PHP : **H**ypertext **P**reprocessor

What is PHP?

- PHP stands for **P**HP: **H**ypertext **P**reprocessor
- 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, JavaScript code, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain HTML
- PHP files have a default file extension of ".php"

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, 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 restrict users to access some pages on your website
- PHP can encrypt data

Why PHP?

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

Set Up PHP on Your Own PC

- However, if your server does not support PHP, you must:
- install a web server
- install PHP
- install a database, such as MySQL
- The official PHP website (PHP.net) has installation instructions for
  PHP: http://php.net/manual/en/install.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".

Comments in PHP

- Example

```
<html>
    <body>
    <?php
        //This is a PHP comment line
        /*
            This is a PHP comment
            block
        */
    ?>
    </body>
</html>
```

PHP Variables

- Variable can have short names (like x and y) or more descriptive names (age, carname, totalvolume).
- Rules for PHP variables:
  - A variable starts with the $ sign, followed by the name of the variable
  - A variable name must begin with a letter or the underscore character
  - A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
  - A variable name should not contain spaces
  - Variable names are case sensitive ($y and $Y are two different variables)

Creating (Declaring) PHP Variables

- PHP has no command for declaring a variable.
- A variable is created the moment you first assign a value to it:
- $txt="Hello world!";
  $x=5;
- After the execution of the statements above, the variable **txt** will hold the value **Hello world!**, and the variable **x**will hold the value **5**.
- **Note:** When you assign a text value to a variable, put quotes around the value.

PHP is a Loosely Typed Language

- In the example above, notice that we did not have to tell PHP which data type the variable is.
- PHP automatically converts the variable to the correct data type, depending on its value.
- In a strongly typed programming language, we will have to declare (define) the type and name of the variable before using it.

PHP Variable Scopes

- The scope of a variable is the part of the script where the variable can be referenced/used.
- PHP has four different variable scopes:
- local
- global
- static
- parameter

Local Scope

- A variable declared **within** a PHP function is local and can only be accessed within that function:
- Example

```
<?php
    $x=5; // global scope

    function myTest()
    {
            echo $x; // local scope
    }

    myTest();
?>
```

Global Scope

- A variable that is defined outside of any function, has a global scope.
- Global variables can be accessed from any part of the script, EXCEPT from within a function.
- To access a global variable from within a function, use the **global** keyword:

Global Scope

- Example
- <?php

```
$x=5; // global scope
$y=10; // global scope

function myTest()
{
        global $x,$y;
        $y=$x+$y;
}

myTest();
echo $y;                    // outputs 15
```

?>

Static Scope

- When a function is completed, all of its variables are normally deleted. However, sometimes you want a local variable to not be deleted.
- To do this, use the **static** keyword when you first declare the variable:

Static Scope

- Example

```
<?php
        function myTest()
                {
                        static $x=0;
                        echo $x;
                        $x++;
                }

        myTest();
        myTest();
        myTest();

?>
```

- OUTPUT ?

Parameter Scope

- A parameter is a local variable whose value is passed to the function by the calling code.
- Parameters are declared in a parameter list as part of the function declaration:
- Example :

```
<?php
        function myTest($x)
            {
                echo $x;
            }
        myTest(5);
?>
```

PHP echo and print Statements

- In PHP there is two basic ways to get output: echo and print.
- There are some differences between echo and print:
  - echo - can output one or more strings
  - print - can only output one string, and returns always 1
- **Tip:** echo is marginally faster compared to print as echo does not return any value.

```
<?php
   echo "<h2>PHP is fun!</h2>";
   echo "Hello world!<br>";
   echo "This", " string", " was", " made", " with multiple
   parameters.";
?>
```

PHP Arrays

- An array stores multiple values in one single variable:

Example

```
<?php
$cars = array("Volvo","BMW","Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " .
$cars[2] . ".";
?>
```

Create an Array in PHP

- In PHP, the array() function is used to create an array:
- Example :

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

- In PHP, there are three types of arrays:
  - **Indexed arrays** - Arrays with numeric index
  - **Associative arrays** - Arrays with named keys
  - **Multidimensional arrays** - Arrays containing one or more arrays

PHP Indexed Arrays

- There are two ways to create indexed arrays:
  - The index can be assigned automatically (index always starts at 0):
  - $cars=array("Volvo","BMW","Toyota");

OR

  - The index can be assigned manually:

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

PHP Associative Arrays

- Associative arrays are arrays that use named keys that you assign to them.
- There are two ways to create an associative array:
  - $age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");

  OR

  - $age['Peter']="35";
    $age['Ben']="37";
    $age['Joe']="43";

PHP Associative Arrays

- Example

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

PHP Multidimensional Arrays

- An array can also contain another array as a value, which in turn can hold other arrays as well. In such a way we can create two- or three-dimensional arrays:

  Example

```
<?php
  // A two-dimensional array:
  $cars = array
    (
    array("Volvo",100,96),
    array("BMW",60,59),
    array("Toyota",110,100)
    );
?>
```

The foreach Loop

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

```
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) - so on the next loop iteration, you'll be looking at the next array value.

The foreach Loop

Example:

```
<?php
   $x=array("one","two","three");
   foreach ($x as $value)
     {
     echo $value . "<br>";
     }
   ?>
```

PHP Functions

- PHP User Defined Functions
  - Besides the built-in PHP functions, we can create our own functions.
  - A function is a block of statements that can be used repeatedly in a program.
  - A function will not execute immediately 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(arg1, arg2…)
  {
        code to be executed;
  }
```

  - **Note:** A function name can start with a letter or underscore (not a number).

GET Vs POST

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because data sent is part of the URL Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

PHP $_GET Variable

- The predefined $_GET variable is used to collect values in a form with method="get"
- Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and has limits on the amount of information to send.
- Example

```
<form action="welcome.php" method="get">
   Name: <input type="text" name="fname">
   Age: <input type="text" name="age">
   <input type="submit">
   </form>
```

- When the user clicks the "Submit" button, the URL sent to the server could look something like this:

  http://www.w3schools.com/welcome.php?fname=Peter&age=37

- The "welcome.php" file can now use the $_GET variable to collect form data as follow:

```
Welcome <?php echo $_GET["fname"]; ?>.<br>
You are <?php echo $_GET["age"]; ?> years old!
```

The $_POST Variable

- The predefined $_POST variable is used to collect values from a form sent with method="post".
- Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.
- **Note:** However, there is an 8 MB max size for the POST method, by default (can be changed by setting the post_max_size in the php.ini file).