UNIT-6

Server Side Scripting using PHP

PHP (Hypertext Preprocessor) is a server scripting language, and a powerful tool for making dynamic and interactive web pages. PHP is widely-used, free, and efficient alternative to competitors such as Microsoft's ASP. PHP scripts are executed on the server and is free to download and use.

PHP file:

→ PHP files can contain text, HTML, CSS, JavaScript, and PHP code.
→ PHP code are 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 encrypt data and many more.

Why PHP?

→ PHP runs on various platforms (Windows, Linux, Unix, etc.).
→ PHP is compatible with almost all servers used today.
→ PHP supports wide range of databases.
→ PHP is free.
→ PHP is easy to learn.

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.
?>
```

we get output with the help of echo and print both are same

PHP can support multiple start and end tags. Each start tag must be closed by it's own end tags.

Comments in PHP:

or // → specifies single line comment.
/*....*/ → specifies multi line comment.

PHP Example to print Hello World!

```
<!DOCTYPE html>
<html>
    <body>
        <h1>My first PHP page</h1>
        <?php
            echo "Hello World!"; OR print="Hello World!";
        ?>
    </body>
</html>
```

✱ Variables:-

In PHP, a variable starts with the $ sign, followed by the name of the variable. Variables are like containers for storing data.

Examples:
```
$txt = "Hello World!";
$x = 5;
$y = 10.5;
```

Rules for PHP variables: (Similar rules as for other languages):

→ A variable starts with $ 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.
→ Variable names are case-sensitive.

✱. PHP data types:-

PHP supports following data types:-

i) String → A string is a sequence of characters. A string can be any text inside quotes. We can use single or double quotes.
Example:- `$x = "Hello World!";`

ii) Integer → An integer data type is a non-decimal number.
Example: `$x = 5985;`

iii) Float → A float is a number with a decimal point.
Example: `$x = 10.365;`

iv) Boolean → A boolean represents two possible states: true or false.
Example:
```
$x = true;
$y = false;
```

v) Array → An array stores multiple values in one single variable.
Example: `$cars = array ("Volvo", "BMW", "Toyota");`

vi) **Null** → Null is a special data type which can have only one value NULL. It specifies that no value is assigned to the variable.

Example: `$x = null;`

⊛ **PHP Strings:** A string is a sequence of characters, like "Hello World!". A string can be any text inside quotes. We can use single or double quotes. The single quotes aren't part of string. They are delimeters, which tell the PHP engine where to start and end of the string is. If we want to include a single quote in our string, we put a backslash (\) before the single quote inside the string.

Example: Defining strings:

```
<?php
    echo('Hello World!');
?>
```

Output: Hello World!

Example: Defining strings with backslash:

```
<?php
    echo('We\'ll do it.');
?>
```

Output: We'll do it.

PHP string functions to manipulate strings:

i) **strlen()** → The PHP strlen() function returns the length of a string.

Example:
```
<?php
    echo strlen("Hello World");  //outputs 11.
?>
```

ii) **str_word_count()** → The PHP str_word_count() function counts the number of words in a string.

Example:
```
<?php
    echo str_word_count("Hello World!"); //outputs 2
?>
```

iii) **strrev()** → The PHP strrev() function reverses a string:

Example:
```
<?php
    echo strrev("Hello World!");//outputs !dlroW olleH
```

iv) **strpos()** → The strpos() function searches for a specific text within string.

Example:
```
<?php
    echo strpos("Hello World", "World"); //outputs 6.
?>
```

This example searches text World in the string "Hello World".

v) **str_replace()** → The PHP str_replace() replaces some characters with some other characters in a string.

Example:
```
<?php
    echo str_replace("World", "Dolly", "Hello World!");
?>
```
(to be replaced → by which to replace → present string)

//outputs Hello Dolly!

⑧. PHP Constants:-

Constants are like variables except that once they are defined they cannot be changed or undefined. It is an identifier (name) for a simple value. Unlike variables, constants are automatically global across the entire script. We can create a constant, using the define() function in PHP as in the syntax below:

Syntax: `define(name, value, case-insensitive)`

Parameters:

name → specifies the name of the constant.
value → specifies the value of the constant.
case-insensitive → specifies whether the constant name should be case-insensitive or not. Default is false.

Example1:- The example below creates a constant with a case-sensitive name:

```
<?php
    define("GREETING1", "Welcome to NoteJunction");
    echo GREETING1;
?>
```

Example2:- The example below creates a constant with a case-insensitive name:

```
<?php
    define("GREETING1", "Welcome to NoteJunction", true);
    echo GREETING1;
?>
```

PHP Constant with const keyword:-

The const keyword defines constants at compile time. It is faster than define() and is always case sensitive.

Example:

```
<?php
    const Message = "Hello const";
    echo Message;
?>
```

⑨. PHP Operators:-

PHP uses several operators like arithmetic, comparision, bitwise, logical, Incrementing/Decrementing, Assignment, Error Control, String, Array, Type, and Execution operators.

We can also categorize operators on the basis of operands in 3 forms:-

Unary operators → works on single operands such as +, −, *, / etc.

Binary operators → works on two operands such as ++, -- etc.

Ternary operators → works on condition such as (?:).

Syntax: condition? if true: if false.

Lesser imp compared to other and exam point of view

almost all similar to JS control statements & switch similar loops.

⊛. Control Structure:-

1) PHP If Else:- PHP if else statement is used to test condition. There are various ways to use if statement in PHP.

i) PHP if statement:

Syntax: if (condition){
    //code to be executed.
}

ii) PHP if-else statement:

Syntax: if (condition){
    //code to be executed if true
}else{
    //code to be executed if false
}

2) PHP switch:

Syntax: switch (expression){
    case value1:
        //code to be executed
        break;
    case value2:
        //code to be executed
        break;
    :
    default:
        //code to be executed if all cases are not matched.
}

3) PHP Loops:

i) For loop: Syntax: for(initialization; condition; increment/decrement){
    //code to be executed
}

Example:
```
<?php
    for($n=1; $n<=10; $n++){
        echo "$n <br/>";
    }
?>
```

ii) For Each Loop: Used to traverse array elements.

Syntax: foreach ($array_name as $key => $data)

iii) While loop: It is used to traverse set of code like for loop and if number of iterations is not known.

Syntax:
```
while (condition) {
    //code to be executed
}
```

iv) do while loop: Similar to while loop but guaranteed to run at least once because condition is checked after executing code.

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

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

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

Example:
```
<?php
function writeMsg(){
    echo ("Hello World!");
}
writeMsg(); //function call
?>
```

PHP Function Arguments:

Arguments are specified after the function name, inside the parentheses. We can add as many arguments as we need, just seperating them with a comma. Information can be passed to functions through arguments. An argument is just like a variable.

Example:
```php
<?php
    function familyName($fname){
        echo "$fname is my family member.<br>";
    }
    familyName("Mukesh");
    familyName("Harish");
?>
```

If we provide here two arguments then we should also provide two values while calling and so on.

 prints next value in next line.

PHP default Argument value:-

Example:
```php
<?php
    function setHeight($minHeight = 50){
        echo "The minimum height is: $minHeight <br>";
    }
    setHeight(350);
    setHeight(80);
    setHeight(); //will use the default value of 50.
```

यसमा argument मा default value दिएर call गर्दा value दिए पहिलेको value print गर्ने अन्यथा 50 default value नाउ लिएर

Returning values from functions:-

To let a function return a value, we use return statement:

Example:-
```php
<?php
    function sum($x, $y){
        $z = $x + $y;
        return $z;
    }
    echo "5+10= ".sum(5,10)."<br>";
    echo "2+4= ".sum(2,4)."<br>";
?>
```

dot is concatenation operator here

⊛. PHP Array:

An array is a special variable, which can hold more than one value at a time. An array can hold many values under a single name, and we can access the values by referring to an index number. In PHP, the array() function is used to create an array. In PHP, there are three types of arrays:

Indexed arrays → Arrays with a numeric index.
Associative arrays → Arrays with named keys.
Multidimensional arrays → Arrays containing one or more arrays.

Creating an array:-

To create an array we use array() language construct.

Example:
```
<?php
    $fruits=array("Apple","Banana");
?>
```

A shortcut for the array() language construct is a pair of square brackets. Example:
```
<?php
    $fruits=["Apple","Banana");
?>
```

1) PHP indexed arrays:-

There are two ways to create indexed arrays: one the index can be assigned automatically:

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

the second by assigning index manually:

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

Example:
```
<?php
    $cars=array("Volvo","BMW","Toyota");
    $arrlength=count($cars);
    for($x=0;$x<$arrlength;$x++){
        echo $cars[$x]."<br>";
    }
?>
```

The count() is used to return the length of an array

2) PHP associative arrays:-

Associative arrays are arrays that use named keys that we 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";
```

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

Output: Peter is 35 years old.

escape if felt hard if is lesser imp

7.1.1) Multidimensional Arrays:-

A multidimensional array is an array containing one or more arrays. Multidimensional arrays stores an another array at each index instead of single element.

Syntax:
```
$array-name = array(array());
```

Example:- Creating multidimensional arrays with array(), and [ ].

```
<?php
$meals = array( 'breakfast' => [ 'Walnut Bun', 'Coffee' ],
                'lunch' => ['Cashew Nuts', 'White Mushrooms'],
                'snack' => ['Dried Mulberries', 'Noodles']
              );
print_r($meals);
echo '<br>';

$lunches = [ ['Chicken', 'Egg', 'Rice'],
             ['Beef', 'Noodles'] ];
print_r($lunches);
echo '<br>';

$flavours = array('Japanese' => array('hot' => 'noodles',
                                      'salty' => 'sauce'),
                  'Chinese' => array('hot' => 'mustard',
                                     'sweet' => 'sauce'));
print_r ($flavours);
echo '<br>';
?>
```

print_r helps to print mixed expressions like multidimensional arrays.

⊛ Creating Class and Objects in PHP:

Like C++ and Java, PHP also supports object oriented programming.

Creating Class:

A class is a "blueprint" for an object, is a code template, used to generate objects. It contains the code for properties and methods. A class is defined by using the class keyword, followed by the name of the class and pair of curly braces { }. All it's properties and methods go inside the braces.

Syntax:
```
<?php
    class fruit {
        //code goes here properties and methods.
    }
?>
```

Creating Objects:

An object is the copied and active form of a class. Classes are nothing without objects. We can create multiple objects from a class. Each object has all the properties and methods defined in the class, but they will have different property values. Objects of a class is created using the new keyword.

Example:- Following is an example of how to create object using new operator.

```
<?php
    class Books{
        //code goes here properties and methods
    }
    // Creating three objects of Books
    $physics = new Books;
    $maths = new Books;
    $chemistry = new Books;
?>
```

⊛. PHP Forms:

Forms are essential components of a website used to register a new account, sign in or login to the account. Using a form in a PHP program is a two-step activity. Step one is to display the form, which involves constructing HTML tags and UI's. Step two is processing the submitted for information submitted by the user.

Accessing Form Elements:

The PHP superglobals $_GET and $_POST are used to collect form-data. $_GET is an array of variables passed to the current script via the URL parameters. $_POST is an array of variables passed to the current script via the HTTP POST method.

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 information to send. The limitation is about 2000 characters. GET may be used for sending non-sensitive data. 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. POST may be used for sending sensitive data. Developers prefer POST for sending form data.

Example:- Form processing in PHP with $_SERVER.

Name of php file we created for form processing

```
<form name="loginform" method="POST" action="./formSubmit.php">
    <div>
        Username: <input type="text" name="username" id="username">
        Password: <input type="password" name="password" id="password">
    </div>
    <div>
        <input type="submit" name="login" value="login">
    </div>
</form>
```

this php file name will be as formSubmit.php

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST"){
    $username = $_POST['username'];
    $password = $_POST['password'];
    print $_POST['username'] ." and ". $_POST['password'];
}
```

Useful Server Variables: [less imp] can be escaped

1) QUERY_STRING → The part of URL after the question mark where the URL parameters live.

2) PATH_INFO → Extra path information tacked onto the end of URL after a slash.

3) SERVER_NAME → The name of website on which the PHP engine is running.

4) DOCUMENT_ROOT → The directory on the web server computer that holds the documents available on the website.

5) HTTP_HOST → The host's name as found in the http header.

6) HTTP_USER_AGENT → The user agents (browser) detail accessing web page.

7) PHP_SELF → The file-name of currently executing string.

8) REMOTE_ADDR → The IP address of remote machine accessing the current page.

less imp exam point of view

Form Validation: [Server Side Validation]

1) Text Validation:- For validating text preg_match() function is used which ~~sera~~ searches string for pattern, returns true if pattern exists, otherwise returns false.

Example:-
```
<?php
    $name = $_POST['name'];
    if (!preg_match ("/^[A-Z a-z]*$/", $name)) {
        echo "Only letters and white space allowed";
    }
    else {
        echo "The name is". $name;
    }
?>
```

2) Email Validation:- For validating email we use filter_var() function and also we use FILTER_VALIDATE_EMAIL as shown in the example below:-

Example: <?php

```
$email = $_POST['email'];
if (filter_var ($email, FILTER_VALIDATE_EMAIL)){
    echo("$email is valid");
}
else{
    echo("$email is invalid");
}
?>
```

iii) Number Validation:- For validating number/integer, we use FILTER_VALIDATE_INT and FILTER_VALIDATE_FLOAT to validate float.

⑧ Cookies and Sessions:- [Imp]

1) Cookie: A cookie is variable that is stored on the visitors computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, we can both create and retrieve cookie values.

Similar to JS cookies only some small changes in syntax

Creating Cookies with PHP:

A cookie is created with the setcookie() function.

Syntax: setcookie (name, value, expire, path, domain, ..);

Only name parameter is required. All other parameters are optional.

To delete a cookie, we use the setcookie() function with an expiration date in the past.

This example can also be called as PHP script to illustrate handling of cookie

Example :- Creating and retriving a cookie:

86400 equals to 1 day multiplied by 30 means 1 month

path parameter

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie ($cookie_name, $cookie_value, time() + (86400 * 30), "/");

if(!isset ($_COOKIE[$cookie_name])) {
  echo $cookie_name . "is not set.";
}
else{
    echo $cookie_name . "is set.";
}
?>
```

2) Session:

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 rather session is stored in server. Session variables hold information about one single user, and are available to all pages in one application.

Why do we need session?

When we work with an application, we open it, do some changes, and then we close it. This is much like session. The computer knows who we are. It knows when we start the application and when we end. But on the internet there is one problem: the web server does not know who we are or what we do, because HTTP address doesn't maintain state. Session variables solve this problem by storing user information to be used across multiple pages. By, default, session variables last until the user closes the browser.

Uses of Session:-

(less imp)

→ Session can help to uniquely identify each client from another.
→ Session is secure and transparent from the user.
→ It is easy to implement and we can store any kind of object.
→ It helps to maintain user state and data all over the application.
→ It stores client data seperately.

Starting PHP Session:-

A PHP session is easily started by making a call to the `session_start()` function. Session variables are stored in associative array called `$_SESSION[]`. We make use of `isset()` function to check if session variable is already set or not.

Example:- Starting session and setting session values:-

```
<?php
    //start the session
      session_start();
    // Set session variables
    $_SESSION["username"] = "john";
    $_SESSION["email"] = "ducky@gmail.com";
    echo "Session variables are set.";
?>
```

Note: To get session variables we just print session variables with echo. for e.g. `echo $_SESSION["username"]` will help to get username. To remove all global variables and destroy session, use `session_unset()` and `session_destroy()`.

⊛. Working with PHP and MySQL : [Imp]

There are three ways of working with MySQL and PHP.

1. MySQLi (object-oriented)
2. MySQLi (procedural)
3. PDO.

We will mostly do with procedural approach in this section.

Creating Connection using procedural approach:

Example:

```
<?php
    $servername = "localhost";
    $username = "root";
    $password = "";

//Creating connection.
    $conn = mysqli_connect ($servername, $username, $password);

//Checking connection.
    if (!$conn) {
        die ("Connection failed: " . mysqli_connect_error());
    }
    echo "Connected successfully";
?>
```

These three lines are basic we will always write these lines as they are while using localhost

empty string since we are using localhost no need of password. Even space is not allowed it will throw an error

required attributes for connection

This is inbuilt function of PHP that will show the type of error occured

die will stop program after this line executes so no need of else.

Note:- Creating connection using object-oriented will be same as code of procedural we just need to change `mysqli_connect` by new `mysqli`. All other things will remain same.

Creating Database using procedural approach:-

Example:- For creating database we will just add below code at the last (i.e, after checking connection) in the example above.

```
// Create database.
$sql = "CREATE DATABASE webTech";
if (mysqli_query($conn, $sql)){
    echo "Database created successfully";
}else{
    echo "Error creating database:" . mysqli_error ($conn);
}
```

as we know sql query for creating database has syntax: CREATE DATABASE database_Name so webTech is db name.

Note:- Now we are using database so after top three basic lines we also add `$dbname="webTech";`

Inserting Multiple Data:-

Example:

```
<?php
    $servername ="localhost";
    $username ="username";
    $password =" ";
    $dbname ="web";
```

our database name on which we are going to work

```
    //Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
    //Check connection
    if (!$conn){
        die("Connection failed:" .mysqli_connect_error());
    }
    //Inserting multiple data
    $sql="INSERT INTO student (firstname, lastname, email)
        VALUES ('John', 'Doe', 'john@example.com');";
$sql .="INSERT INTO student (firstname, lastname, email)
        VALUES ('Mary', 'Moe', 'mary@example.com');";
if(mysqli_multi_query($conn, $sql)){
    echo "Records inserted successfully";
}else{
    echo "Error". $sql. "<br".mysqli_error($conn);
}
mysqli_close($conn);
?>
```

Note:- For selecting, deleting, updating records in table also the code is same as above just sql query is changed according to what we are going to do. Sql queries for selecting, deleting, updating we have already read in database subject SQL chapter.

⊛. Working with PHP and MySQL : [Imp]

There are three ways of working with MySQL and PHP.

1. MySQLi (object-oriented)
2. MySQLi (procedural)
3. PDO.

We will mostly do with procedural approach in this section.

Creating Connection using procedural approach:

Example:

```
<?php
    $servername = "localhost";
    $username = "root";
    $password = "";

//Creating connection.
    $conn = mysqli_connect ($servername, $username, $password);

//Checking connection.
    if (!$conn) {
        die ("Connection failed: " . mysqli_connect_error());
    }
    echo "Connected successfully";
?>
```

These three lines are basic we will always write these lines as they are while using localhost

empty string since we are using localhost no need of password. Even space is not allowed it will throw an error

required attributes for connection

This is inbuilt function of PHP that will show the type of error occured

die will stop program after this line executes so no need of else.

Note:- Creating connection using object-oriented will be same as code of procedural we just need to change `mysqli_connect` by `new mysqli`. All other things will remain same.

Creating Database using procedural approach:-

Example:- For creating database we will just add below code at the last (i.e, after checking connection) in the example above.

```
// Create database.
$sql = "CREATE DATABASE webTech";
if (mysqli_query($conn, $sql)) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . mysqli_error ($conn);
}
```

as we know sql query for creating database has syntax: CREATE DATABASE database_Name so webTech is db name.

Note:- Now we are using database so after top three basic lines we also add `$dbname = "webTech"`

⊛. Introduction to PHP Frameworks:-

(lesser imp exam point of view)

1) Codeigniter: CodeIgniter is a powerful PHP framework with a very small footprint, built for developers who need a simple and elegant toolkit to create full-featured web applications.

Installation Instructions:

→ Unzip the package.
→ Upload the CodeIgniter folders and files to server. Normally index.php file will be at root.
→ Open the application/config/config.php file with a text editor and set base URL.
→ If database is being used, open the application/config/database.php file with a text editor and set database settings.

2) Laravel: Laravel is a web application framework with expressive, elegant syntax. Laravel attempts to take the pain out of development by easing common tasks used in the majority of web projects such as aunthentication, routing, sessions and caching. Laravel is a powerful MVC PHP framework.

Installation: Laravel utilizes Composer to manage dependicies. So, before using Laravel, we make sure that we have Composer installed on our machine. Laravel can be installed by various methods like via laravel installer, via Composer Create-Project, via local development server etc. Via Composer Create-Project, Laravel can be installed issuing following command on our terminal:

```
composer create-project --prefer-dist laravel/laravel blog
```

3) Introduction to Wordpress:-

Wordpress is a free and open source content management system (CMS) based on PHP and MySQL. It is the most widely used CMS software in the world. WordPress is an industry leading website and blog building tool. There is an entire community of WordPress users and developers who are able to assist one another. Users have countless options for free and paid plugins and site templates for their use in creating their site.

⊛ Long Questions from this chapter will be mainly asked like this:-

Q1. Construct a web page that takes name, address and phone number from the user and stores this information in the database using server side script. Use Javascript to validate form data for phone number.

Q2. Prepare a form that should contain text box, selection list and radio button. Write PHP script to store data from the form into database using database connection and appropriate query.

⇒ Solution of these questions can be found on collegenote website in old questions and solutions section.

If my notes really helped you,then you can support me on esewa for my hardwork.

Esewa ID: 9806470952