**Part 7- Practical 2**

**Aim:** Study and demonstrate php syntax, data type, variable, function, array, superglobal variable and form (Self-learning exercise using this link: <https://www.w3schools.com/php/>).

**Basic PHP Syntax**

A PHP script can be placed anywhere in the document.

A PHP script starts with <?php and ends with ?>:

<?php

// PHP code goes here

?>

The default file extension for PHP files is ".php".

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

Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page:

<!DOCTYPE html>

<html>

<body>

<h1>My first PHP page</h1>

<?php  
echo "Hello World!";  
?>

</body>

</html>

**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
* Resource

SOME FEW DATATYPES' EXAMPLES ARE AS FOLLOWS:

String:

<?php

$x = "Hello world!";

$y = 'Hello world!';

echo $x;

echo "<br>";

echo $y;

?>

Integer:

<?php  
$x = 5985;  
var\_dump($x);  
?>

**Creating (Declaring) PHP Variables**

In PHP, a variable starts with the $ sign, followed by the name of the variable:

<?php

$txt = "Hello world!";

$x = 5;

$y = 10.5;

?>

After the execution of the statements above, the variable $txt will hold the value Hello world!, the variable $x will hold the value 5, and the variable $y will hold the value 10.5.

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

**Note:** Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.

Rules for PHP variables:

* A variable starts with the $ sign, followed by the name of the variable
* A variable name must start with a letter or the underscore character
* A variable name cannot start with a number
* A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )
* Variable names are case-sensitive ($age and $AGE are two different variables)

**PHP Built-in Functions**

PHP has over 1000 built-in functions that can be called directly, from within a script, to perform a specific task.

**PHP User Defined Functions**

Besides the built-in PHP functions, it is possible to create your own functions.

* A function is a block of statements that can be used repeatedly in a program.
* A function will not execute automatically when a page loads.
* A function will be executed by a call to the function.

**Create a User Defined Function in PHP**

A user-defined function declaration starts with the word function:

Syntax

function *functionName*() {

*code to be executed*;

}

**PHP Arrays**

An array stores multiple values in one single variable:

<?php

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

echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";

?>

**PHP Global Variables - Superglobals**

Some predefined variables in PHP 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.

The PHP superglobal variables are:

* $GLOBALS
* $\_SERVER
* $\_REQUEST
* $\_POST
* $\_GET
* $\_FILES
* $\_ENV
* $\_COOKIE
* $\_SESSION

**PHP Form Handling**

The PHP superglobals $\_GET and $\_POST are used to collect form-data.

The example below displays a simple HTML form with two input fields and a submit button:

<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>

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>

The output could be something like this:

Welcome John

Your email address is john.doe@example.com

**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.