Compiled By:
Ranjan Raj Aryal
Additional Associate Professor
Cosmos College of Management and Technology

Unit 5: PHP

Origins and Uses of PHP:

- PHP was developed by Rasmus Lerdorf, a member of Apache Group.
- Originally PHP was acronym for Personal Home Page because developer first developed a package called Personal Home Page Tools.
- Later it is began using the PHP: **Hypertext Preprocessor.**

Uses:

- File Access: PHP can read and write files
- Database Access: Using PHP with MySQL we can read and write information in the database.
- PHP is great platform to develop static as well as dynamic websites easily.
- PHP runs on different platforms (Windows, Linux, Unix, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP is easy to learn and runs efficiently on the server side.

Overview of PHP:

- PHP stands for **P**HP: **H**ypertext **P**reprocessor
- PHP is a server-side scripting language.
- PHP scripts are executed on the server.
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is open source software.
- PHP is free to download and use.
- PHP files can contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML.
- PHP files have a file extension of ".php", ".php3", or ".phtml"
- PHP has an extensive library function, making it a flexible and powerful tool for server side software development.
- PHP use dynamic typing, as does JavaScript. Variables are not type declared. The type of variable is set every time it is assigned a value.
- PHP code can not be viewed by client when he views selected source, PHP codes are interpreted on server and result are displayed on clients browser. Clients can only views the result of PHP code in XHTML form.

General Syntactic characteristics:

Basic Syntax:

A PHP scripting block always starts with **<?php** and ends with **?>**. A PHP scripting block can be placed anywhere in the document.

A PHP file normally contains HTML tags, just like an HTML file, and some PHP scripting code.

Below, we have an example of a simple PHP script which sends the text "Hello World" to the browser:

```
<html>
<body>
<?php
echo "Hello World";
?>
</body>
</html>
```

Comments in PHP:

In PHP, we use // to make a single-line comment or /* and */ to make a large comment block.

```
<html>
<body>

<?php
//This is a comment

/*
This is
a comment
block
*/
?>

</body>
</html>
```

Primitives, Operations, and Expressions:

PHP has four scalar types: Boolean, integer, double and string.
PHP has two compound types: array and object.
PHP has two special types: resource and NULL(not required for now.)

Variables

All variables in PHP start with a $ sign symbol. The correct way of declaring a variable in PHP: $var_name = value;

In PHP, a variable does not need to be declared before adding a value to it.
PHP automatically converts the variable to the correct data type, depending on its value.
In PHP, the variable is declared automatically when you use it.

String

A string variable is used to store and manipulate text.

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

Operator and Expression:

1. Arithmetic Operator: **+, -, *, /, %, ++, --**

2. Comparison Operator: **= =, !=,** <, >, >=, <=

3. Logical Operator: &&, ||, !

4. Assignment Operator: **=, +=, -=, *=, /***
   1. *(C += A is equivalent to C = C + A)*

5. Conditional Operator: **? :**

```
$a = 10;
$b = 20;
/* If condition is true then assign a to result
otheriwse b */
$result = ($a > $b ) ? $a :$b;
```

6. Concatenation operator (**.**)

   There is only one string operator in PHP.

   The concatenation operator (**.**) is used to put two string values together.
   To concatenate two string variables together, use the concatenation operator:

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

Output: Hello World! What a nice day!

Decision Making statement:

1. If statement:

Syntax: if (*condition*) *code to be executed if condition is true;*

```
<?php
$d=date("D");
if ($d=="Fri") echo "Have a nice weekend!";
?>
```

2. If else:

Syntax:

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

Example:

```
<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
else
  echo "Have a nice day!";
?>
```

3. If else If:

Syntax:

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

Example:

```
<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
elseif ($d=="Sun")
  echo "Have a nice Sunday!";
else
  echo "Have a nice day!";
?>
```

4. Switch: If you want to select one of many blocks of code to be executed, use the Switch statement.

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;
}
```

Example:

```
<?php
switch ($x)
{
case 1:
  echo "Number 1";
  break;
case 2:
  echo "Number 2";
  break;
case 3:
  echo "Number 3";
  break;
default:
  echo "No number between 1 and 3";
}
?>
```

Looping Statement:

- **for -** loops through a block of code a specified number of times.

```
for (initialization; condition; increment)
{
  code to be executed;
}
```

- **while -** loops through a block of code if and as long as a specified condition is true.

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

- **do...while -** loops through a block of code once, and then repeats the loop as long as a special condition is trur.

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

- **foreach -** loops through a block of code for each element in an array.

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

}
```

Arrays in PHP:

An array is a special variable, which can store multiple values in one single variable. For example if you want to store 100 numbers then instead of defining 100 variables its easy to define an array of 100 length.

In PHP, there are three kind of arrays:

- **Numeric array** - An array with a numeric index
- **Associative array** - An array where each ID key is associated with a value
- **Multidimensional array** - An array containing one or more arrays

1. **Numeric Array**: A numeric array stores each array element with a numeric index. In the following example the index are automatically assigned (the index starts at 0):

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

Mannually:

```
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
```

To Display:

```
<?php
foreach( $cars as $value )
{
  echo "car name is $value <br />";
}
 ?>
```

2. Associative array: The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. An associative array, each ID key is associated with a value.

In this example we use an array to assign ages to the different persons:

```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

Different way:

$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";

To access Value in PHP:

```
<?php
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";

echo "Peter is " . $ages['Peter'] . " years old.";
?>
```

3. Multidimensional Array: In a multidimensional 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.

```
<?php
$families = array
  (
  "Griffin"=>array
  (
  "Peter",
  "Lois",
  "Megan"
  ),
  "Quagmire"=>array
  (
  "Glenn"
  ),
  "Brown"=>array
  (
  "Cleveland",
  "Loretta",
  "Junior"
  )
  );

echo "Is " . $families['Griffin'][2] .
" a part of the Griffin family?";

?>
```

<u>Function in PHP:</u>

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.

<u>Creating a function :</u>

Syntax:

```
function functionName()
{
code to be executed;
}
```

Example:

```
<?php
function writeName()
{
echo "Kai Jim Refsnes";
}

echo "My name is ";
writeName();
?>
```

Adding parameter to function:

```
<?php
function writeName($fname)
{
echo $fname . " Refsnes.<br />";
}

echo "My name is ";
writeName("Kai Jim");
echo "My sister's name is ";
writeName("Hege");
echo "My brother's name is ";
writeName("Stale");
?>
```

OUTPUT:

```
My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes.
My brother's name is Stale Refsnes.
```

Function Returning a value:

```
<?php
function addFunction($num1, $num2)
{
  $sum = $num1 + $num2;
  return $sum;
}
$return_value = addFunction(10, 20);
echo "Returned value from the function : $return_value
?>
```

OUTPUT: Returned value from the function : 30

PHP String

A PHP string is a sequence of characters i.e. used to store and manipulate text. There are 4 ways to specify string in PHP.

- single quoted
- double quoted
- heredoc syntax
- newdoc syntax (since PHP 5.3)

Single Quoted PHP String

We can create a string in PHP by enclosing text in a single quote. It is the easiest way to specify string in PHP.

```
        <?php
                $str='Hello text within single quote';
                echo $str;
?>
```

We can store multiple line text, special characters and escape sequences in a single quoted PHP string

```
        <?php
                $str1='Hello text
                multiple line
                text within single quoted string';
                $str2='Using double "quote" directly inside single quoted string';
                $str3='Using escape sequences \n in single quoted string';
                echo "$str1 <br/> $str2 <br/> $str3";
        ?>
```

Double Quoted PHP String

- In PHP, we can specify string through enclosing text within double quote also. But escape sequences and variables will be interpreted using double quote PHP strings.

```
        <?php
               $str="Hello text within double quote";
              echo $str;
        ?>
```

Note: you **can't use double quote directly** inside double quoted string.

```
<?php
        $str1="Using double "quote" directly inside double quoted string";
        echo $str1;
?>
```

This will create following error message:

Parse error: syntax error, unexpected 'quote' (T_STRING) in C:\wamp\www\string1.php on line 2

- We can store multiple line text, special characters and escape sequences in a double quoted PHP string.

```
<?php
$str1="Hello text
multiple line
text within double quoted string";
$str2="Using double \"quote\" with backslash inside double quoted string";
$str3="Using escape sequences \n in double quoted string";
echo "$str1 <br/> $str2 <br/> $str3";
?>
```

- In double quoted strings, **variable will be interpreted**

```
<?php
$num1=10;
echo "Number is: $num1";
?>
```

<u>PHP STRING FUNTIONS</u>

- PHP provides various string functions to access and manipulate strings. A list of important PHP string functions are given below.

1. The strtolower() function returns string in lowercase letter

Syntax:

```
string strtolower ( string $string )
```

Example:

```
<?php
$str="My name is KHAN";
$str=strtolower($str);
echo $str;
?>
```

2. The strtoupper() function returns string in uppercase letter.

Syntax:

```
string strtoupper ( string $string )
```

3. The ucfirst() function returns string converting first character into uppercase. It doesn't change the case of other characters.

Syntax:

```
string ucfirst ( string $str )
```

4. The lcfirst() function returns string converting first character into lowercase. It doesn't change the case of other characters.

Syntax:

```
string lcfirst ( string $str )
```

5. The ucwords() function returns string converting first character of each word into uppercase.
Syntax:

```
string ucwords ( string $str )
```

6. The strrev() function returns reversed string.
Syntax:

```
string ucwords ( string $str )
<?php
$str="my name is Sonoo jaiswal";
$str=strrev($str);
echo $str;
?>
```

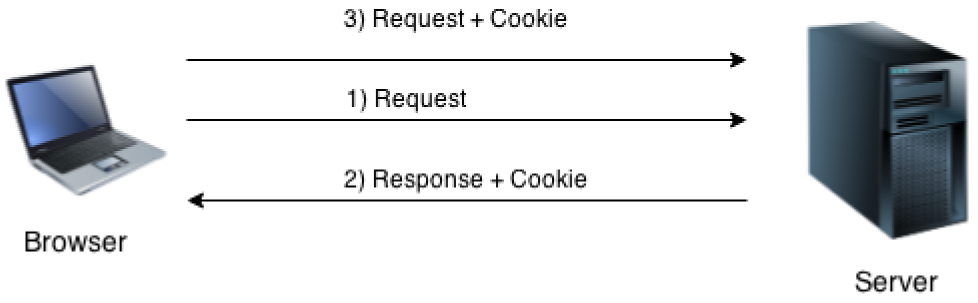
output

```
lawsiaj oonoS si eman ym
```

7. strlen() is used to return the length of a string.

PHP Cookie

- PHP cookie is a small piece of information which is stored at client browser. It is used to recognize the user.
- Cookie is created at server side and saved to client browser. Each time when client sends request to the server, cookie is embedded with request. Such way, cookie can be received at the server side.



In short, cookie can be created, sent and received at server end.

Note: PHP Cookie must be used before <html> tag.

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.

Example

1. setcookie("CookieName", "CookieValue");*/* defining name and value only*/*
2.setcookie("CookieName", "CookieValue", time()+1*60*60);*//using expiry in 1 hour (1*60*60 seconds or 3600 seconds)*
3.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*

Example program

```
<?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.
Output:
Cookie Value: Sonoo

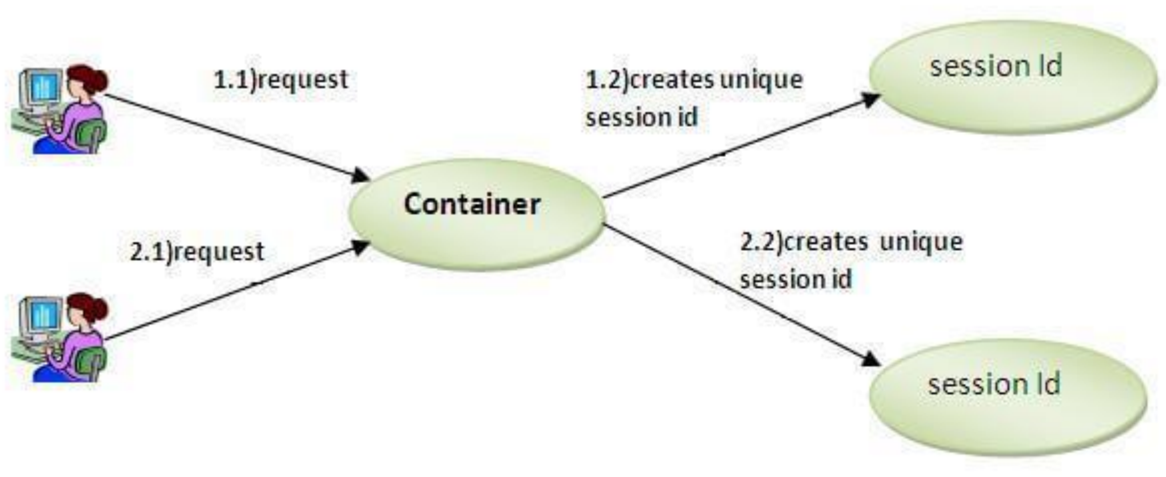
PHP Delete Cookie

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

```
<?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 technique is widely used in shopping websites where we need to store and pass cart information e.g. username, product code, product name, product price etc from one page to another.
- PHP session creates unique user id for each browser to recognize the user and avoid conflict between multiple browsers.



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 Program

1)

```
 <?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>
```

2)

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

PHP Destroying Session

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

```
<?php
session_start();
session_destroy();
?>
```

PHP MySQL Connect:

Since PHP 5.5, **mysql_connect()** extension is *deprecated*. Now it is recommended to use one of the 2 alternatives.

- **mysqli_connect()**
- **PDO::__construct()**

PHP mysqli_connect()

- PHP **mysqli_connect() function** is used to connect with MySQL database. It returns *resource* if connection is established or *null*.

Syntax

resource mysqli_connect (server, username, password)

PHP mysqli_close()

- PHP **mysqli_close() function** is used to disconnect with MySQL database. It returns *true* if connection is closed or *false*.

Syntax

bool mysqli_close(resource $resource_link)

PHP MySQL Connect Example program:

```
<?php
$host = 'localhost:3306';
$user = '';
$pass = '';
$conn = mysqli_connect($host, $user, $pass);
if(! $conn )
{
  die('Could not connect: ' . mysqli_error());
}
echo 'Connected successfully';
mysqli_close($conn);
?>
```

output

```
Connected successfully
```

PHP MySQLi Create Database Example

```
<?php
$host = 'localhost:3306';
$user = '';
$pass = '';
$conn = mysqli_connect($host, $user, $pass);
if(! $conn )
{
  die('Could not connect: ' . mysqli_connect_error());
}
echo 'Connected successfully<br/>';

$sql = 'CREATE Database mydb';
if(mysqli_query( $conn,$sql)){
  echo "Database mydb created successfully.";
}else{
echo "Sorry, database creation failed ".mysqli_error($conn);
}
mysqli_close($conn);
?>
```

Output:

```
Connected successfully
Database mydb created successfully.
```

PHP MySQLi Create Table Example

```
<?php
$host = 'localhost:3306';
$user = '';
$pass = '';
$dbname = 'test';

$conn = mysqli_connect($host, $user, $pass,$dbname);
if(!$conn){
  die('Could not connect: '.mysqli_connect_error());
}
echo 'Connected successfully<br/>';

$sql = "create table emp5(id INT AUTO_INCREMENT,name VARCHAR(20) NOT NULL,
emp_salary INT NOT NULL,primary key (id))";
if(mysqli_query($conn, $sql)){
 echo "Table emp5 created successfully";
}else{
echo "Could not create table: ". mysqli_error($conn);
}
mysqli_close($conn);
?>
```

Output:

```
Connected successfully
Table emp5 created successfully
```

PHP MySQLi Select Query Example

```
<?php
```

```
$host = 'localhost:3306';
$user = '';
$pass = '';
$dbname = 'test';
$conn = mysqli_connect($host, $user, $pass,$dbname);
if(!$conn){
  die('Could not connect: '.mysqli_connect_error());
}
echo 'Connected successfully<br/>';

$sql = 'SELECT * FROM emp4';
$retval=mysqli_query($conn, $sql);

if(mysqli_num_rows($retval) > 0){
 while($row = mysqli_fetch_assoc($retval)){
    echo "EMP ID :{$row['id']}  <br> ".
         "EMP NAME : {$row['name']} <br> ".
         "EMP SALARY : {$row['salary']} <br> ".
         "--------------------------------<br>";
 } //end of while
}else{
echo "0 results";
}
mysqli_close($conn);
?>
```

PHP MySQLi Insert Record Example

```
<?php
$host = 'localhost:3306';
$user = '';
$pass = '';
$dbname = 'test';

$conn = mysqli_connect($host, $user, $pass,$dbname);
if(!$conn){
  die('Could not connect: '.mysqli_connect_error());
}
echo 'Connected successfully<br/>';
```

```
$sql = 'INSERT INTO emp4(name,salary) VALUES ("sonoo", 9000)';
if(mysqli_query($conn, $sql)){
 echo "Record inserted successfully";
}else{
echo "Could not insert record: ". mysqli_error($conn);
}

mysqli_close($conn);
?>
```

PHP MySQLi Update Record Example

```
<?php
$host = 'localhost:3306';
$user = '';
$pass = '';
$dbname = 'test';

$conn = mysqli_connect($host, $user, $pass,$dbname);
if(!$conn){
  die('Could not connect: '.mysqli_connect_error());
}
echo 'Connected successfully<br/>';

$id=2;
$name="Rahul";
$salary=80000;
$sql = "update emp4 set name=\"$name\", salary=$salary where id=$id";
if(mysqli_query($conn, $sql)){
 echo "Record updated successfully";
}else{
echo "Could not update record: ". mysqli_error($conn);
}

mysqli_close($conn);
?>
```

PHP MySQLi Delete Record Example

```
<?php
$host = 'localhost:3306';
$user = '';
$pass = '';
```

```
$dbname = 'test';

$conn = mysqli_connect($host, $user, $pass,$dbname);
if(!$conn){
  die('Could not connect: '.mysqli_connect_error());
}
echo 'Connected successfully<br/>';

$id=2;
$sql = "delete from emp4 where id=$id";
if(mysqli_query($conn, $sql)){
 echo "Record deleted successfully";
}else{
echo "Could not deleted record: ". mysqli_error($conn);
}

mysqli_close($conn);
?>
```

PHP JSON

- A common use of JSON is to read data from a web server, and display the data in a web page.
- PHP allows us to encode and decode JSON by the help of json_encode() and json_decode functions.

1) PHP json_encode

- The json_encode() function returns the JSON representation of a value. In other words, it converts PHP variable (containing array) into JSON.

Syntax:

string json_encode (mixed $value [, **int** $options = 0 [, **int** $depth = 512]])

PHP json_encode example 1

```
<?php
$arr = array('a' => 1, 'b' => 2, 'c' => 3, 'd' => 4, 'e' => 5);
echo json_encode($arr);
?>
```

Output

```
{"a":1,"b":2,"c":3,"d":4,"e":5}
```

PHP json_encode example 2

```
<?php
$arr2 = array('firstName' => 'Rahul', 'lastName' => 'Kumar', 'email' =>
'rahul@gmail.com');
echo json_encode($arr2);
?>
```

Output

{"firstName":"Rahul","lastName":"Kumar","email":"rahul@gmail.com"}

2) PHP json_decode

The json_decode() function decodes the JSON string. In other words, it converts JSON string into a PHP variable.

Syntax:

mixed json_decode (string $json [, bool $assoc = false [, int $depth = 512 [, int $options = 0]]])

PHP json_decode example 1

```
<?php
$json = '{"a":1,"b":2,"c":3,"d":4,"e":5}';
var_dump(json_decode($json, true));//true means returned object will be
converted into associative array
?>
```

Output

```
array(5) {
    ["a"] => int(1)
    ["b"] => int(2)
    ["c"] => int(3)
    ["d"] => int(4)
    ["e"] => int(5)
}
```

PHP - GET & POST Methods

There are two ways the browser client can send information to the web server.

- The GET Method
- The POST Method

Before the browser sends the information, it encodes it using a scheme called URL encoding. In this scheme, name/value pairs are joined with equal signs and different pairs are separated by the ampersand.

The GET Method

- The GET method produces a long string that appears in your server logs, in the browser's Location: box.
- The GET method is restricted to send upto 1024 characters only.
- Never use GET method if you have password or other sensitive information to be sent to the server.
- GET can't be used to send binary data, like images or word documents, to the server.
- The data sent by GET method can be accessed using QUERY_STRING environment variable.
- The PHP provides **$_GET** associative array to access all the sent information using GET method.

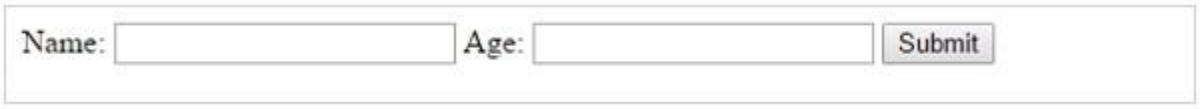
```
<?php
   if( $_GET["name"] || $_GET["age"] ) {
      echo "Welcome ". $_GET['name']. "<br />";
      echo "You are ". $_GET['age']. " years old.";

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

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

   </body>
</html>
```

It will produce the following result –



<u>The POST Method</u>

The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.

- The POST method does not have any restriction on data size to be sent.
- The POST method can be used to send ASCII as well as binary data.

- The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.
- The PHP provides **$_POST** associative array to access all the sent information using POST method.

```
<?php

   if( $_POST["name"] || $_POST["age"] ) {

      if (preg_match("/[^A-Za-z'-]/",$_POST['name'] )) {

         die ("invalid name and name should be alpha");

      }

      echo "Welcome ". $_POST['name']. "<br />";

      echo "You are ". $_POST['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" />

      </form>


   </body>

</html>
```

it will produce the following result −

PHP File Handling

- PHP has several functions for creating, reading, uploading, and editing files.
- You often need to open and process a file for different tasks.
- You can do a lot of damage if you do something wrong. Common errors are: editing the wrong file, filling a hard-drive with garbage data, and deleting the content of a file by accident.

PHP readfile() Function

- The readfile() function reads a file and writes it to the output buffer.
- This function is useful to open up a file and read its contents

Example

```
<?php
echo readfile("sample.txt");
?>
```

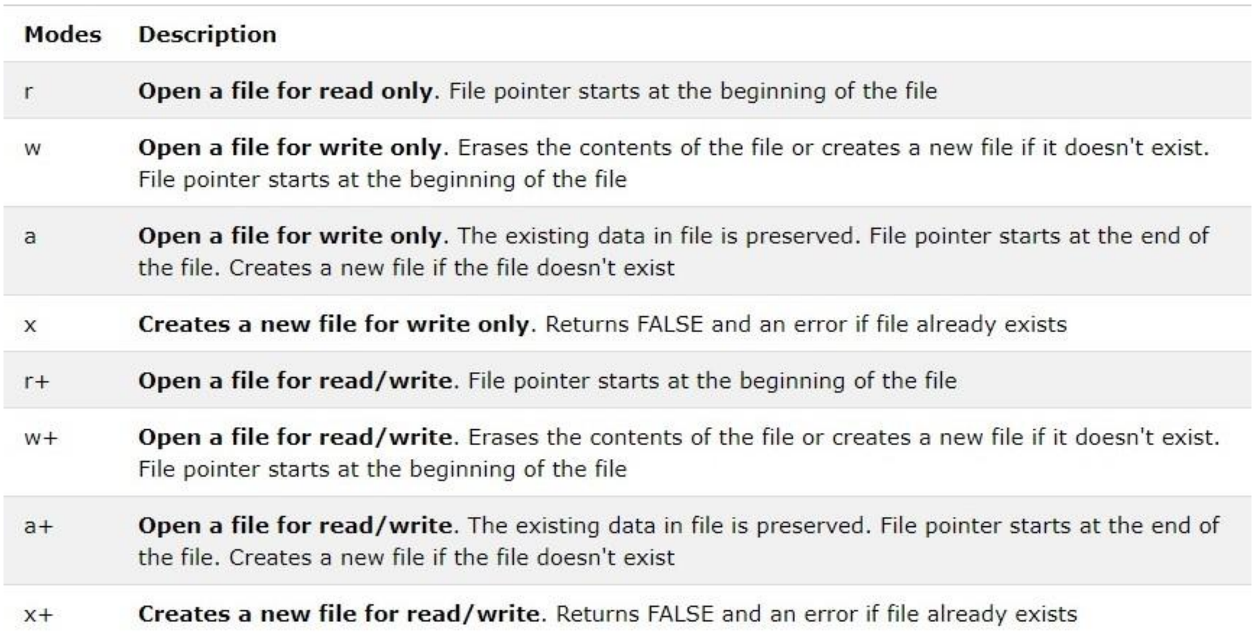
PHP Open File - fopen()

- A better method to open files is with the fopen() function. This function gives you more options than the readfile() function.

Example

```
<?php
$myfile = fopen("sample.txt", "r") or die("Unable to open
file!");
echo fread($myfile,filesize("sample.txt"));
fclose($myfile);
?>
```

The file may be opened in one of the following modes:



Modes	Description
r	**Open a file for read only**. File pointer starts at the beginning of the file
w	**Open a file for write only**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a	**Open a file for write only**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x	**Creates a new file for write only**. Returns FALSE and an error if file already exists
r+	**Open a file for read/write**. File pointer starts at the beginning of the file
w+	**Open a file for read/write**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a+	**Open a file for read/write**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x+	**Creates a new file for read/write**. Returns FALSE and an error if file already exists

PHP Read File - fread()

- The fread() function reads from an open file.
- The first parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.
- The following PHP code reads the "sample.txt" file to the end:

Example:
```
fread($myfile,filesize("sample.txt"));
```

PHP Close File - fclose()

- The `fclose()` function is used to close an open file.
- It's a good programming practice to close all files after you have finished with them. You don't want an open file running around on your server taking up resources!
- The fclose() requires the name of the file (or a variable that holds the filename) we want to close:

Example
```
<?php
$myfile = fopen("sample.txt", "r");
// some code to be executed....
```

```
fclose($myfile);
?>
```

PHP Check End-Of-File - feof()

- The feof() function checks if the "end-of-file" (EOF) has been reached.
- The feof() function is useful for looping through data of unknown length.
- The example below reads the "sample.txt" file line by line, until end-of-file is reached:

Example

```
<?php
$myfile = fopen("sample.txt", "r") or die("Unable to open file!");
// Output one line until end-of-file
while(!feof($myfile)) {
  echo fgets($myfile) . "<br>";
}
fclose($myfile);
?>
```