Pandit Deendayal Energy University, Gandhinagar

School of Technology

Department of Computer Science & Engineering

Introduction to Web Technology (23CP306P)

Name: Ravi Jamanbhai Makwana

Enrolment No: 21BCP418

Semester: 5

Division: 6

Group: 12

Branch: Computer Science Engineering

Practical: 4

Aim: Implement the server-side scripting using PHP language

Hardware Requirement:

1. **Computer:** A desktop or laptop computer with sufficient processing power and memory to run web applications. XAMPP is not resource-intensive, so most modern computers should suffice.
2. **Storage Space:** Adequate storage space for your web development projects and the XAMPP software. A few gigabytes of free space should be more than enough.
3. **RAM:** At least 2GB of RAM for smooth operation. More RAM may be required if you plan to run resource-intensive applications.

Software Requirement:

1. **Operating System:** XAMPP is compatible with Windows, macOS, and Linux. Ensure your system meets the following OS-specific requirements:
   - Windows: XP, Vista, 7, 8, 10
   - macOS: 10.6 or later
   - Linux: Any modern distribution with a 32- or 64-bit architecture.

2. **Web Browser:** You will need a web browser for testing your web applications. Popular choices include Google Chrome, Mozilla Firefox, and Microsoft Edge.

3. **XAMPP Software:** Download and install the XAMPP software from the official website (https://www.apachefriends.org/). Make sure to download the version that matches your operating system.

4. **Text Editor or IDE:** A text editor or integrated development environment (IDE) for writing and editing your web application code. Popular options include Visual Studio Code, Sublime Text, and PHP Storm.

5. **Database Management Tool:** If your web application uses databases, you may need a database management tool such as phpMyAdmin (included in XAMPP) or MySQL Workbench.

6. **Optional: Version Control System:** Consider using a version control system (e.g., Git) for tracking changes in your code.

7. **Optional:** Content Management System (CMS): If you plan to use a CMS like WordPress, Joomla, or Drupal, make sure to meet their specific software requirements.

Knowledge Requirement:

1. Basic Programming Concepts:

- Variables, data types, and operators.
- Control structures (if statements, loops).
- Functions and procedures.

2. PHP Syntax:

- Familiarity with PHP syntax, including variables, arrays, loops, and functions.

3. Web Development Basics:

- Understanding of HTML and CSS.
- Knowledge of how web servers work.

4. Server-Side Scripting:

- Understanding of the difference between server-side and client-side scripting.

5. PHP Data Types:

- Scalars (int, float, string, boolean).
- Composite data types (arrays, objects).
- Special data types (NULL).

6. Functions and Control Structures:

- How to define and call functions.
- Conditional statements (if, else, switch).
- Looping (for, while, foreach).

7. Super Globals:

- Knowledge of PHP's superglobal arrays like `$_GET`, `$_POST`, `$_SESSION`, and `$_COOKIE`.

8. Error Handling:

- Understanding of error reporting and handling mechanisms.

9. Working with Databases:

- Connecting to databases using PHP (e.g., MySQL, PostgreSQL, SQLite).
- Executing SQL queries safely.

10. Form Handling:

- Handling form submissions and user input validation.

11. Sessions and Cookies:

- Working with sessions and cookies for user state management.

12. File Handling:

- Reading and writing files, and working with directories.

13. Security Best Practices:

- Awareness of common security vulnerabilities (e.g., SQL injection, XSS) and how to mitigate them.

14. Object-Oriented Programming (OOP):

- Basics of OOP in PHP, including classes, objects, inheritance, and encapsulation.

15. MVC (Model-View-Controller) Architecture:

- Understanding the concept of separating the application into models, views, and controllers.

16. API Integration:

- How to work with RESTful and other types of APIs in PHP.

17. Frameworks and Libraries:

- Familiarity with popular PHP frameworks (e.g., Laravel, Symfony, CodeIgniter) and libraries.

18. Composer:

- Dependency management using Composer, a PHP package manager.

19. Debugging and Profiling:

- Using debugging tools and techniques to find and fix issues in your code.

20. Version Control:

- Using version control systems like Git to manage your PHP projects.

21. Server Configuration:

- Knowledge of server configurations, such as PHP.ini settings.

22. Caching:

- Implementing caching mechanisms to improve application performance.

23. Web Security:

- Knowledge of best practices for securing PHP applications and servers.

24. Regular Expressions:

- Understanding and using regular expressions for pattern matching.

25. Web Application Deployment:

- Deploying PHP applications on web servers.

26. Performance Optimization:

- Techniques for optimizing PHP code and improving application performance.

27. API Documentation:

- How to create and document APIs in PHP.

28. Testing and Test-Driven Development (TDD):
- Strategies for testing PHP code and implementing TDD practices.

29. Continuous Integration/Continuous Deployment (CI/CD):
- Integrating PHP projects into CI/CD pipelines for automated testing and deployment.

Theory:

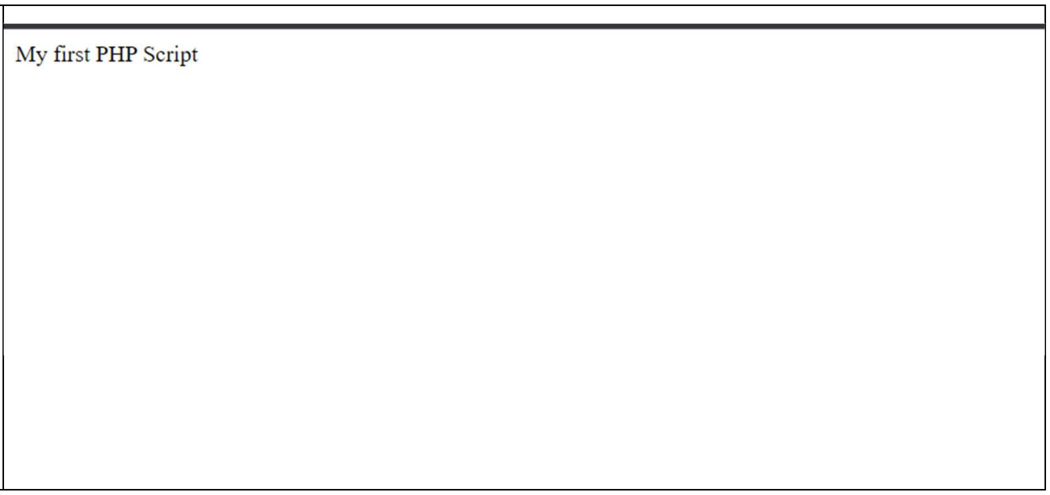
Introduction to PHP

PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages. PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP. PHP is an acronym for "PHP: Hypertext Preprocessor" PHP is a widely-used, open-source scripting language PHP scripts are executed on the server.

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>First Practical</title>
</head>
<body>
<?php
echo "My first PHP Script";
?>

</body>
</html>
```

Output:	 My first PHP Script

Variable Declaration in PHP

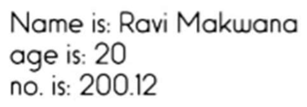
Rules for Declaring PHP Variables:

In PHP, variable names are denoted by a dollar sign ($) followed by the variable's name. These names are subject to certain rules: they can consist of alphanumeric characters and underscores (A-z, 0-9, _), but they must start with a letter or an underscore (_) character. Spaces are not allowed within variable names. It's essential to note that variable names in PHP are case-sensitive, meaning that $name and $NAME are considered distinct variables. Furthermore, they cannot begin with numbers or special symbols. Adhering to these conventions is crucial when working with PHP variables to ensure proper functionality and avoid errors in your code.

Code:

```
<?php
$str = "Ravi Makwana";
$age = 20;
$no = 200.12;
echo "Name is: $str <br/>"; echo "age is: $age <br/>"; echo "no
is: $no <br/>"
?>
```

Output:



```
Name is: Ravi Makwana
age is: 20
no. is: 200.12
```

Control Statements in PHP

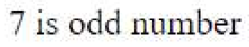
PHP If-Else Statement

The if-else statement executes one block of code if a specified condition is true and another block if the condition is false.

Code:

```
<?php
     $num=7; if($num%2==0){
echo "$num is even number";
}else{
echo "$num is odd number";
}
?>
```

Output:



```
7 is odd number
```

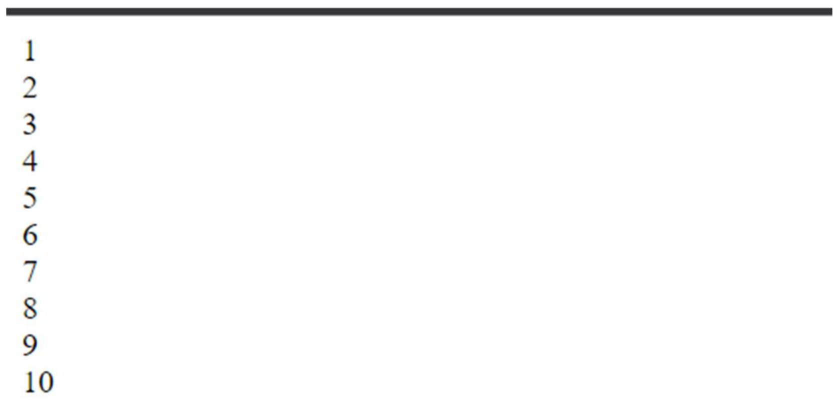
PHP For Loop

The for loop is used to execute a block of code for a specified number of times when the number of iterations is known.

Code:

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

Output:



```
1
2
3
4
5
6
7
8
9
10
```

PHP Arrays

PHP arrays are used to store multiple values of similar types in a single variable.

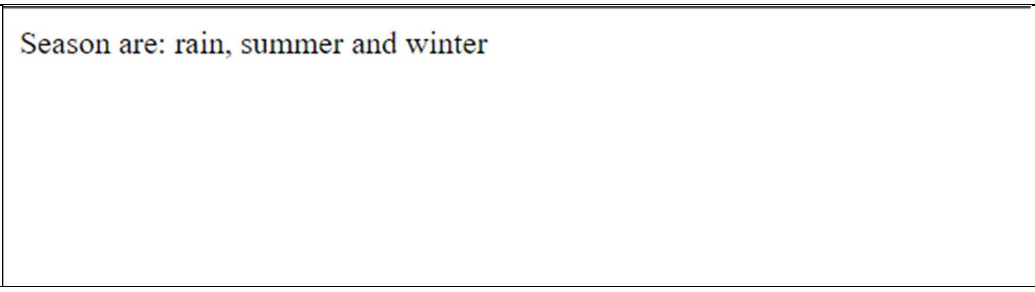
There are three types of arrays in PHP:

Indexed Array:

Code:

```php
<?php
$season=array("rain","summer","winter");
echo "Season are: $season[0], $season[1] and $season[2]";
?>
```

Output:



```
Season are: rain, summer and winter
```

Associative Array

Code:

```
<?php
$salary=array("Tapan"=>"500000","Rohit"=>"550000",
"Rahul"=>"600000");
echo "Ravi salary: ".$salary["Tapan"]."<br/>"; echo "Rohit salary:
".$salary["Rohit"]."<br/>"; echo "Rahul salary:
".$salary["Rahul"]."<br/>";
?>
```

Output:

```
Ravi salary: 500000
Rohit salary: 550000
Rahul salary: 600000
```

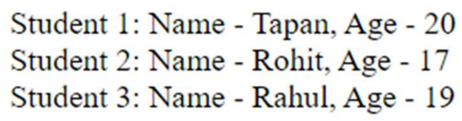
Multidimensional Array

Code:

```
<?php
$studentData = array(
array("name" => "Tapan", "age" => 20), array("name" => "Rohit",
"age" => 17), array("name" => "Rahul", "age" => 19)
);

echo "Student 1: Name - " . $studentData[0]["name"] . ", Age - "
. $studentData[0]["age"] . "<br>";
echo "Student 2: Name - " . $studentData[1]["name"] . ", Age - "
. $studentData[1]["age"] . "<br>";
echo "Student 3: Name - " . $studentData[2]["name"] . ", Age - "
. $studentData[2]["age"];
?>
```

Output:



```
Student 1: Name - Tapan, Age - 20
Student 2: Name - Rohit, Age - 17
Student 3: Name - Rahul, Age - 19
```

PHP Form Handling

PHP Get Form

The get request is the default form request and sends data as part of the URL. It's not secure for sensitive data.

HTML Code:

```
<form action="first.php" method="get">
Name: <input type="text" name="name" />
    <input type="submit" value="visit"
/>
</form>
```

PHP Code:

```
<?php
$name=$_GET["name"];
echo "Welcome, $name";
?>
```

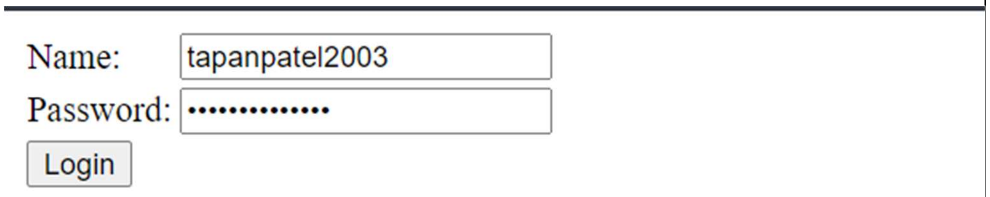
Output:	Name: Tapan Patel visit

PHP Post Form

The post request is used for sensitive or large data as it's not visible in the URL.

HTML Code:	
	```
	<!DOCTYPE html>
	<html>
	<head>
	</head>
	<body>
	  <form action="first.php" method="post">
	    <table>
	      <tr>
	        <td>Name:</td>
	        <td><input type="text" name="name" required /></td>
	      </tr>
	      <tr>
	        <td>Password:</td>
	        <td><input type="password" name="password" required
	/></td>
	      </tr>
	      <tr>
	        <td colspan="2"><input type="submit" value="Login"
	/></td>
	      </tr>
	    </table>
	  </form>
	</body>
	</html>
	```

PHP Code:	
	```
	<?php
	$name=$_POST["name"];
	$password=$_POST["password"];
	echo "Welcome: $name, your password is: $password";
	?>
	```

Output:	 Name: tapanpatel2003 Password: ••••••••••••••

Form Validation

To ensure the data is safe and valid, you should validate and sanitize form input. Here's an example of form validation:

Code:	(see code below)

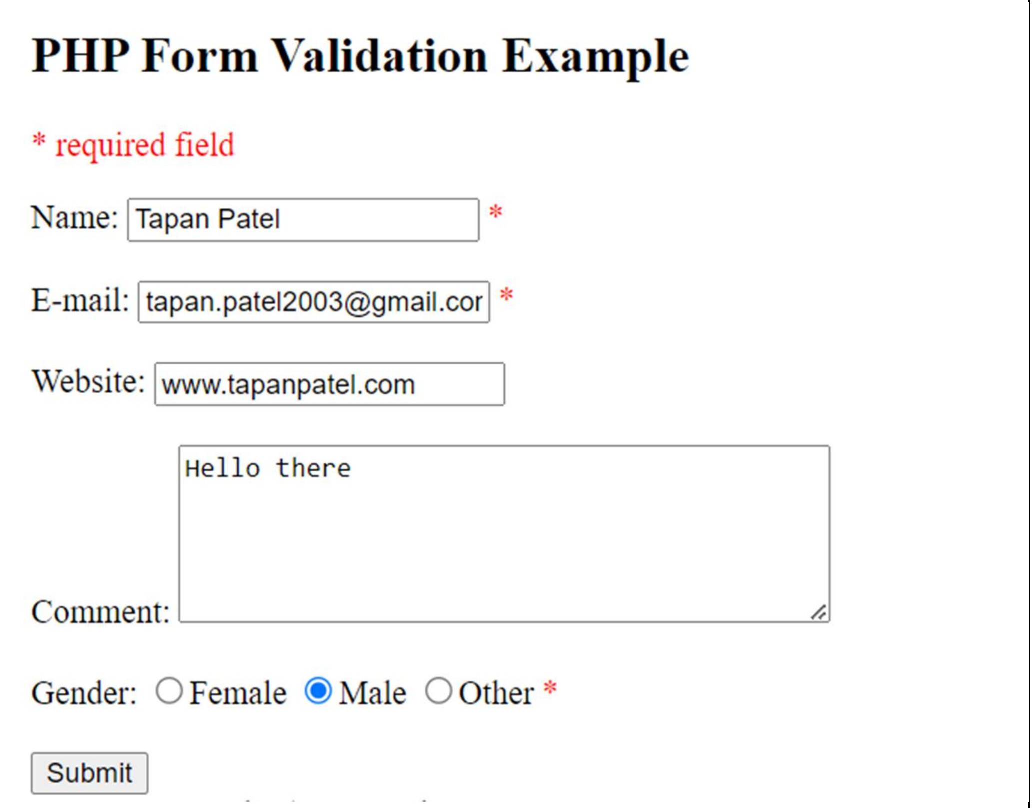
```
<!DOCTYPE html>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>
<?php

$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty($_POST["name"])) {
        $nameErr = "Name is required";
    } else {
        $name = test_input($_POST["name"]);
    }
    if (empty($_POST["email"])) {
        $emailErr = "Email is required";
    } else {
        $email = test_input($_POST["email"]);
    }
    if (empty($_POST["website"])) {
        $website = "";
    } else {
        $website = test_input($_POST["website"]);
    }
    if (empty($_POST["comment"])) {
        $comment = "";
    } else {
        $comment = test_input($_POST["comment"]);
    }
    if (empty($_POST["gender"])) {
        $genderErr = "Gender is required";
    } else {
        $gender = test_input($_POST["gender"]);
    }
}

function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
?>
<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field</span></p>
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
Name: <input type="text" name="name">
<span class="error">* <?php echo $nameErr;?></span>
<br><br>
E-mail: <input type="text" name="email">
<span class="error">* <?php echo $emailErr;?></span>
<br><br>
Website: <input type="text" name="website">
<span class="error"> <?php echo $websiteErr;?></span>
<br><br>
Comment: <textarea name="comment" rows="5" cols="40"></textarea>
<br><br>
Gender:
<input type="radio" name="gender" value="female">Female
<input type="radio" name="gender" value="male">Male
<input type="radio" name="gender" value="other">Other
<span class="error">* <?php echo $genderErr;?></span>
<br><br>
<input type="submit" name="submit" value="Submit">
</form>
<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>
</body>
</html>
```

Output:



PHP Form Validation Example

* required field

Name:	Tapan Patel *
E-mail:	tapan.patel2003@gmail.cor *
Website:	www.tapanpatel.com
Comment:	Hello there
Gender:	Female [✓ Male] Other *

PHP Cookie

PHP cookies are like digital sticky notes that websites can leave on your computer. They store small pieces of information, like your preferences or login status, so the website can remember you and provide a better experience next time you visit.

PHP setcookie() Function

PHP setcookie() function is used to set cookie with HTTP response. Once cookie is set, you can access it by $_COOKIE superglobal variable.

Syntax:

bool setcookie (string $name [, string $value [, int $expire = 0 [, string $path [, string $domain [, bool $secure = false [, bool $httponly = false]]]]]])

Example:

setcookie("CookieName", "CookieValue"); // Defining name and value only

setcookie("CookieName", "CookieValue", time() + 1 * 60 * 60); // Using expiry in 1 hour

PHP $_COOKIE Superglobal

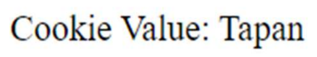
The $_COOKIE superglobal variable is used to retrieve cookies.

Setcookie.php	`<?php`
	`setcookie("user", "Tapan");`
	`?>`

Displaycookie.php

```
<html>
<body>
<?php
if (!isset($_COOKIE["user"])) {
    echo "Sorry, the cookie is not found!";
} else {
    echo "Cookie Value: " . $_COOKIE["user"];
}
?>
</body>
</html>
```

Output:



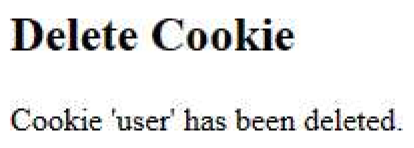
```
Cookie Value: Tapan
```

Deleting a Cookie

You can delete a cookie by setting its expiration date in the past.

Code:

```
<?php
if (isset($_COOKIE["user"])) {
    setcookie("user", "", time() - 3600, "/");
    $message = "Cookie 'user' has been deleted.";
} else {
    $message = "Cookie 'user' does not exist.";
}
?>
<!DOCTYPE html>
<html>
<head>
    <title>Delete Cookie</title>
</head>
<body>
<h2>Delete Cookie</h2>
<p><?php echo $message; ?></p>
</body>
</html>
```

Output:	 **Delete Cookie** Cookie 'user' has been deleted.

PHP Session

PHP sessions are like short-term memory for websites. When you visit a website, it creates a session to remember you while you're on the site. It's like a temporary storage space for information specific to your visit, such as your login status or items in your shopping cart. Sessions help websites provide a personalized experience during your visit, and they forget everything once you leave.

PHP Session Start

The session_start() function is used to start a new or resume an existing session. It returns the existing session if it already exists.

Syntax:

```
bool session_start()
```

Example:

```
session_start();
```

PHP $_SESSION

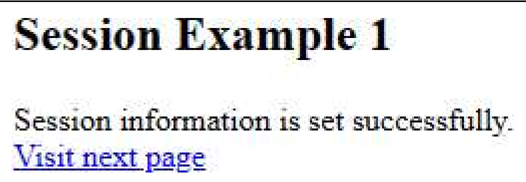
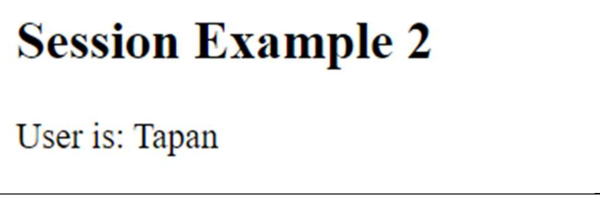
The $_SESSION superglobal is an associative array used to set and get session variable values.

Session1.php

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<head>
    <title>Session 1</title>
</head>
<body>
<h2>Session Example 1</h2>
<?php
$_SESSION["user"] = "Tapan";
echo "Session information is set successfully.<br/>";
?>
<a href="session2.php">Visit next page</a>
</body>
</html>
```

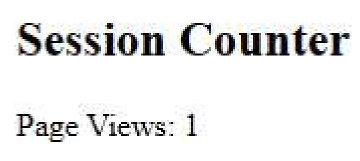
Session2.php

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<head>
    <title>Session 2</title>
</head>
<body>
<h2>Session Example 2</h2>
<?php
echo "User is: " . $_SESSION["user"];
?>
</body>
</html>
```

Output:	 **Session Example 1** Session information is set successfully. Visit next page  **Session Example 2** User is: Tapan

Code:

```
<?php
session_start();
if (!isset($_SESSION['counter'])) {
    $_SESSION['counter'] = 1;
} else {
    $_SESSION['counter']++;
}
?>
<!DOCTYPE html>
<html>
<head>
    <title>Session Counter</title>
</head>
<body>
<h2>Session Counter</h2>
<?php
echo "Page Views: " . $_SESSION['counter'];
?>
</body>
</html>
```

Output:	 **Session Counter** Page Views: 1

Destroying a Session

You can destroy all session variables completely using session_destroy().

Code:

```php
<?php
session_start();
session_destroy();
echo "Session Destroed";
?>
```

Output:

```
Session Destroed
```

PHP File Handling

PHP file handling is how websites deal with files. They can open files, read from them, write to them, and even delete them. It's like how you work with documents on your computer but done by the website's code.

PHP Open File - fopen()

The fopen() function is used to open a file.

Code:

```php
<?php
$filename = "example.txt";
$file = fopen($filename, "r");

if ($file) {
    echo "File $filename opened successfully.";
    fclose($file);
} else {
    echo "Failed to open the file.";
}
?>
```

Output:

```
File example.txt opened successfully.
```

PHP Read File - fread()

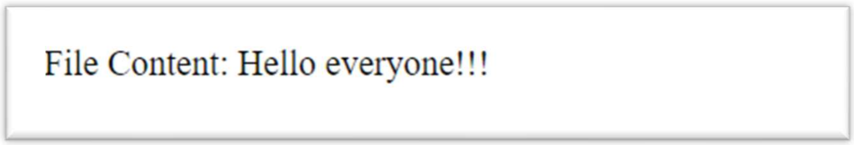
The fread() function is used to read the content of a file.

Code:

```php
<?php
$filename = "example.txt";

if (file_exists($filename)) {
    $fileContent = file_get_contents($filename);
    echo "File content:\n" . $fileContent;
} else {
    echo "File does not exist.";
}
?>
```

Output:



```
File Content: Hello everyone!!!
```

PHP Write File - fwrite()

The fwrite() function is used to write the content of a string into a file.

Code:

```php
<?php
$filename = "example.txt";
$content = "Good Bye!";
```

```php
$file = fopen($filename, "w");

if ($file) {
    fwrite($file, $content);
    fclose($file);
    echo "Content has been written to the file.";
} else {
    echo "Failed to open the file for writing.";
}
?>
```

Output:

```
Content has been written to the file.
```

PHP Close File - fclose()

The fclose() function is used to close an open file pointer.

Code:

```php
<?php
$filename = "example.txt";
$file = fopen($filename, "r");

if ($file) {
    fclose($file);
    echo "File closed successfully.";
} else {
    echo "Failed to open the file.";
}
?>
```

Output:

```
File closed successfully.
```

PHP Delete File - unlink()

The unlink() function is used to delete a file.

Code:

```php
<?php
$filename = "example.txt";

if (file_exists($filename)) {
    if (unlink($filename)) {
        echo "File deleted successfully.";
    } else {
        echo "Failed to delete the file.";
    }
} else {
    echo "File does not exist.";
}
?>
```

Output:

```
File deleted successfully.
```

PHP File Upload

PHP allows you to upload single and multiple files through few lines of code only.

PHP file upload features allows you to upload binary and text files both. Moreover, you can have the full control over the file to be uploaded through PHP authentication and file operation functions.

PHP $_FILES

The $_FILES global contains all the information about a file being uploaded, including its name, type, size, temp name, and errors.

$_FILES['filename']['name'] : returns file name.

$_FILES['filename']['type']: returns MIME type of the file.

$_FILES['filename']['size'] : returns size of the file (in bytes).

$_FILES['filename']['tmp_name'] : returns temporary file name of the file which was stored on the server.

$_FILES['filename']['error'] : returns error code associated with this file

move_uploaded_file() function

The move_uploaded_file() function moves the uploaded file to a new location. The move_uploaded_file() function checks internally if the file is uploaded thorough the POST request. It moves the file if it is uploaded through the POST request.

Syntax

bool move_uploaded_file (string $filename , string $destination)

PHP File Upload Example

Uploadfile.html

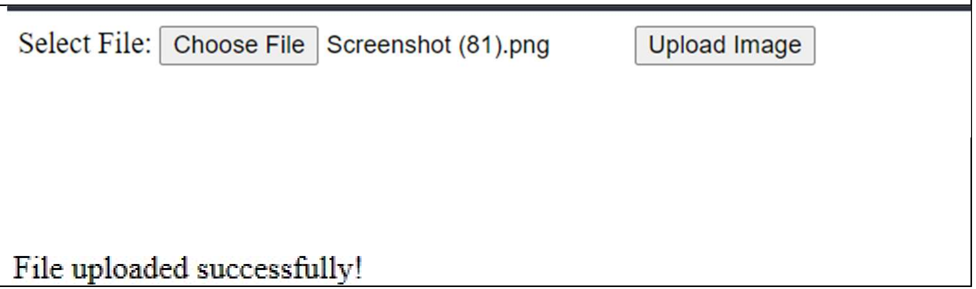
```
<!DOCTYPE html>
<html>
<head>
    <title>File Upload Example</title>
</head>
<body>
    <form action="uploadfiles.php" method="post"
enctype="multipart/form-data">
        Select File:
        <input type="file" name="fileToUpload" />
        <input type="submit" value="Upload Image"
name="submit" />
    </form>
</body>
</html>
```

Uploadfiles.php

```
<?php
$uploadDirectory = "C:/xampp/htdocs/web-tech-lab/File
Handling/Files";
```

```php
if (isset($_POST["submit"]) &&
isset($_FILES["fileToUpload"])) {
    $targetPath = $uploadDirectory .
basename($_FILES["fileToUpload"]["name"]);
    if (file_exists($targetPath)) {
        echo "Sorry, the file already exists. Please rename
the file and try again.";
    } else {
        if
(move_uploaded_file($_FILES["fileToUpload"]["tmp_name"],
$targetPath)) {
            echo "File uploaded successfully!";
        } else {
            echo "Sorry, there was an error uploading your
file. Please try again.";
        }
    }
} else {
    echo "Please select a file to upload.";
}
?>
```

Output:



```
File uploaded successfully!
```

Conclusion:

In this manual, our focus has been on delving into the realm of web technology, with a primary emphasis on harnessing the power of PHP to craft dynamic and captivating web pages. PHP stands out as a formidable scripting language, enabling web developers to create a myriad of compelling features and functionalities for websites. Its widespread popularity and open-source nature have cemented its status as the go-to choice for a multitude of web development projects.

Throughout this guide, we've journeyed through the essential building blocks of PHP. We began by acquainting ourselves with the fundamentals, understanding what PHP is and how it facilitates the creation of engaging web content. Subsequently, we explored the realm of variables, learning how to store and manipulate data efficiently. We also delved into decision-making processes and repetitive actions through if-else statements and loops. Furthermore, we delved into the world of data organization with arrays and data retrieval from web forms. Additionally, we addressed the intricacies of managing user data with cookies and sessions, which enhance web experiences by preserving user-specific information. Finally, we explored file handling, mastering the techniques of reading, writing, and manipulating data files in the web development context. These topics collectively equip you with the necessary knowledge to create feature-rich and secure web pages using PHP.

References:

https://www.w3schools.com/php/