Introduction

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
- 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"

What Can PHP Do?

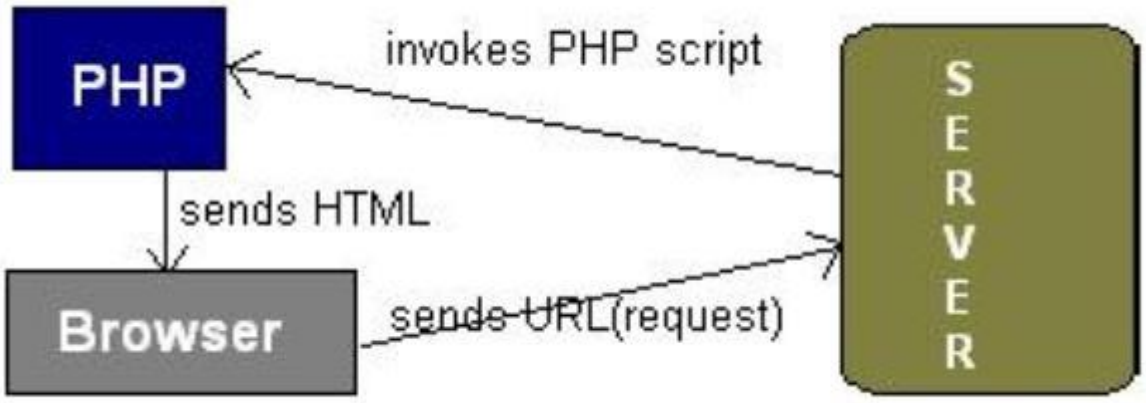
- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access

Why PHP?

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports 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

- **Note:** PHP statements end with a semicolon (;).

Working of PHP…



- URL is typed in the browser.
- The browser sends a request to the web server.
- The web server then calls the PHP script on that page.
- The PHP module executes the script, which then sends out the result in the form of HTML back to the browser, which can be seen on the screen.

SIDE by SIDE

- PHP File:
- <html>
- <head>
- <title> PHP Introduction </title> </head>
- <body>
- This is HTML!
- **<?php**
- **echo 'This is PHP!
';**
- **?>**
- **</body>**
- **</html>**

- Output: resulting HTML
- <html>
- <head>
- <title> PHP Introduction </title> </head>
- <body>
- This is HTML!
- **This is PHP!
**
- **</body>**
- **</html>**

PHP BASICS…

- **SYNTAX:**

<?php
s1; s2; ?>
PHP only parses code within its delimiters.

PHP Basics includes:

1) CONSTANTS:

- Named with capital letters.
- Must begin with a letter or underscore .
- Cannot begin with a number.
- Case-sensitive.

Eg: define (“FAVMOVIE”, “The Life of Brian”);

2) VARIABLES:

- Prefixed with a dollar symbol.
- Type not to be specified.
- Variable name should not have spaces, dot or dashes but underscore can be there.
- They need to be declared before adding a value to it.

Example: $s = “SR”;

3) DATATYPES:

- String
- Integer
- Boolean
- Float
- Object
- Resources

4) OPERATORS and OPERANDS:

Operands are the entities that have some values in them. The operators are used to compare the two conditions .

5) COMMENTS:

// A comment on a single line
Another single line comment
/* Multi-line comment */

6) DISPLAY STATEMENTS:

<?php echo "I like About" ?>
<?php print "I like About" ?>

7) ARRAYS:

It holds a string of related data.

8) CONDITIONAL STATEMENTS:

It allows our program to make choices.

9) LOOPS:

When we want the same block of code to run over & over again in a sequence.

- **while** - loops through a block of code while the 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.
- **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

10) FUNCTIONS:

A function is something that performs a specific task.

STRUCTURE OF PHP

Run

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

<h1>My first PHP page</h1>

<?php
echo "Hello World! <br/>";
print "vinodkumar Kp";
?>

</body>
</html>
```

My first PHP page

Hello World!
vinodkumar Kp

PhP case sensitivity;

- PHP Case Sensitivity
- In PHP, keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are not case-sensitive.
- **Note:** However; all variable names are case-sensitive!

Run

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

<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>

</body>
</html>
```

Hello World!

Hello World!

Hello World!

Establishing sec...

- Comments in php

Run

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

<?php
print"comments in php <br>";
print"single line comment<br>";
//sinngle line
#single line
print"multiple line comments<br>";
/*
This is a multiple-lines
comment
*/
?>

</body>
</html>
```

comments in php
single line comment
multiple line comments

PHP Variables

Run

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

<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;

echo $txt;
echo "<br>";
echo $x;
echo "<br>";
echo $y;
echo "<br>";
echo $x+$y;
echo "<br>";
echo " hi vinod $txt ";
?>

</body>
</html>
```

Hello world!
5
10.5
15.5
hi vinod Hello world!

PHP output

Run

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

<?php
print "<h2>PHP is Fun!</h2>";
print "Hello world!<br><br>
hi vinod<br>";
print "I'm about to learn PHP! <br>";
print(aaa);
print "<br>";
print(47);
?>

</body>
</html>
```

PHP is Fun!

Hello world!

hi vinod
I'm about to learn PHP!
aaa
47

Run

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

<?php
$day= "wednesday";
$month="october";
print $day;
print "<br>";
// php borrows printf from c
printf("%s",$month);
print"<br>";
$number=12345;
printf("%d",$number);
print "<br>";

?>

</body>
</html>
```

wednesday
october
12345

PHP Data Types

PHP supports the following data types:
String
Integer
Float (floating point numbers - also called double)
Boolean
Array
Object
NULL
Resource

PHP String

A string is a sequence of characters, like "Hello world!".

A string can be any text inside quotes. You can use single or double quotes:

Run

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

<?php
$x = "Hello world!";
$y = 'Hello world!';

echo $x;
echo "<br>";
echo $y;
?>

</body>
</html>
```

Hello world!
Hello world!

Waiting for securepubads.g.doubleclick.net...

PHP Integer

An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

Rules for integers:

An integer must have at least one digit

An integer must not have a decimal point

An integer can be either positive or negative

Integers can be specified in: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation

In the following example $x is an integer. The PHP var_dump() function returns the data type and value:

Run

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

<?php
$x = 5985;
$y = 10.5;
var_dump($x,$y);
?>

</body>
</html>
```

int(5985) float(10.5)

Php array:

Run ❯ Result Size: 1346 x 246 Get your own website

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

<?php
$cars = array("Volvo","BMW","Toyota");
var_dump($cars);

?>

</body>
</html>
```

array(3) { [0]=> string(5) "Volvo" [1]=> string(3) "BMW" [2]=> string(6) "Toyota" }

3:36 PM
10/22/2021

LOOP:

Run

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

<?php
for ($x = 0; $x <= 10; $x++) {
  echo "The number is: $x <br>";
}
?>

</body>
</html>
```

The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
The number is: 6
The number is: 7
The number is: 8
The number is: 9
The number is: 10

Php arrays:

Run

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

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

</body>
</html>
```

I like Volvo
I like Volvo

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

Run

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

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

</body>
</html>
```

3

PHP Indexed Arrays

Run

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

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

for($x = 0; $x < $arrlength; $x++) {
  echo $cars[$x];
  echo "<br>";
}
?>

</body>
</html>
```

Volvo
BMW
Toyota

PHP Associative Arrays

Run

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

<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>

</body>
</html>
```

Peter is 35 years old.

PHP Multidimensional Arrays

Run

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

<?php
$cars = array (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
);

echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
?>

</body>
</html>
```

Volvo: In stock: 22, sold: 18.

BMW: In stock: 15, sold: 13.

Saab: In stock: 5, sold: 2.

Land Rover: In stock: 17, sold: 15.

Php functions:

Run

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

<?php
function writeMsg() {
  echo "Hello world! <br>";
}

writeMsg();
//functional call take 2 arguments

function familyName($fname, $year) {
  echo "$fname Refsnes. Born in $year <br>";
}

familyName("Hege","1975");
familyName("Stale","1978");
familyName("Kai Jim","1983");


?>

</body>
</html>
```

Hello world!
Hege Refsnes. Born in 1975
Stale Refsnes. Born in 1978
Kai Jim Refsnes. Born in 1983

PHP Form Handling

- $_GET
- $_POST

Display of welcome.html

```html
<!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>
```

Name:
E-mail:
Submit

javascript:void(0)

Display of welcome.php

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

  </body>
  </html>
```

- Output:

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

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

</body>
</html>
```

Welcome vinodkumar.kp
Your email address is: vinodkumar.kp@dr-ait.org

PHP Cookies

- 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.
- Finally it stores user information.

- PHP Create/Retrieve a Cookie
- The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 * 30).
- We then retrieve the value of the cookie "user" (using the global variable $_COOKIE). We also use the isset() function to find out if the cookie is set:

- 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.
- **Note:** The setcookie() function must appear BEFORE the <html> tag.

- Example :

```
<!DOCTYPE html>
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 =
1 day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
     echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
     echo "Cookie '" . $cookie_name . "' is set!<br>";
     echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

<p><strong>Note:</strong> You might have to reload the page to see the value of
the cookie.</p>

</body>
</html>
```

output

```
<!DOCTYPE html>
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
     echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
```

Cookie named 'user' is not set!

Note: You might have to reload the page to see the value of the cookie.

PHP Sessions

- A session is a way to store information (in variables) to be used across multiple pages.
- Unlike a cookie, the information is not stored on the users computer.
- HTTP is stateless
- 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).

Start a PHP Session

- A session is started with the session_start() function.
- Session variables are set with the PHP global variable: $_SESSION.
- **Note:** The session_start() function must be the very first thing in your document. Before any HTML tags.

demo_session1.php

```php
<?php
// 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>
```

Session variables are set.

PHP FILES:

- PHP Manipulating Files
- PHP has several functions for creating, reading, uploading, and editing files.
- PHP readfile() Function
- The readfile() function reads a file and writes it to the output buffer.
- 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.

We will use the text file, "webdictionary.txt", during the lessons:

AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language

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

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

</body>
</html>
```

OUTPUT:

- AJAX = Asynchronous JavaScript and XML CSS = Cascading Style Sheets HTML = Hyper Text Markup Language PHP = PHP Hypertext Preprocessor SQL = Structured Query Language SVG = Scalable Vector Graphics XML = EXtensible Markup Language

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

- fclose() requires the name of the file (or a variable that holds the filename) we want to close:
- <?php
  $myfile = fopen("webdictionary.txt", "r");
  // some code to be executed....
  fclose($myfile);
  ?>

- PHP Read Single Line - fgets()
- The fgets() function is used to read a single line from a file.

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

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

</body>
</html>
```

AJAX = Asynchronous JavaScript and XML

- PHP Write to File - fwrite()
- The fwrite() function is used to write to a file.
- The first parameter of fwrite() contains the name of the file to write to and the second parameter is the string to be written.

Jquery

- jQuery is a JavaScript Library.
- jQuery greatly simplifies JavaScript programming.
- jQuery is easy to learn.
- The purpose of jQuery is to make it much easier to use JavaScript on your website.

- Before you start studying jQuery, you should have a basic knowledge of:
- HTML
- CSS
- JavaScript

- What is jQuery?
- jQuery is a lightweight, "write less, do more", JavaScript library.
- The purpose of jQuery is to make it much easier to use JavaScript on your website.
- jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.

- jQuery Syntax
- With jQuery you select (query) HTML elements and perform "actions" on them.
- The jQuery syntax is tailor-made for **selecting** HTML elements and performing some **action** on the element(s).
- Basic syntax is: **$(*selector*).*action*()**
- A $ sign to define/access jQuery
- A (*selector*) to "query (or find)" HTML elements
- A jQuery *action*() to be performed on the element(s)

- $(this).hide() - hides the current element.
- $("p").hide() - hides all <p> elements.
- $(".test").hide() - hides all elements with class="test".
- $("#test").hide() - hides the element with id="test".