Ch2 - Lesson 1:
Server Side Scripting Basics

OVERVIEW

- **Webpage:**
  - Document, typically written in HTML that is almost always accessible via HTTP.
  - Pages on which, information is displayed on the web.
  - Can be static or dynamic.

OVERVIEW

- **Scripting language :**
  - A programming language in a simple text format.
  - Code written in a scripting language does not have to be compiled, but interpreted.
  - Scripts can be written to run either **server-side** or **client-side.**
  - A script must be interpreted at the time it is requested from the web server.

OVERVIEW

- **Scripting language :**
  - A **client-side script** is executed on the client, by the browser.
  - **Client-scripting**
    - Is often used to validate data entered on a form by the user, before the form is submitted to the server
    - Present data on a browser and manage interactions between user and application

OVERVIEW

- **Server side scripting:**
  - Is executed on the server and produces HTML and which is then output HTML to the client.

INTRODUCTION TO SERVER SIDE SCRIPTING

- **Server-side scripting:**
  - A web technology in which a user's request is fulfilled by running a script directly on the web server to generate **dynamic webpages**.
  - Use to develop interactive web sites that interface to databases or other data stores.
  - The primary advantage of server-side scripting is the ability to highly customize the response based on the user's requirements, access rights, or queries from data stores.

INTRODUCTION TO SERVER SIDE SCRIPTING

- **Server-side scripting:**
  - From security point of view, they are never visible to the browser as these scripts are executed on the server and produce HTML corresponding to user's input to the page.
  - They are written in Java, Asp.Net, PHP, ColdFusion, Perl, Ruby, Go, Python
  - Executed by a web server when the user requests a document

INTRODUCTION TO SERVER SIDE SCRIPTING

- **Server-side scripting:**
  - They produce output in a format understandable by web browsers (usually HTML)
  - The user cannot see the script's source and may not even be aware that a script was executed.
  - Documents produced by server-side scripts may, in turn, contain client-side scripts.

INTRODUCTION TO SERVER SIDE SCRIPTING

- **Server-side scripting:**
  - Its mostly about connecting websites with back end services such as databases and data sources
  - Enables two way communication:
    - **Client to server** → users request to server for data and resources
    - **Server to client** → responses sent to users computer

INTRODUCTION TO SERVER SIDE SCRIPTING

- **Server-side scripting:**
  - A server side script can:
    - Dynamically edit, change or add any content to a Web page to make it more useful for individual users
    - Respond to user queries or data submitted from HTML forms
    - Access any data or databases and return the result to a browser
    - Provide security since server side codes cannot be viewed from a browser

INTRODUCTION TO SERVER SIDE SCRIPTING

- Server-side script languages:

- ❑PHP
- ❑ASP
- ❑Java (JSP)
- ❑JS using server side scripts
- ❑Perl, Ruby
- ❑Go, Python

Our focus : php

WHAT IS PHP

- **What is PHP?**
  - Hypertext Preprocessor
  - **Preprocessor** → script runs on the web server, not on the users computer
  - Works with many databases.
    - Eg. **MySQL**, **Informix**, **Oracle**, **Sybase**, **Solid**, **PostgreSQL**, **Generic ODBC**, **Microsoft SQL Server**
  - PHP files can contain text, HTML tags and scripts

WHAT IS PHP

- **What is PHP?**
  - PHP files are returned to the browser as plain HTML
  - PHP file extension: ***.php**, **php3**, **php4**, **php5** or **phtml**
- **Why PHP?**
  - Allows easy storage and retrieval of information from supported databases
  - **Accessibility:** You can reach the internet from **any browser**, **any device**, **anytime**, **anywhere**

WHAT IS PHP

- **What is PHP?**
  - **Manageability:** It does not require distribution of application code and it is easy to change code.
  - **Security:** source code is not exposed. Once user is authenticated, can only allow certain actions. Also allows data encryption.
  - **Scalability:** Web-based 3-tier architecture can scale out

GETTING STARTED WITH PHP

- **Requirements**
  - A computer with web server (**Apache**, **IIS** etc), database server and PHP engine installed.
  - We can use web server as different flavors.
    - Previous traditions were, installing this different software independently.
    - This days, we install all this components as a single integrated environment
    - Different options: WAMP (for Windows), LAMP (for Linux), XAMP,

GETTING STARTED WITH PHP

- **Requirements**
  - **Text Editors:** Notepad, Notepad++, Sublime Text, Visual Studio Code, PHPStorm, CodeLobster etc
  - Web browser for displaying result. (IE, Chrome, Firefox, Opera, Safari etc)

PHP LANGUAGE FUNDAMENTALS

- **Basic Syntax:**
  - Opening and closing tags:
    - Canonical php tags: **<?php   …… ?>**
    - Short open tags (SGML-style): **<?   ……. ?>**
    - ASP-style tags: **<% ……%>**
    - HTML-script tags**:**   **<script language="php"> ……… </script>**
  - for maximum compatibility **<?php ….?>** is recommended.

PHP LANGUAGE FUNDAMENTALS

- **Output statement:**
  - Statements end with semi-colons
  - Two options to display output data on the browser. **echo** or **print**
  - Echo has no return value but print has.
  - Echo may take many parameters (evenif not mandatory), print can only take one argument.
  - Echo is faster than print

PHP LANGUAGE FUNDAMENTALS

- **Output statement:**
  - Echo and print general format:

```
echo output1, output2, output3 …;

echo (output statement);

print output statement;

print (output statement);
```

PHP LANGUAGE FUNDAMENTALS

- **Output statement:**
  - Example:
    - `echo 123; //output: 123`
    - `echo “Hello World!”; //` **`Hello world!`**
    - `echo (“Hello World!”); //` **`Hello world!`**
    - `echo “Hello”,”World!”; //` **`Hello World!`**
    - `echo Hello World!; //` **`error,`** `string should be enclosed in quotes`
    - `print (“Hello world!”); //` **`Hello world!`**

PHP LANGUAGE FUNDAMENTALS

- **Output statement:**
  - Its possible to embed HTML tag within output code.

    ```
    echo "<u> <i> Hello world!</i></u>";
    ```

  - Multiline printing:
    - Embed **<br/>** tag or use **<<<end** with print command.

PHP LANGUAGE FUNDAMENTALS

- **Comments**
  - Single-line comment:
    - # this is a comment or
    - // this is a comment → this is recommended
  - Multi-line comments
    - /* This is a multiline comment example */

PHP LANGUAGE FUNDAMENTALS

- **Variables**
  - All variables in PHP are denoted with a leading dollar sign **($)**
  - The value of the variable is its most recent assignment
  - Assigned with **=** operator
  - We can assign variables before declaring it
  - Do not have intrinsic data types
  - PHP can automatically convert data from one type into another when necessary

PHP LANGUAGE FUNDAMENTALS

- **Variable Naming rules**
  - Must start with a alphabet or _ (underscore) character
  - Can contain only characters (a-zA-Z0-9) and _ (underscore)
  - Can’t contain spaces
  - Variables are case sensitive
- **Declaration:**
  - [$variable_name=initial_value]

PHP LANGUAGE FUNDAMENTALS

- **Data types**
  - Total of 8 data types
    - Integer, double, Boolean, null, strings
    - Arrays, objects and resources
      - The 1st 5 are simple types and the last two (arrays and objects) are compound types

PHP LANGUAGE FUNDAMENTALS

- **Data types**
  - Integers:

```
$int_var = 12345;
$another_int = -12345 + 12345;
```

  - Doubles:

```
$pi= 3.14;
$version=1.12;
```

PHP LANGUAGE FUNDAMENTALS

- **Data types**
  - **Boolean:** two possible values; **true** or **false**
  - **NULL:** special type which has only one value. To give a variable a NULL value, assign it like:

    $my_var=NULL; // or null (it is not cases sensitive)
  - A variable with NULL value has the ff properties
    - Evaluates to false in a Boolean context
    - Returns false when tested with **isSet()** function

PHP LANGUAGE FUNDAMENTALS

- **Data types**
  - **Examples:**
  - **strings:** sequences of characters

```
$string_1 = "This is a string in double quotes";
$string_2 = "This is a somewhat longer, singly quoted
string";
$string_39 = "This string has thirty-nine characters";
$string_0 = ""; // a string with zero characters
```

PHP LANGUAGE FUNDAMENTALS

- **Data types**
  - **Single and double quotes**

```
<?php
$variable = "name";
$literally = 'My $variable will not print!\\n';
print($literally);
$literally = "My $variable will print!\\n";
print($literally);
?>
```

Output:

```
My $variable will not print!\n
My name will print
```

PHP LANGUAGE FUNDAMENTALS

- PHP provides a large number of **predefined variables** to all scripts.
  - **Superglobals** — Superglobals are built-in variables that are always available in all scopes
  - **$GLOBALS** — References all variables available in global scope
  - **$_SERVER** — Server and execution environment information
  - **$_GET** — HTTP GET variables

PHP LANGUAGE FUNDAMENTALS

- **$_POST** — HTTP POST variables
- **$_FILES** — HTTP File Upload variables
- **$_REQUEST** — HTTP Request variables, and can replace $_POST, $_GET and $_COOKIE variables

PHP LANGUAGE FUNDAMENTALS

- **$_SESSION** — Session variables
- **$_COOKIE** — HTTP Cookies
- **$php_errormsg** — The previous error message
- **$HTTP_RAW_POST_DATA** — Raw POST data

PHP LANGUAGE FUNDAMENTALS

- **$_SESSION** — Session variables
- **$_COOKIE** — HTTP Cookies
- **$http_response_header** — HTTP response headers
- **$argc** — The number of arguments passed to script
- **$argv** — Array of arguments passed to script

PHP LANGUAGE FUNDAMENTALS

- **Variable Scope:**
  - Local, and global variables
  - Functions and static variables
- Local variables:

```
<?
$x = 4;
function assignx () {
$x = 0;
print "\$x inside function is $x.
";
}
assignx();
print "\$x outside of function is
$x. ";
?>
```

Output:

```
$x inside function is 0.
$x outside of function is 4.
```

PHP LANGUAGE FUNDAMENTALS

- **Function parameters**

```
<?
// multiply a value by 10 and
return it to the caller
function multiply ($value) {
$value = $value * 10;
return $value;
}
$retval = multiply (10);
Print "Return value is $retval\n";
?>
```

Output:

```
Return value is 100
```

PHP LANGUAGE FUNDAMENTALS

- **Global variables:** can be accessed in any part of the program.

```
<?
$somevar = 15;
function addit() {
GLOBAL $somevar;
$somevar++;
print "Somevar is $somevar";
}
addit();
?>
```

Output:

```
Somevar is 16
```

PHP LANGUAGE FUNDAMENTALS

- **Static variables:**

```
<?
function keep_track() {
STATIC $count = 0;
$count++;
print $count;
print " ";
}
keep_track();
keep_track();
keep_track();
?>
```

Output:

```
1
2
3
```

PHP LANGUAGE FUNDAMENTALS

- **Constants:**
  - **Constant:** a variable whose value doesn’t change throughout the execution of the program.
  - Use define(**“const_name”,value**) to define a constant variable
  - No need to prefix it with **$** sign
  - To access its value, simply use name of the constant you created
  - You can also use function **constant()** to access its value

PHP LANGUAGE FUNDAMENTALS

- **Example:**

```
<?php
define("MINSIZE", 50);
echo MINSIZE;
echo constant("MINSIZE"); // same thing as the
previous line
?>
```

PHP LANGUAGE FUNDAMENTALS

- **Differences between constants and variables:**
  - No need to write $ sign in constants
  - Constants couldn’t be defined by simple assignment. We use **define()** function
  - Constants can be accessed anywhere in the program regardless of scoping rules
  - Once the constants have been set, couldn’t be redefined or undefined.

PHP LANGUAGE FUNDAMENTALS

- **Working with numbers:**
  - PHP treats numbers into two groups: integers and floating points
  - Doesn’t make you worry about the differences between the two
  - Can automatically convert from one into another type
  - 1.5 is not 1 but its 1.5 unlike other programming languages
  - In PHP 1+”1” is 2.

PHP LANGUAGE FUNDAMENTALS

- **String functions:**
  - **is_numeric()** → checking the given value is numeric value

```
if (is_numeric('five')) { /* false */ }
if (is_numeric(5))      { /* true  */ }
if (is_numeric('5'))    { /* true  */ }
```

  - Rounding floating point numbers: **round(), ceil() and floor()**

```
$number = round(2.4);   // $number = 2
$number = ceil(2.4);    // $number = 3
$number = floor(2.4);   // $number = 2
```

PHP LANGUAGE FUNDAMENTALS

- **Operating on a series of integers:**
  - **range():** returns an array populated with integers

```
foreach(range($start,$end) as $i) {
    echo “$i<br>”;
}
```

  - Is similar to a for loop:

```
for ($i = $start; $i <= $end; $i += $increment) {
    echo “$i<br>”;  }
```

PHP LANGUAGE FUNDAMENTALS

- **Generating random numbers within a range:**
  - use **mt_rand();** function

```
// random number between $upper and $lower,
inclusive
$random_number = mt_rand($lower, $upper);
```

  - Calculating exponents:

```
$exp = exp(2);          // 7.3890560989307
$exp = pow( 2, M_E);    // 6.5808859910179
$pow = pow( 2, 10);     // 1024
```

PHP LANGUAGE FUNDAMENTALS

- **Formatting numbers:**
  - Use the **number_format();** function to format a number as integer

```
$number = 1234.56;
print number_format($number);    // 1,235 because
number is rounded up
```

  - Specify a number of decimal places to format as a decimal:

```
print number_format($number, 2); // 1,234.56
```

Thank You!