?>
```

Form validation

The first thing we will do is to pass all variables through PHP's htmlspecialchars() function.

When we use the htmlspecialchars() function; then if a user tries to submit the following in a text field:

<script>location.href('http://www.hacked.com')</script>

- this would not be executed, because it would be saved as HTML escaped code, like this:

<script>location.href('http://www.hacked.com')</script>

The code is now safe to be displayed on a page or inside an e-mail.

We will also do two more things when the user submits the form:

1. Strip unnecessary characters (extra space, tab, newline) from the user input data (with the PHP trim() function)
2. Remove backslashes (\) from the user input data (with the PHP stripslashes() function)

The next step is to create a function that will do all the checking for us (which is much more convenient than writing the same code over and over again).

We will name the function test_input().

Now, we can check each $_POST variable with the test_input() function, and the script looks like this:

Example:

```
<html>
<head>
</head>
<body>

<?php
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["name"]);
  $email = test_input($_POST["email"]);
  $website = test_input($_POST["website"]);
  $comment = test_input($_POST["comment"]);
  $gender = test_input($_POST["gender"]);
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<h2>PHP Form Validation Example</h2>
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name">
  <br><br>
  E-mail: <input type="text" name="email">
  <br><br>
  Website: <input type="text" name="website">
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"></textarea>
  <br><br>
  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
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```

Required Form validation

From the validation rules table on the previous page, we see that the "Name", "E-mail", and "Gender" fields are required. These fields cannot be empty and must be filled out in the HTML form.

Example:

```
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
```

```
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
  }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field</span></p>
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name">
```

```
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  E-mail: <input type="text" name="email">
  <span class="error">* <?php echo $emailErr;?></span>
  <br><br>
  Website: <input type="text" name="website">
  <span class="error"><?php echo $websiteErr;?></span>
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"></textarea>
  <br><br>
  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
  <span class="error">* <?php echo $genderErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```

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";
}
```

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";
}
```

PHP Cookie

PHP cookie is a small piece of information which is stored at client browser. It is used to recognize the user.

PHP setcookie() function

PHP setcookie() function is used to set cookie with HTTP response. Once cookie is set, you can access it by $_COOKIE superglobal variable.

Syntax

bool setcookie (string $name [, string $value [, int $expire = 0 [, string $path

[, string $domain [, bool $secure = false [, bool $httponly = false]]]]])

Example

setcookie("CookieName", "CookieValue");/* defining name and value only*/

setcookie("CookieName", "CookieValue", time()+1*60*60);//using expiry in 1 hour(1*60*60 seconds or 3600 seconds)

setcookie("CookieName", "CookieValue", time()+1*60*60, "/mypath/", "mydomain.com", 1);

PHP $_COOKIE

PHP $_COOKIE superglobal variable is used to get cookie.

Example

$value=$_COOKIE["CookieName"];//returns cookie value

PHP Cookie Example

File: cookie1.php

<?php

```
setcookie("user", "Sonoo");

?>

<html>

<body>

<?php

if(!isset($_COOKIE["user"])) {

    echo "Sorry, cookie is not found!";

} else {

    echo "<br/>Cookie Value: " . $_COOKIE["user"];

}

?>

</body>

</html>
```

Output:

```
Sorry, cookie is not found!
```

Firstly cookie is not set. But, if you *refresh* the page, you will see cookie is set now.

PHP Delete Cookie

If you set the expiration date in past, cookie will be deleted.

File: cookie1.php

```
<?php

setcookie ("CookieName", "", time() - 3600);// set the expiration date to one hour ago

?>
```

PHP Session

PHP session is used to store and pass information from one page to another temporarily (until user close the website).

PHP session_start() function

PHP session_start() function is used to start the session. It starts a new or resumes existing session. It returns existing session if session is created already. If session is not available, it creates and returns new session.

Syntax

```
bool session_start ( void )
```

Example

```
session_start();
```

PHP $_SESSION

PHP $_SESSION is an associative array that contains all session variables. It is used to set and get session variable values.

Example: Store information

```
$_SESSION["user"] = "Sachin";
```

Example: Get information

```
echo $_SESSION["user"];
```

PHP Session Example

File: session1.php

```
<?php

session_start();

?>

<html>

<body>

<?php

$_SESSION["user"] = "Sachin";

echo "Session information are set successfully.<br/>";

?>

<a href="session2.php">Visit next page</a>

</body>

</html>
```

File: session2.php

```
<?php

session_start();

?>

<html>

<body>
```

```
<?php
echo "User is: ".$_SESSION["user"];
?>
</body>
</html>
```

PHP Session Counter Example

File: sessioncounter.php

```
<?php
  session_start();


  if (!isset($_SESSION['counter'])) {
    $_SESSION['counter'] = 1;
  } else {
    $_SESSION['counter']++;
  }
  echo ("Page Views: ".$_SESSION['counter']);
?>
```

PHP Destroying Session

PHP session_destroy() function is used to destroy all session variables completely.

File: session3.php

```
<?php

session_start();

session_destroy();

?>
```

PHP File Handling

PHP File System allows us to create file, read file line by line, read file character by character, write file, append file, delete file and close file.

PHP Open File - fopen()

The PHP fopen() function is used to open a file.

Syntax: resource fopen (string $filename , string $mode [, bool $use_include_path = false [, resource $context]])

Example

```
<?php

$handle = fopen("c:\\folder\\file.txt", "r");

?>
```

PHP Close File - fclose()

The PHP fclose() function is used to close an open file pointer.

Syntax

ool fclose (resource $handle)

Example

```
<?php

fclose($handle);

?>
```

PHP Read File - fread()

The PHP fread() function is used to read the content of the file. It accepts two arguments: resource and file size.

Syntax

```
string fread ( resource $handle , int $length )
```

Example

```
<?php

$filename = "c:\\myfile.txt";

$handle = fopen($filename, "r");//open file in read mode



$contents = fread($handle, filesize($filename));//read file



echo $contents;//printing data of file

fclose($handle);//close file

?>
```

PHP Write File - fwrite()

The PHP fwrite() function is used to write content of the string into file.

Syntax

int fwrite (resource $handle , string $string [, int $length])

Example

```
<?php

$fp = fopen('data.txt', 'w');//open file in write mode

fwrite($fp, 'hello ');

fwrite($fp, 'php file');

fclose($fp);



echo "File written successfully";

?>
```

PHP Delete File - unlink()

The PHP unlink() function is used to delete file.

Syntax: bool unlink (string $filename [, resource $context])

Example

```
<?php
unlink('data.txt');

echo "File deleted successfully";  ?>
```

PHP File Upload

PHP allows you to upload single and multiple files through few lines of code only.

PHP file upload features allows you to upload binary and text files both. Moreover, you can have the full control over the file to be uploaded through PHP authentication and file operation functions.

PHP $_FILES

The PHP global $_FILES contains all the information of file. By the help of $_FILES global, we can get file name, file type, file size, temp file name and errors associated with file.

Here, we are assuming that file name is *filename*.

$_FILES['filename']['name']

returns file name.

$_FILES['filename']['type']

returns MIME type of the file.

$_FILES['filename']['size']

returns size of the file (in bytes).

$_FILES['filename']['tmp_name']

returns temporary file name of the file which was stored on the server.

$_FILES['filename']['error']

returns error code associated with this file.

---

move_uploaded_file() function

The move_uploaded_file() function moves the uploaded file to a new location. The move_uploaded_file() function checks internally if the file is uploaded thorough the POST request. It moves the file if it is uploaded through the POST request.

Syntax

```
bool move_uploaded_file ( string $filename , string $destination )
```

PHP File Upload Example

File: uploadform.html

```
<form action="uploader.php" method="post" enctype="multipart/form-data">

   Select File:

   <input type="file" name="fileToUpload"/>

   <input type="submit" value="Upload Image" name="submit"/>

</form>
```

File: uploader.php

```
<?php

$target_path = "e:/";

$target_path = $target_path.basename( $_FILES['fileToUpload']['name']);

if(move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $target_path)) {

   echo "File uploaded successfully!";
```

```
} else{
    echo "Sorry, file not uploaded, please try again!";
}
?>
```