PHP

What is PHP

- PHP is a server side scripting language. that is used to develop Static websites or Dynamic websites or Web applications. PHP stands for Hypertext Pre-processor, that earlier stood for Personal Home Pages.
- PHP scripts can only be interpreted on a server that has PHP installed.
- The client computers accessing the PHP scripts require a web browser only.
- A PHP file contains PHP tags and ends with the extension ".php".

What is a Scripting Language

- A script is a set of programming instructions that is interpreted at runtime.
- The purpose of the scripts is usually to enhance the performance or perform routine tasks for an application.
- Server side scripts are interpreted on the server while client side scripts are
- interpreted by the client application.
- PHP is a server side script that is interpreted on the server while JavaScript is an example of a client side script that is interpreted by the client browser. Both PHP and JavaScript can be embedded into HTML pages.

Programming language	Scripting language
Has all the features needed to develop complete applications.	Mostly used for routine tasks
The code has to be compiled before itcan be executed	The code is usually executed without compiling
Does not need to be embedded into other languages	Is usually embedded into other software environments.

Difference between programming language and scription language

Why use PHP

- It handles dynamic content, database as well as session tracking for the website.
- You can create sessions in PHP.
- It can access cookies variable and also set cookies.
- It helps to encrypt the data and apply validation.
- PHP supports several protocols such as HTTP, POP3, SNMP, LDAP, IMAP, and many more.

- Using PHP language, you can control the user to access some pages of your website.
- As PHP is easy to install and set up, this is the main reason why PHP is the best language
- to learn.
- PHP can handle the forms, such as - collect the data from users using forms, save it into
- the database, and return useful information to the user. For example - Registration form.

Why Learn PHP

- Easy to Learn
- Free of Cost
- Flexible
- Supports nearly all databases
- Secured
- Huge Community Support

Features of PHP

- Open-Source and Free
- PHP Server-Side Scripting
- Interpreted language
- Database connectivity
- Object-oriented programming (OOP)
- Built-in functions

- Session management
- Security feature

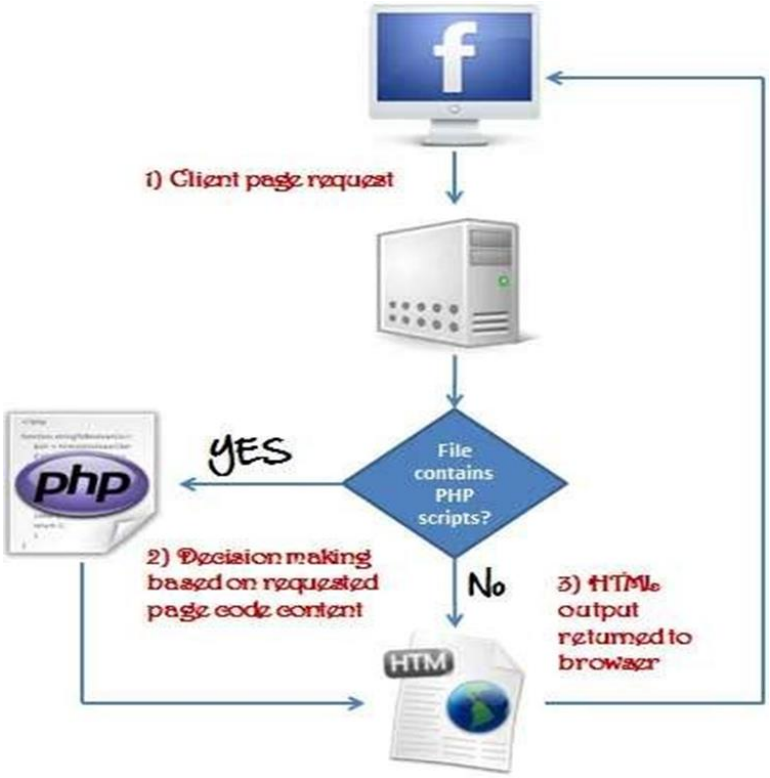
History of PHP

- PHP is developed by Rasmus Lerdorf in 1994 the very first version of PHP that
- simply designed to set the Common Gateway Interface (CGI) binaries.
- The latest version of PHP is PHP version 8 which is released on November 24, 2022.
- It can be easily embedded with HTML files.
- HTML codes can also be written in a PHP file. The PHP codes are executed on the server side whereas HTML codes are directly executed on the browser.

Applications of PHP

- Server-side web development
- Content management systems
- E-commerce websites
- Database-driven applications
- Web APIs

Architecture of a PHP web application



What is XAMPP

XAMPP is an open-source, cross-platform web server that consists of a web server, MySQL database engine, and PHP and Perl programming packages. It is compiled and maintained by Apache. It allows users to create WordPress websites

online using a local web server on their computer. It supports Windows, Linux, and Mac.

It is compiled and maintained by apache. The acronym XAMPP stands for;

X – [cross platform operating systems] meaning it can run on any OS Mac OX , Windows , Linux etc.

A – Apache – this is the web server software.

M – MySQL – Database.

P – PHP

P – Perl – scripting language

PHP | Basic Syntax

Escaping To PHP:

Writing the PHP code inside **<?php ….?>** is called Escaping to PHP.

The script starts with **<?php and ends with ?>.** These tags are also called 'Canonical PHP tags'

```
<?php
echo "hello world!";
?>
```

Data Types

- Integers − Whole numbers, without a decimal point, like 4195.
- Doubles − Floating-point numbers like 3.14159 or 49.1.

- Booleans − Have only two possible values, either true or false.
- NULL − Special type that only has one value: NULL.
- Strings − Sequences of characters, like 'PHP supports string operations.'
- Arrays − Named and indexed collections of other values.
- Objects − Instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- Resources − Special variables that hold references to resources external to PHP (such as database connections).

PHP is a loosely typed language; it does not have explicit defined data types

PHP Variable

- A variable is a name given to a memory location that stores data at runtime.
- The scope of a variable determines its visibility.
- A Php global variable is accessible to all the scripts in an application.
- A local variable is only accessible to the script that it was defined in.
- All variable names must start with the dollar sign e.g.
- Variable names are case sensitive; this means $my_var is different from $MY_VAR
- All variables names must start with a letter follow other characters e.g. $my_var1. $1my_var is not a legal variable name.
- Variable names must not contain any spaces, “$first name” is not a legal variable name. You can instead use an underscore in place of the space e.g. $first_name. You cant use characters such as the dollar or minus sign to separate variable names.

```
<?php
$my_var = 1;
echo $my_var;
?>
```

Constants

- Constants can be defined using the const keyword or define() function.
- Constants do not have $ in front of them like variables have.
- Constants can be accessed from anywhere without regard to variable scoping rules.

Comments in PHP

Comments help in reminding the developer about the code if it's re-visited after a period of time.

- Single Line Comment (//) or (#)
- Multi-line or Multiple line Comment (/*…*/)

Echo/Print

The echo statement outputs one or more expressions, with no additional newlines or spaces.

Echo "pen"," ","pencil"," ","eraser";

The print statement is similar to echo, but it outputs an expression

Print "pen"

echo statement	print statement
echo accepts a list of arguments (multiple arguments can be passed), separated by commas.	print accepts only one argument at a time.
It returns no value or returns void.	It returns the value 1.
It displays the outputs of one or more strings separated by commas.	The print outputs only the strings.
It is comparatively faster than the print statement.	It is slower than an echo statement.

Type Casting

The term "Type Casting" refers to conversion of one type of data to another.

```
<?php
$a = 9.99;
$b = (int)$a; var_dump($b);
?>
```

PHP Operators

PHP Operator is a symbol i.e used to perform operations on operands. In simple words,

operators are used to perform operations on variables or values

$num=10+20; //+ is the operator and 10,20 are operands

$a=10;

$b=20;

Snum=$a+$b;

Arithmetic Operators

operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division

%	Modulus
++	Increment
––	Decrement

Program

```
<?php
   $var = 1;       // Assign the integer 1 to $var
   echo $var++;    // Print 1, $var is now equal to 2
   echo ++$var;    // Print 3, $var is now equal to 3
   echo --$var;    // Print 2, $var is now equal to 2
   echo $var--;    // Print 2, $var is now equal to 1
?>
```

Assignment operators

PHP assignment operators applied to assign the result of an expression to a variable. = is a fundamental assignment operator in PHP. It means that the left operand gets set to the value of the assignment expression on the right.

Operator	Description

=	Assign
+=	Increments then assigns
-=	Decrements then assigns
*=	Multiplies then assigns
/=	Divides then assigns
%=	Modulus then assigns

Program

```
<?php
    $x = 10;
    echo $x; // Outputs: 10
    $x = 20;
    $x += 30;
    echo $x; // Outputs: 50
    $x = 50;
    $x -= 20;
    echo $x; // Outputs: 30
    $x = 5;
    $x *= 25;
    echo $x; // Outputs: 125
```

```
$x = 50;

$x /= 10;

echo $x; // Outputs: 5

$x = 100;

$x %= 15;

echo $x; // Outputs: 10
?>
```

Comparison Operators

Use of PHP Comparison Operators is comparing two values (number or string).

```
<?php
    $x = 25;
    $y = 35;
    $z = "25";
    var_dump($x == $z);  // Outputs: boolean true
    var_dump($x === $z); // Outputs: boolean false
    var_dump($x != $y);  // Outputs: boolean true
    var_dump($x !== $z); // Outputs: boolean true
    var_dump($x < $y);   // Outputs: boolean true
    var_dump($x > $y);   // Outputs: boolean false
    var_dump($x <= $y);  // Outputs: boolean true
```

```
var_dump($x >= $y);  // Outputs: boolean false
?>
```

Incrementing and Decrementing Operators

Operator	Name	Effect
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

Program

```
<?php
    $x = 10;
    echo ++$x; // Outputs: 11
    echo $x;   // Outputs: 11
    $x = 10;
    echo $x++; // Outputs: 10
    echo $x;   // Outputs: 11

    $x = 10;
```

```
echo --$x; // Outputs: 9
echo $x;   // Outputs: 9
$x = 10;
echo $x--; // Outputs: 10
echo $x;   // Outputs: 9
?>
```

Logical operators

Operator	Description
&&	And operator performs logical conjunction on two expressions (if both expressions evaluate to True, result is True. If either expression evaluates to False, the result is False)
||	Or operator performs a logical disjunction on two expressions (if either or both expressions evaluate to True, the result is True).
!	Not operator performs logical negation on an expression.

Program

```
<?php
        $year = 2014;
        // Leap years are divisible by 400 or by 4 but not 100
        if(($year % 400 == 0) || (($year % 100 != 0) && ($year % 4 == 0))){
           echo "$year is a leap year.";
        } else{
           echo "$year is not a leap year.";
        }
?>
```

Ternary operators

(Conditional statement) ? (Statement_1) : (Statement_2);

```
        <?php
        $result = 62;
        echo ($result >= 40) ? "Passed" : " Failed";
        ?>
```

Concatenation operator

Operator	Description
.	The PHP concatenation operator (.) is used to combine two string values to create one string.

.= Concatenation assignment.

Conditional Statements in PHP

- if else statement
- elseif statement
- switch statement

if statement

syntax

```
if(condition){
   // Code to be executed
}
```

Example

```
<?php
$d = date("D");
if($d == "Fri"){
   echo "Have a nice weekend!";
}
?>
```

The if...else Statement

Syntax

```
if(condition){
```

```
    // Code to be executed if condition is true
} else{
    // Code to be executed if condition is false
}
```

Example

```
<?php
$d = date("D");
if($d == "Fri"){
    echo "Have a nice weekend!";
} else{
    echo "Have a nice day!";
}
?>
```

The if...elseif...else Statement

Syntax

```
if(condition1){
    // Code to be executed if condition1 is true
} elseif(condition2){
    // Code to be executed if the condition1 is false and condition2 is true
} else{
    // Code to be executed if both condition1 and condition2 are false
}
```

Example

```
<?php
```

```
$d = date("D");
if($d == "Fri"){
   echo "Have a nice weekend!";
} else if($d == "Sun"){
   echo "Have a nice Sunday!";
} else{
   echo "Have a nice day!";
}
?>
```

PHP Switch…Case Statements

Syntax

```
switch(n){
   case label1:
      // Code to be executed if n=label1
      break;
   case label2:
      // Code to be executed if n=label2
      break;
   ...
   default:
      // Code to be executed if n is different from all labels
}
```

Example

```
<?php
```

```
$today = date("D");
switch($today){
    case "Mon":
        echo "Today is Monday. Clean your house.";
        break;
    case "Tue":
        echo "Today is Tuesday. Buy some food.";
        break;
    case "Wed":
        echo "Today is Wednesday. Visit a doctor.";
        break;
    case "Thu":
        echo "Today is Thursday. Repair your car.";
        break;
    case "Fri":
        echo "Today is Friday. Party tonight.";
        break;
    case "Sat":
        echo "Today is Saturday. Its movie time.";
        break;
    case "Sun":
        echo "Today is Sunday. Do some rest.";
        break;
    default:
```

```
        echo "No information available for that day.";
        break;
    }
    ?>
```

PHP Loops

while — loops through a block of code as long as the condition specified evaluates to true.

do…while — the block of code executed once and then condition is evaluated. If the condition is true the statement is repeated as long as the specified condition is true.

for — loops through a block of code until the counter reaches a specified number.

foreach — loops through a block of code for each element in an array.

PHP while Loop

Syntax

```
    while(condition){
       // Code to be executed
    }
```

Example

```
    <?php
    $i = 1;
    while($i <= 3){
```

```
    $i++;
    echo "The number is " . $i . "<br>";
}
?>
```

PHP do…while Loop

do{

// Code to be executed

}while(condition);

Example

```
<?php
$i = 1;
do{
    $i++;
    echo "The number is " . $i . "<br>";
}
while($i <= 3);
?>
```

PHP for Loop

Syntax

```
for(initialization; condition; increment){
    // Code to be executed
}
```

Example

```
<?php
for($i=1; $i<=3; $i++){
   echo "The number is " . $i . "<br>";
}
?>
```

PHP foreach Loop

Syntax

```
foreach($array as $value){
   // Code to be executed }
```

Example

```
<?php
$colors = array("Red", "Green", "Blue");
// Loop through colors array
foreach($colors as $value){
   echo $value . "<br>";
}
?>
```

PHP Arrays

Types of Arrays

Indexed array — An array with a numeric key.

Associative array — An array where each key has its own specific value.

Multidimensional array — An array containing one or more arrays within itself.

Indexed Arrays

An indexed or numeric array stores each array element with a numeric index

Example

```
<?php
// Define an indexed array
$colors = array("Red", "Green", "Blue");
?>
```

Associative Arrays

In an associative array, the keys assigned to values can be arbitrary and user defined strings.

Example

```
<?php
// Define an associative array
$ages = array("Peter"=>22, "Clark"=>32, "John"=>28);
?>
```

Multidimensional Arrays

Example

```
<?php
// Define a multidimensional array
$contacts = array(
    array(
        "name" => "Peter Parker",
        "email" => "peterparker@mail.com",
```

```
    ),
    array(
        "name" => "Clark Kent",
        "email" => "clarkkent@mail.com",
    ),
    array(
        "name" => "Harry Potter",
        "email" => "harrypotter@mail.com",
    )
);
// Access nested value
echo "Peter Parker's Email-id is: " . $contacts[0]["email"];
?>
```

PHP | Functions

The function is a self-contained block of statements that can repeatedly be executed whenever we need it.

There are two types of functions

- ❖ Internal (built-in) Functions
- ❖ User Defined Functions

Syntax

```
function function_name(){
    executable code;
}
```

Example

```
<?php
function state()
{
   echo "Tamilnadu";
}
// Calling the function
state();
?>
```

```
function function_name($first_parameter, $second_parameter) {
   executable code;
}
<?php
// function along with three parameters
function pro($num1, $num2, $num3)
{
   $product = $num1 * $num2 * $num3;
   echo "The product is $product";
}
// Calling the function
// Passing three arguments
pro(2, 3, 5);
```

?>

PHP GET and POST

A web browser communicates with the server typically using one of the two HTTP (Hypertext Transfer Protocol) methods — GET and POST. Both methods pass the information differently and have different advantages and disadvantages, as described below.

The GET Method

In GET method the data is sent as URL parameters that are usually strings of name and value pairs separated by ampersands (&).

http://www.example.com/action.php?name=john&age=24

Advantages and Disadvantages of Using the GET Method

- Since the data sent by the GET method are displayed in the URL, it is possible to bookmark the page with specific query string values.
- The GET method is not suitable for passing sensitive information such as the username and password, because these are fully visible in the URL query string as well as potentially stored in the client browser's memory as a visited page.

- Because the GET method assigns data to a server environment variable, the length of the URL is limited. So, there is a limitation for the total data to be sent.

Program

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>Example of PHP GET method</title>
</head>
<body>
<?php
if(isset($_GET["name"])){
   echo "<p>Hi, " . $_GET["name"] . "</p>";
}
?>
<form method="get" action="<?php echo $_SERVER["PHP_SELF"];?>">
   <label for="inputName">Name:</label>
   <input type="text" name="name" id="inputName">
   <input type="submit" value="Submit">
</form>
</body>
```

The POST Method

In POST method the data is sent to the server as a package in a separate communication with the processing script. Data sent through POST method will not visible in the URL.

Like $_GET, PHP provide another superglobal variable $_POST to access all the information sent via post method or submitted through an HTML form using the method="post".

Advantages and Disadvantages of Using the POST Method

- It is more secure than GET because user-entered information is never visible in the URL query string or in the server logs.
- There is a much larger limit on the amount of data that can be passed and one can send text data as well as binary data (uploading a file) using POST.
- Since the data sent by the POST method is not visible in the URL, so it is not possible to bookmark the page with specific query.

Program

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>Example of PHP POST method</title>
</head>
<body>
<?php
if(isset($_POST["name"])){
```

```
    echo "<p>Hi, " . $_POST["name"] . "</p>";
}
?>
<form method="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
    <label for="inputName">Name:</label>
    <input type="text" name="name" id="inputName">
    <input type="submit" value="Submit">
</form>
</body>
```

The PHP Date() Function

The PHP date() function convert a timestamp to a more readable date and time.

Program

```
<?php
$today = date("d/m/Y");
echo $today;
?>
```

Formatting the Dates and Times with PHP

d - Represent day of the month; two digits with leading zeros (01 or 31)

D - Represent day of the week in text as an abbreviation (Mon to Sun)

m - Represent month in numbers with leading zeros (01 or 12)

M - Represent month in text, abbreviated (Jan to Dec)

y - Represent year in two digits (08 or 14)

Y - Represent year in four digits (2008 or 2014)

Example

```
<?php
echo date("d/m/Y") . "<br>";
echo date("d-m-Y") . "<br>";
echo date("d.m.Y");
?>
```

h - Represent hour in 12-hour format with leading zeros (01 to 12)

H - Represent hour in in 24-hour format with leading zeros (00 to 23)

i - Represent minutes with leading zeros (00 to 59)

s - Represent seconds with leading zeros (00 to 59)

a - Represent lowercase ante meridiem and post meridiem (am or pm)

A - Represent uppercase Ante meridiem and Post meridiem (AM or PM)

Example

```
<?php
echo date("h:i:s") . "<br>";
echo date("F d, Y h:i:s A") . "<br>";
echo date("h:i a");
?>
```

PHP Include and Require Files

The include() and require() statement allow you to include the code contained in a PHP file within another PHP file. Including a file produces the same result as copying the script from the file specified and pasted into the location where it is called.

Syntax:

include("path/to/filename"); -Or- include "path/to/filename";

require("path/to/filename"); -Or- require "path/to/filename";

The include_once and require_once Statements

If you accidentally include the same file (typically functions or classes files) more than one time within your code using the include or require statements, it may cause conflicts.

To prevent this situation, PHP provides include_once and require_once statements. These statements behave in the same way as include and require statements with one exception.

The include_once and require_once statements will only include the file once even if asked to include it a second time i.e. if the specified file has already been included in a previous statement, the file is not included again.

Example: myfun.php

```
<?php
function multiplySelf($var){
   $var *= $var; // multiply variable by itself
   echo $var;
```

```
}
?>
<?php
// Including file
require "my_functions.php";
// Calling the function
multiplySelf(2); // Output: 4
echo "<br>";
// Including file once again
require "my_functions.php";
// Calling the function
multiplySelf(5); // Doesn't execute
?>
```

When you run the above script, you will see the error message something like this: "Fatal error: Cannot redeclare multiplySelf()". This occurs because the 'my_functions.php' included twice, this means the function multiplySelf() is defined twice, which caused PHP to stop script execution and generate fatal error. Now rewrite the above example with require_once.

```
<?php
// Including file
require_once "my_functions.php";
// Calling the function
multiplySelf(2); // Output: 4
echo "<br>";
```

```
// Including file once again
require_once "my_functions.php";
// Calling the function
multiplySelf(5); // Output: 25
?>
```

PHP Cookies

A cookie is a small text file that lets you store a small amount of data (nearly 4KB) on the user's computer. They are typically used to keeping track of information such as username that the site can retrieve to personalize the page when user visit the website next time.

The setcookie() function is used to set a cookie in PHP. Make sure you call the setcookie() function before any output generated by your script otherwise cookie will not set.

Syntax:

```
setcookie(name, value, expire, path, domain, secure);
```

Example:

```
<?php
// Setting a cookie
setcookie("username", "John Carter", time()+30*24*60*60);
?>
```

Accessing Cookies Values

The PHP $_COOKIE superglobal variable is used to retrieve a cookie value. It typically an associative array that contains a list of all the cookies values sent by the browser in the current request, keyed by cookie name.

Example

```
<?php
// Accessing an individual cookie value
echo $_COOKIE["username"];
?>
```

It's a good practice to check whether a cookie is set or not before accessing its value.

```
<?php
// Verifying whether a cookie is set or not
if(isset($_COOKIE["username"])){
   echo "Hi " . $_COOKIE["username"];
} else{
   echo "Welcome Guest!";
}
?>
```

Removing Cookies

You can delete a cookie by calling the same setcookie() function with the cookie name and any value (such as an empty string) however this time you need the set the expiration date in the past

Example

```
<?php
// Deleting a cookie
setcookie("username", "", time()-3600);
?>
```

PHP Sessions

Although you can store data using cookies but it has some security issues. Since cookies are stored on user's computer it is possible for an attacker to easily modify a cookie content to insert potentially harmful data in your application that might break your application.

Also every time the browser requests a URL to the server, all the cookie data for a website is automatically sent to the server within the request. It means if you have stored 5 cookies on user's system, each having 4KB in size, the browser needs to upload 20KB of data each time the user views a page, which can affect your site's performance.

You can solve both of these issues by using the PHP session. A PHP session stores data on the server rather than user's computer. In a session based environment, every user is identified through a unique number called session identifier or SID.

This unique session ID is used to link each user with their own information on the server like emails, posts, etc.

Starting a PHP Session

```
<?php
// Starting session
session_start();
?>
```

Storing and Accessing Session Data

Example 1:

```
<?php
// Starting session
session_start();
// Storing session data
$_SESSION["firstname"] = "Peter";
$_SESSION["lastname"] = "Seker";
?>
```

Example 2:

```
<?php
// Starting session
session_start();
// Accessing session data
echo 'Hi, ' . $_SESSION["firstname"] . ' ' . $_SESSION["lastname"];
?>
```

Destroying a Session

If you want to remove certain session data, simply unset the corresponding key of the $_SESSION associative array

```
<?php
// Starting session
session_start();
// Destroying session
session_destroy();
?>
```

PHP Send Emails

PHP built-in mail() function for creating and sending email messages to one or more recipients dynamically from your PHP application either in a plain-text form or formatted HTML.

mail(to, subject, message, headers, parameters)

Example 1:

```
<?php
$to = 'maryjane@email.com';
$subject = 'Marriage Proposal';
$message = 'Hi Jane, will you marry me?';
$from = 'peterparker@email.com';
// Sending email
```

```
if(mail($to, $subject, $message)){
   echo 'Your mail has been sent successfully.';
} else{
   echo 'Unable to send email. Please try again.';
}
?>
```

Example 2:

```
<?php
$to = 'maryjane@email.com';
$subject = 'Marriage Proposal';
$from = 'peterparker@email.com';
// To send HTML mail, the Content-type header must be set
$headers  = 'MIME-Version: 1.0' . "\r\n";
$headers .= 'Content-type: text/html; charset=iso-8859-1' . "\r\n";
// Create email headers
$headers .= 'From: '.$from."\r\n".
   'Reply-To: '.$from."\r\n" .
   'X-Mailer: PHP/' . phpversion();
// Compose a simple HTML email message
```

```
$message = '<html><body>';
$message .= '<h1 style="color:#f40;">Hi Jane!</h1>';
$message .= '<p style="color:#080;font-size:18px;">Will you marry me?</p>';
$message .= '</body></html>';
// Sending email
if(mail($to, $subject, $message, $headers)){
    echo 'Your mail has been sent successfully.';
} else{
    echo 'Unable to send email. Please try again.';
}
?>
```

PHP FORM

Program 1:

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <title>Contact Form</title>
</head>
<body>
    <h2>Contact Us</h2>
    <p>Please fill in this form and send us.</p>
    <form action="process-form.php" method="post">
        <p>
            <label for="inputName">Name:<sup>*</sup></label>
            <input type="text" name="name" id="inputName">
        </p>
        <p>
            <label for="inputEmail">Email:<sup>*</sup></label>
            <input type="text" name="email" id="inputEmail">
        </p>
        <p>
            <label for="inputSubject">Subject:</label>
            <input type="text" name="subject" id="inputSubject">
        </p>
```

```
        <p>
            <label for="inputComment">Message:<sup>*</sup></label>
            <textarea name="message" id="inputComment" rows="5"
cols="30"></textarea>
        </p>
        <input type="submit" value="Submit">
        <input type="reset" value="Reset">
    </form>
</body>
</html>
```

Program 2:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Contact Form</title>
</head>
<body>
    <h1>Thank You</h1>
```

```
    <p>Here is the information you have submitted:</p>

    <ol>

        <li><em>Name:</em> <?php echo $_POST["name"]?></li>

        <li><em>Email:</em> <?php echo $_POST["email"]?></li>

        <li><em>Subject:</em> <?php echo $_POST["subject"]?></li>

        <li><em>Message:</em> <?php echo $_POST["message"]?></li>

    </ol>

</body>

</html>
```

PHP FORM VALIDATION

Program 1: index.php

```
<!DOCTYPE html>
<html>

<head>
    <title>PHP Form Validation</title>
    <link rel="stylesheet" href="style.css">
</head>

<body>
    <?php require "validation.php" ?>


    <p class="msg">A Simple Registration Form     </p>
    <span class="error">(*) indicates required field</span>

    <!-- Using 'post' method for secure data sharing -->
```

```
    <form method="post" class="container" role="form"
          action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);
?>">
        <table>
            <tr class="tag">
                <td><label for="userName">Enter your Username</label>
                    <span class="error">*</span>
                </td>
                <td><input type="text" name="userName"
class="details"></td>
            </tr>
            <tr>
                <td></td>
                <td><span class="error">
                        <?php echo $userNameErr; ?>
                    </span></td>
            </tr>

            <tr class="tag">
                <td><label for="emailID">Enter your Email Id</label>
                    <span class="error">*</span>
                </td>
                <td><input type="text" name="emailID"
class="details"></td>
            </tr>
            <tr>
                <td></td>
                <td><span class="error">
                        <?php echo $emailIDErr; ?>
                    </span></td>
            </tr>

            <tr class="tag">
                <td><label for="phoneNo">Enter your Phone
Number</label>
                    <span class="error">*</span>
                </td>
                <td><input type="text" name="phoneNo"
class="details"></td>
            </tr>
            <tr>
```

```
                <td></td>
                <td><span class="error">
                        <?php echo $phoneNoErr; ?>
                    </span></td>
            </tr>

            <tr class="tag">
                <td><label for="website">Enter your Your
Website</label>
                </td>
                <td><input type="text" name="website"
class="details"></td>
            </tr>
            <tr>
                <td></td>
                <td><span class="error">
                        <?php echo $websiteErr; ?>
                    </span></td>
            </tr>

            <tr class="tag">
                <td><label for="gender">Enter your Gender</label>
                    <span class="error">*</span>
                </td>
                <td>
                    <input type="radio" name="gender" value="male">
Male
                    <input type="radio" name="gender" value="female">
Female
                    <input type="radio" name="gender" value="other">
Other
                </td>
            </tr>
            <tr>
                <td></td>
                <td><span class="error">
                        <?php echo $genderErr; ?>
                    </span></td>
            </tr>

            <tr class="tag">
```

```
                <td><label for="tc">Agree to our Terms &
Conditions</label>
                    <span class="error">*</span>
                </td>
                <td><input type="checkbox" name="tc"></td>
            </tr>
            <tr>
                <td></td>
                <td><span class="error">
                        <?php echo $tcErr; ?>
                    </span></td>
            </tr>

            <tr class="tag">
                <td><input type="submit" name="submit" value="Submit"
class="btn"></td>
                <td></td>
            </tr>
        </table>
    </form>

    <?php
     // Checking if submit button is pressed or not
     if (isset ($_POST['submit'])) {
          // Checking if there is any error or not
          if ($userNameErr == "" && $emailIDErr == "" &&
               $phoneNoErr == "" && $genderErr == "" &&
               $websiteErr == "" && $tcErr == "") {
               echo "<p class='msg'>You have been sucessfully

registered    </p>";
               echo "<h3>Your Details are :</h3>";
               echo "<p class='info'>User Name : " . $userName .
"</p>";
               echo "<p class='info'>EmailID ID : " . $emailID .
"</p>";
               echo "<p class='info'>Phone Number : " . $phoneNo .
"</p>";
               echo "<p class='info'>Website : " . $website .
"</p>";
               echo "<p class='info'>Gender : " . $gender . "</p>";
          } else {
```

```
                echo "<p class='msg'>You shared Invalid details
                    <br/>Please provide correct data!</p>";
            }
        }
        ?>

</body>

</html>
```

Program 2: validation.php

```
<!DOCTYPE html>
<html>

<head>
    <title>PHP Form Validation</title>
    <link rel="stylesheet" href="style.css">
</head>

<body>
    <?php require "validation.php" ?>


    <p class="msg">A Simple Registration Form    </p>
    <span class="error">(*) indicates required field</span>

    <!-- Using 'post' method for secure data sharing -->

    <form method="post" class="container" role="form"
          action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);
?>">
        <table>
            <tr class="tag">
                <td><label for="userName">Enter your Username</label>
                    <span class="error">*</span>
                </td>
```

```
                <td><input type="text" name="userName"
class="details"></td>
            </tr>
            <tr>
                <td></td>
                <td><span class="error">
                        <?php echo $userNameErr; ?>
                    </span></td>
            </tr>

            <tr class="tag">
                <td><label for="emailID">Enter your Email Id</label>
                    <span class="error">*</span>
                </td>
                <td><input type="text" name="emailID"
class="details"></td>
            </tr>
            <tr>
                <td></td>
                <td><span class="error">
                        <?php echo $emailIDErr; ?>
                    </span></td>
            </tr>

            <tr class="tag">
                <td><label for="phoneNo">Enter your Phone
Number</label>
                    <span class="error">*</span>
                </td>
                <td><input type="text" name="phoneNo"
class="details"></td>
            </tr>
            <tr>
                <td></td>
                <td><span class="error">
                        <?php echo $phoneNoErr; ?>
                    </span></td>
            </tr>

            <tr class="tag">
                <td><label for="website">Enter your Your
Website</label>
```

```
                </td>
                <td><input type="text" name="website"
class="details"></td>
            </tr>
            <tr>
                <td></td>
                <td><span class="error">
                        <?php echo $websiteErr; ?>
                    </span></td>
            </tr>

            <tr class="tag">
                <td><label for="gender">Enter your Gender</label>
                    <span class="error">*</span>
                </td>
                <td>
                    <input type="radio" name="gender" value="male">
Male
                    <input type="radio" name="gender" value="female">
Female
                    <input type="radio" name="gender" value="other">
Other
                </td>
            </tr>
            <tr>
                <td></td>
                <td><span class="error">
                        <?php echo $genderErr; ?>
                    </span></td>
            </tr>

            <tr class="tag">
                <td><label for="tc">Agree to our Terms &
Conditions</label>
                    <span class="error">*</span>
                </td>
                <td><input type="checkbox" name="tc"></td>
            </tr>
            <tr>
                <td></td>
                <td><span class="error">
                        <?php echo $tcErr; ?>
```

```
                    </span></td>
            </tr>

            <tr class="tag">
                <td><input type="submit" name="submit" value="Submit"
class="btn"></td>
                <td></td>
            </tr>
        </table>
    </form>

    <?php
     // Checking if submit button is pressed or not
     if (isset ($_POST['submit'])) {
          // Checking if there is any error or not
          if ($userNameErr == "" && $emailIDErr == "" &&
               $phoneNoErr == "" && $genderErr == "" &&
               $websiteErr == "" && $tcErr == "") {
               echo "<p class='msg'>You have been sucessfully

registered    </p>";
               echo "<h3>Your Details are :</h3>";
               echo "<p class='info'>User Name : " . $userName .
"</p>";
               echo "<p class='info'>EmailID ID : " . $emailID .
"</p>";
               echo "<p class='info'>Phone Number : " . $phoneNo .
"</p>";
               echo "<p class='info'>Website : " . $website .
"</p>";
               echo "<p class='info'>Gender : " . $gender . "</p>";
          } else {

               echo "<p class='msg'>You shared Invalid details
                   <br/>Please provide correct data!</p>";
          }
     }
     ?>

</body>
```

```
</html>
```

Program 3: style.css

```
body {
  display: flex;
  align-items: center;
  flex-direction: column;
  font-family: "Gill Sans", "Gill Sans MT", Calibri, "Trebuchet MS",
sans-serif;
}

.container {
  border: 1px solid black;
  padding: 30px;
  border-radius: 10px;
  box-shadow: 5px 5px #77777788;
}

.error {
  color: red;
  margin-bottom: 30px;
}

.msg {
  color: brown;
  margin: 20px 0px 10px 0px;
  font-size: 25px;
  font-weight: bold;
}

.tags {
  margin: 0px 0px 20px 0px;
}

input {
  border-radius: 10px;
}
```

```css
.details {
  width: 200px;
  height: 20px;
}

.info {
  margin: 2px 0;
}

.btn {
  margin-top: 20px;
  width: 120px;
  height: 30px;
  background-color: black;
  color: white;
}

/* ================================ */
#countdown-container {
  display: flex;
  justify-content: center;
  align-items: center;
}

.countdown-section {
  text-align: center;
  margin: 0 10px;
}

.countdown-section span {
  font-size: 26px;
  font-weight: bold;
  color: #000;
}

.countdown-label {
  font-size: 14px;
  color: #777;
}
```

PHP OOPS

Classes and Objects

A class is a blueprint for creating objects while an object is an instance of a class.

Example:

```
<?php
class Car {
    // Properties
    public $color;
    // Methods
    function setColor($color) {
        $this->color = $color;
    }
}
// Create an object
$myCar = new Car();
$myCar->setColor('blue');
?>
```

Properties and Methods

Properties represent characteristics of an object, and methods are actions an object can perform. Visibility keywords such as public, protected, or private define access level.

Example:

```
<?php
class Car {
    private $model;
    public function setModel($model) {
        $this->model = $model;
    }
    public function getModel() {
        return $this->model;
    }
}
$myCar = new Car();
$myCar->setModel('Tesla Model S');
echo $myCar->getModel();
?>
```

Inheritance

Inheritance allows a class to take on properties and methods from another class.

Example:

```
<?php
class Car {
    protected $model;
```

```
    public function setModel($model) {
        $this->model = $model;
    }
    public function getModel() {
        return $this->model;
    }
}
class ElectricCar extends Car {
    private $batteryLife;
    public function setBatteryLife($life) {
        $this->batteryLife = $life;
    }
    public function getBatteryLife() {
        return $this->batteryLife;
    }
}
$myTesla = new ElectricCar();
$myTesla->setModel('Tesla Model 3');
$myTesla->setBatteryLife('310 miles');
?>
```

Encapsulation

Encapsulation is achieved by restricting direct access to some of an object's components and can be enforced using visibility keywords

Example

```
<?php
class Car {
   private $model;
   public function setModel($model) {
      $this->model = $model;
   }
   public function getModel() {
      return $this->model;
   }
}
?>
```

Polymorphism

Polymorphism allows methods to have different implementations based on the subclass they belong to.

```
<?php
class Car {
   protected $model;
   public function setModel($model) {
      $this->model = $model;
   }
   public function getModel() {
      return $this->model;
```

```
    }
}
class ElectricCar extends Car {
    public function getModel() {
        return 'Electric Car: ' . $this->model;
    }
}
$myCar = new Car();
$myCar->setModel('Audi A4');
$myTesla = new ElectricCar();
$myTesla->setModel('Tesla Model X');
echo $myCar->getModel();
echo "\n";
echo $myTesla->getModel();
?>
```

Interfaces

Interfaces specify what methods a class must implement, without defining how these methods should be handeled.

Example:

```
<?php
interface Chargeable {
```

```
    public function charge();
}
interface GPSInterface {
    public function getCoordinates();
}
class ElectricCar extends Car implements Chargeable, GPSInterface {
    public function charge() {
        // Implementation for charge
    }
    public function getCoordinates() {
        // Implementation for getCoordinates
    }
}
?>
```

Abstract Classes

Abstract classes cannot be instantiated and are used as a base for other classes. Methods defined as abstract in an abstract class must be implemented by subclasses

Example:

```
<?php
abstract class Vehicle {
    protected $model;

    abstract public function startEngine();
```

```
    public function setModel($model) {
        $this->model = $model;
    }
    public function getModel() {
        return $this->model;
    }
}
class Car extends Vehicle {
    public function startEngine() {
        echo "Engine of " . $this->getModel() . " started";
    }
}
$myCar = new Car();
$myCar->setModel('Fiat Punto');
$myCar->startEngine();
?>
```

PHP crud operation

Database Connection

```
<?php
$db=new mysqli("localhost","root","","manicrud");
if(!$db)
{
	die('error in database'.mysqli_error($db));
	//echo"erroro in database";
}
?>
```

Create

```
<?php
include 'dbconnect.php';
if(isset($_POST['submit']))
{
	$name=$_POST['name'];
	$password=$_POST['password'];
	$email=$_POST['email'];
	$mobile=$_POST['mobile'];
	$address=$_POST['address'];
	$query="insert into democrud(name,password,email,mobile,address)
	values('$name','$password','$email','$mobile','$address')";
	$result=mysqli_query($db,$query);
```

```
    if($result)
        {
            echo"Registered successfully";
            header("location:read.php");
        }
    else
        {
            die('error'.mysqli_error($db));
        }
}
?>
<html lang="en">
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css">
<title>crud2</title>
</head>
<body>
<h1><center> USER FORM</title></h1>
<div class="container my-5">
<form method="post">
<div class="form-group">
<label>NAME:</label>
```

```
<input type="text" name="name" placeholder="enter your name" class="form-control">

</div>

<div class="form-group">

<label>PASSWORD:</label>

<input type="password" name="password" placeholder="enter your password" class="form-control">

</div>

<div class="form-group">

<label>EMAIL:</label>

<input type="email" name="email" placeholder="enter your email" class="form-control">

</div>

<div class="form-group">

<label>MOBILE:</label>

<input type="text" name="mobile" placeholder="enter your mobile" class="form-control">

</div>

<div class="form-group">

<label>ADDRESS:</label>

<input type="address" name="address" placeholder="enter your address" class="form-control">

</div>

<button class="btn btn-info" type="submit" name="submit">REGISTER</button>

        <button class="btn btn-danger" type="reset">RESET</button>
```

```
</form>
</div>
</body>
</html>
```

Read

```
<?php
include'dbconnect.php';
?>

<html lang="en">
<head>
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>crud1</title>
</head>

<body>
<div class="container">

<button class="btn btn-success my-5" ><a href="create.php" class="text-
light">Creat Profile</a></button>
```

```
<table class="table">
<thead>
<tr>
<th>id</th>
<td>Name</td>
<td>Password</td>
<td>Email</td>
<td>Mobile</td>
<td>Address</td>
<td>Operation</td>
</tr>
</thead>
<tbody>
<?php
$qry="SELECT * FROM democrud";
$result=mysqli_query($db,$qry);
if($result)
{
        while($row=mysqli_fetch_assoc($result))
        {
   $id=$row['id'];
        $name=$row['name'];
        $password=$row['password'];
        $email=$row['email'];
```

```
$mobile=$row['mobile'];
$address=$row['address'];
echo'<tr>
<th>'.$id.'</th>
<td>'.$name.'</td>
<td>'.$password.'</td>
<td>'.$email.'</td>
<td>'.$mobile.'</td>
<td>'.$address.'</td>

<td><button class="btn btn-warning" class="text-dark"><a href="updated.php?  upid='.$id.'class="text-light" ">Update</a></button>
<button class="btn btn-warning" class="text-dark"><a href="deleted.php?deldid='.$id.' class="text-light"">delete</a></button>
</td>
</tr>';
}
}
?>
</tbody>
</table>
</body>
</html>
```

update

```
<?php
include'dbconnect.php';
$id=$_GET['upid'];
$qry=$_GET['upid'];
$qry="SELECT * FROM democrud where id='$id'";
$result=mysqli_query($db,$qry);
if($row=mysqli_fetch_assoc($result)){
$name=$row['name'];
$password=$row['password'];
$email=$row['email'];
$mobile=$row['mobile'];
$address=$row['address'];
if(isset($_POST['update']))
{
$name=$_POST['name'];
$password=$_POST['password'];
$email=$_POST['email'];
$mobile=$_POST['mobile'];
$address=$_POST['address'];

$query="UPDATE democrud set
id='$id',name='$name',password='$password',email='$email',mobile='$mobile',add
ress='$address' where id='$id'";
```

```
$result=mysqli_query($db,$query);
if($result)
{
        echo"updated successfully";
        header('location:read.php');
}
else
{
        die(mysqli_error($db));
}
}
}
?>

<html lang="en">
<head>-
<meta name="viewport" content="width=device-width, initial-scale=1">

   <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css">

<title>crud3</title>
</head>
```

```
<body>
<h1><center> UPDATE FORM</title></h1>
<div class="container my-5">
<form method="post">
<div class="form-group">
<label>NAME:</label>
<input type="text" name="name" placeholder="enter your name" class="form-control"  value=<?php echo $name;?>>
</div>
<div class="form-group">
<label>PASSWORD:</label>
<input type="password" name="password" placeholder="enter your password" class="form-control" value=<?php echo $password;?>>
</div>
<div class="form-group">
<label>EMAIL:</label>
<input type="email" name="email" placeholder="enter your email" class="form-control" value=<?php echo $email;?>>
</div>
<div class="form-group">
<label>MOBILE:</label>
<input type="text" name="mobile" placeholder="enter your mobile" class="form-control" value=<?php echo $mobile;?>>
</div>
<div class="form-group">
```

```
<label>ADDRESS:</label>

<input type="address" name="address" placeholder="enter your address"
class="form-control" value=<?php echo $address;?>>

</div>

        <button class="btn btn-info" type="submit" name="update"
color="white">UPDATE</button>

        <button class="btn btn-danger" type="reset"><a href="deleted.php">
CANCEL</a></button>

</form>

</div>

</body>

</html>
```

Delete

```
<?php

include"dbconnect.php";

if(isset($_GET['deldid']))

{

$id=$_GET['deldid'];

$sql="DELETE FROM democrud where id='$id'";

$result=mysqli_query($db,$sql);

if($result)
```

```
{
        //echo"<script>alert('are you sure')</script>";
        header("location:read.php");
}
else
{
        die(mysqli_error($db));
}
}
        ?>
```