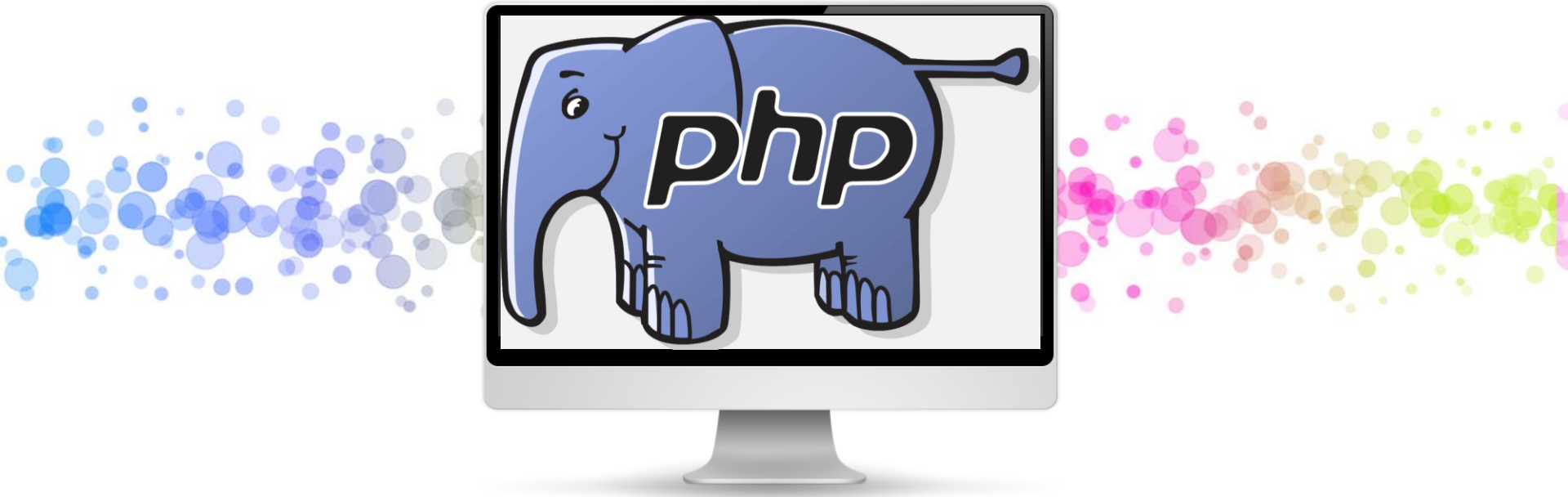
php

Topics

01 Form Handling

02 Form Validation

Form Handling

Capturing Form Data with PHP

Superglobal	Description
$_GET	Contains a list of all the field names and values sent by a form u sing the get method (i.e. via the URL parameters).
$_POST	Contains a list of all the field names and values sent by a form u sing the post method (data will not visible in the URL).
$_REQUEST	Contains the values of both the $_GET and $_POST variables a s well as the values of the $_COOKIE superglobal variable.

Form Validation

An HTML form contains various input fields such as text box, checkbox, radio buttons, subm it button, and checklist, etc. These input fields need to be validated, which ensures that the user has entered information in all the required fields and also validates that the information provided by the user is valid and correct.

There is no guarantee that the information provided by the user is always correct. PHP validates the data at the server-side, which is submitted by HTML form.
You need to validate a few things:

1. Empty String
2. Validate String
3. Validate Numbers
4. Validate Email
5. Input length etc

1. Empty String

```
if (empty ($_POST["name"])) {
    $errMsg = "Error! You didn't enter the Name.";
    echo $errMsg;
} else {
    $name = $_POST["name"];
}
```

2. Validate String

```
$name = $_POST ["Name"];
if (!preg_match ("/^[a-zA-z]*$/", $name) ) {
    $ErrMsg = "Only alphabets and whitespace are allowed.";
            echo $ErrMsg;
} else {
    echo $name;
}
```

3.Validate Number

```
$mobileno = $_POST ["Mobile_no"];
if (!preg_match ("/^[0-9]*$/", $mobileno) ){
    $ErrMsg = "Only numeric value is allowed.";
    echo $ErrMsg;
} else {
    echo $mobileno;
}
```

4. Validate Email

```
$email = $_POST ["Email"];
$pattern = "^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,
3})$^";
if (!preg_match ($pattern, $email) ){
    $ErrMsg = "Email is not valid.";
            echo $ErrMsg;
} else {
    echo "Your valid email address is: " .$email;
}
```

5.Input Length Validation

```
$mobileno = strlen ($_POST ["Mobile"]);
$length = strlen ($mobileno);  //9

if ( $length==10) {
 echo "Your Mobile number is: " .$mobileno;
 } else {
   $ErrMsg = "Mobile must have 10 digits.";
            echo $ErrMsg;
}
```

6.Button Click Validate

```
if (isset ($_POST['submit']) {
    echo "Submit button is clicked.";
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        echo "Data is sent using POST method ";
    }
} else {
    echo "Data is not submitted";
}
```

PHP Filters

Validating data = Determine if the data is in proper form.
Sanitizing data = Remove any illegal character from the data.

The PHP filter extension has many of the functions needed for checking user input, and is designed to make data validation easier and quicker.
The filter_list() function can be used to list what the PHP filter extension offers:

```
<table>
  <tr>
    <td>Filter Name</td>
    <td>Filter ID</td>
  </tr>
  <?php
  foreach (filter_list() as $id =>$filter) {
    echo '<tr><td>' . $filter . '</td><td>' . filter_id($filter) . '</td></tr>';
  }
  ?>
</table>
```

Why Use Filters?

Many web applications receive external input. External input/data can be:

- ❑User input from a form
- ❑Cookies
- ❑Web services data
- ❑Server variables
- ❑Database query results

PHP filter_var() Function :

The **filter_var()** function both validate and sanitize data.

The **filter_var()** function filters a single variable with a specified filter.

It takes two pieces of data:

- ❖The variable you want to check
- ❖The type of check to use

Sanitize a String

Example:

```
<?php
$str = "<h1>Hello World!</h1>";
$newstr = filter_var($str, FILTER_SANITIZE_STRING);
echo $newstr;
?>
```

O/P Hello World!

Validate an Integer

Example:

```
<?php
$int = 100;

if (!filter_var($int, FILTER_VALIDATE_INT) === false) {
  echo("Integer is valid");
} else {
  echo("Integer is not valid");
}
?>
```

O/P Integer is valid

Sanitize and Validate an Email Address

Example:

```
<?php
$email = “example@gmail.com";

// Remove all illegal characters from email
$email = filter_var($email, FILTER_SANITIZE_EMAIL);

// Validate e-mail
if (!filter_var($email, FILTER_VALIDATE_EMAIL) === false) {
  echo("$email is a valid email address");
} else {
  echo("$email is not a valid email address");
}
?>
```

O/P example@gmail.com is a valid email address

Sanitize and Validate a URL

Example:

```
<?php
$url = "https://www.opentechz.com";

// Remove all illegal characters from a url
$url = filter_var($url, FILTER_SANITIZE_URL);

// Validate url
if (!filter_var($url, FILTER_VALIDATE_URL) === false) {
  echo("$url is a valid URL");
} else {
  echo("$url is not a valid URL");
}
?>
```

O/P https://www.opentechz.com is a valid URL

Topics

01 Cookies

02 Sessions

What is a Cookie

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 retri eve to personalize the page when user visit the website next time.

Note: Each time the browser requests a page to the server, all the data in the cookie is automatically sent to the server within the request.

Setting a Cookie in PHP

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. The basic syntax of this function can be given with:

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

Parameter	Description
name	The name of the cookie
value	The value of the cookie. Do not store sensitive information since this value is stored on the user's computer.
Expires	The expiry date in UNIX timestamp format. After this time cookie will become inaccessible. The default value is 0.
Path	Specify the path on the server for which the cookie will be available. If set to /, the cookie will be available within the entire domain.
Domain	Specify the domain for which the cookie is available to e.g www.example.com.
Secure	This field, if present, indicates that the cookie should be sent only if a secure HTTPS connection exists.

Example

Here's an example that uses setcookie() function to create a cookie named **username** and assign the value value john to it. It also specify that the cookie will expire after 30 days **(30 days * 24 hours * 60 min * 60 sec).**

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

Note 1: If the expiration time of the cookie is set to 0, or omitted, the co okie will expire at the end of the ses sion i.e. when the browser closes.

Note 2: All the arguments except the name are op tional. You may also replace an argument with an empty string ("") in order to skip that argument, ho wever to skip the expire argument use a zero (0) instead, since it is an integer.

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"];
 ?>
```

O/P – John

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

Note:
You can use the print_r() function like print_r($_COOKIE);
to see the structure of this $_COOKIE associative array, like you with other arrays.

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);
?>
```

Tip: You should pass exactly the same path, domain, and other arguments that you have used when you first created the cookie in order to ensure that the correct cookie is deleted.

What is a Session

Although you can store data using cookies but it has some security issue s. Since cookies are stored on user's computer it is possible for an attack er to easily modify a cookie content to insert potentially harmful data in y our 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 i n size, the browser needs to upload 20KB of data each time the user vie ws a page, which can affect your site's performance.

You can solve both of these issues by using the PHP session. A PHP ses sion stores data on the server rather than user's computer. In a session b ased environment, every user is identified through a unique number calle d session identifier or SID. This unique session ID is used to link each us er with their own information on the server like emails, posts, etc.

Starting a PHP Session

Before you can store any information in session variables, you must first start up the session. To begin a new session, simply call the PHP session_start() function. It will create a new ses sion and generate a unique session ID for the user.

Example :

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

The session_start() function first checks to see if a session already exists by looking for the p resence of a session ID. If it finds one, i.e. if the session is already started, it sets up the sess ion variables and if doesn't, it starts a new session by creating a new session ID.

Note: . You must call the session_start() function at the beginning of the page i.e. be fore any output generated by your script in the browser, much like you do while setti ng the cookies with setcookie() function

Storing and Accessing Session Data

You can store all your session data as key-value pairs in the $_SESSION[] superglo bal array. The stored data can be accessed during lifetime of a session. Consider the following script, which creates a new session and registers two session variables.

Example:

```
<?php
// Starting session session_start();
 // Storing session data
$_SESSION["firstname"] = “Bibhu Ranjan";
 $_SESSION["lastname"] = “Mohanty";
 ?>
<?php
// Starting session  if page is different
session_start();
// Accessing session data echo 'Hi, ' . $_SESSION["firstname"] . ' ' . $_SESSION["lastname"];
 ?>
```

Note: To access the session data in the same page there is no need to recreate the session since it has been already started on the top of the page.

Destroying a Session

If you want to remove certain session data, simply unset the corresponding key of the **$_SESSION** associative array, as shown in the following example:

Example:1

```
<?php
 // Starting session
 session_start();
 // Removing session data
 if(isset($_SESSION["lastname"])){
 unset($_SESSION["lastname"]);
 }
 ?>
```

Example:2

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

Note: Before destroying a session with the session_destroy() function, you need to firs t recreate the session environment if it is not already there using the session_start() function, so that there is something to destroy.

Setting Default Time

```
<?php
// Start the session
session_start();

// Set the session timeout (in seconds)
$session_lifetime = 1800; // 30 minutes
ini_set('session.gc_maxlifetime', $session_lifetime);

// Set the session cookie lifetime
setcookie(session_name(), session_id(), time() + $session_lifetime);

// Your session data
$_SESSION['username'] = 'JohnDoe';
?>
```

Modifying php.ini

- session.gc_maxlifetime = 1800 ; // Time in seconds (e.g., 1800 = 30 minutes)
- session.cookie_lifetime = 1800 ; // Time in seconds (this ensures that the session cookie lasts for the same durati

Topics

01 Date and Time

02 Include Files

INFORMATIONS

The PHP **date()** function convert a timestamp to a more readable date and time. The computer stores dates and times in a format called UNIX Timestamp, which measures time as a number of seconds since the beginning of the Unix epoch (midnight Greenwich Mean Time on January 1, 2024 i.e. January 1, 2024 00:00:00 GMT).

Syntax : date(*format*, *timestamp*)

Note:*timestamp* is an optional parameter

Example:

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

Formatting the Dates and Times with PHP

The format parameter of the **date()** function is in fact a string that can contain multiple characters allowing you to generate a date string containing various components of the date and time, like day of the week, AM or PM, etc.

Commonly Used Format…..

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 2024)
l - (lowercase 'L') - Represents the day of the week

The parts of the date can be separated by inserting other characters, like hyphens (-), dots (.), slashes (/), or spaces to add additional visual formatting .

Formatting the Dates and Times with PHP

Similarly you can use the following characters to format the time string:

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)

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

PHP time() Function

The time() function is used to get the current time as a Unix timestamp (the number of seconds since the beginning of the Unix epoch: January 1 1970 00:00:00 GMT).

```
<?php
$timestamp = time();
 echo($timestamp);
?>
```

We can convert this timestamp to a human readable date through passing it to the pr eviously introduce date() function.

```
<?php
$timestamp = 1394003958;
echo(date("F d, Y h:i:s", $timestamp));
 ?>
```

PHP mktime() Function

The **mktime()** function is used to create the timestamp based on a specific date and time. If no date and time is provided, the timestamp for the current date and time is returned.

The syntax of the mktime() function can be given with:

mktime(*hour*, *minute*, *second*, *month*, *day*, *year*)

Example

```
<?php
// Create the timestamp for a particular date
echo mktime(15, 20, 12, 5, 10, 2014);
?>
```

The mktime() function can also be used to find a particular date in future after a specific time period. As in the following example, which displays the date which falls on after 30 month from the current date?

```
<?php
 $futureDate = mktime(0, 0, 0, date("m")+30, date("d"), date("Y"));
echo date("d/m/Y", $futureDate);
?>
```

Date From a String With strtotime()

The PHP **strtotime()** function is used to convert a human readable date string into a Unix timestamp (the number of seconds since January 1 1970 00:00:00 GMT).

Syntax

strtotime(*time, now*)

```
<?php
        $d= strtotime ("10:30pm April 15 2014");
        echo "Created date is " . date("Y-m-d h:i:sa", $d);
        $d= strtotime ("tomorrow");
        echo date("Y-m-d h:i:sa", $d) . "<br>";
        $d= strtotime ("next Saturday");
        echo date("Y-m-d h:i:sa", $d) . "<br>";
        $d= strtotime("+3 Months");
        echo date("Y-m-d h:i:sa", $d) . "<br>";
?>
```

Including a PHP File into Another PHP File

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.

You can save a lot of time and work through including files — Just store a block of code in a separate file and include it wherever you want using the **include()** and **require()** statements instead of typing the entire block of code multiple times.

Syntax:

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

Example

//main.php

```
<?php require "my_variables.php"; ?>
<?php require "my_functions.php"; ?>
 <!DOCTYPE html>
<html lang="en">
        <head>
         <title><?php displayTitle($home_page); ?></title>
         </head>
        <body>
        <?php include "header.php"; ?>
         <?php include "menu.php"; ?>
        <h1>Welcome to Our Website!</h1>
        <p>Here you will find lots of useful information.</p>
        <?php include "footer.php";
        ?>
        </body>
</html>
```

Difference between require() and include():

include()	require()
The include() function does not stop the execution of the script even if any error occurs.	The require() function will stop the execution of the script when an error occurs.
The include() function does not give a fatal error.	The require() function gives a fatal error
The include() function is mostly used when the file is not required and the application should continue to execute its process when the file is not found.	The require() function is mostly used when the file is mandatory for the application.
The include() function will only produc e a warning (E_WARNING) and the s cript will continue to execute.	The require() will produce a fatal error (E_COMPILE_ERROR) along with the warning.

include_once and require_once Statements

If you accidentally include the same file (typically functions or classes files) more tha n one time within your code using the include or require statements, it may cause co nflicts. To prevent this situation, PHP provides **include_once** and **require_once** stat ements. These statements behave in the same way as include and require statemen ts with one exception.

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

Syntax:

include_once("path/to/filename"); -Or- include_once "path/to/filename";
require_once("path/to/filename"); -Or- require_once "path/to/filename";

Example

//multiplication.php

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

//main.php

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

//main.php

```
<?php
// Including file
require_once " multiplication.php";
// Calling the function multiplySelf(2);
// Output: 4 echo "<br>";
// Including file once again
require_once " multiplication.php";
// Calling the function
multiplySelf(5); // Output: 25
?>
```

File System

Parsing Directory

File Upload

Working with Files in PHP

Since PHP is a server side programming language, it allows you to work wit h files and directories stored on the web server. In this session you will le arn how to create, access, and manipulate files on your web server using t he PHP file system functions.

Opening a File with PHP fopen() Function

To work with a file you first need to open the file. The PHP fopen() function i s used to open a file. The basic syntax of this function can be given with:

Syntax: fopen(*filename*, *mode*)

Example :

```
<?php
$handle = fopen("data.txt", "r");
?>
```

Differents Mode

Modes	What it does
r	**Open a file for read only**. File pointer starts at the beginning of the file.
r+	**Open a file for read/write**. File pointer starts at the beginning of the file.
w	**Open a file for write only**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file.
w+	**Open a file for read/write**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file.
a	**Open a file for write only**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist.
a+	**Open a file for read/write**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist.
x	**Creates a new file for write only**. Returns FALSE and an error if file already exists.
x+	**Creates a new file for read/write**. Returns FALSE and an error if file already exists.

File Exist

If you try to open a file that doesn't exist, PHP will generate a warning mes sage. So, to avoid these error messages you should always implement a simple check whether a file or directory exists or not before trying to access it, with the PHP file_exists() function.

Example :

```
<?php
  $file = "data.txt";
  // Check the existence of file
  if(file_exists($file)){
  // Attempt to open the file
  $handle = fopen($file, "r");
   } else{
  echo "ERROR: File does not exist.";
  }
  ?>
```

Reading from Files with PHP fread()

The fread() function can be used to read a specified number of characters fro m a file. The basic syntax of this function can be given with.

Syntax: fread(*file handle*, *length in bytes*)

```
<?php
$file = "data.txt";
// Check the existence of file
 if( file_exists($file)){
        // Open the file for reading
       $handle = fopen($file, "r") or die("ERROR: Cannot open the file.");
       $content _fixed= fread($handle, "20"); // Read fixed number of bytes from the file
       $content_entire = fread($handle, filesize($file)); // Read entired file
        fclose($handle); // Closing the file handle
       // Display the file content
       echo $content_ fixed;
       echo $content_entire;
} else{
echo "ERROR: File does not exist.";
}
?>
```

2. The easiest way to read the entire contents of a file in PHP is with the **readfile( )** function.

This function allows you to read the contents of a file without needing to open it. T he following example will generate the same output as above example:

```
<?php
        $file = "data.txt";
         // Check the existence of file
         if(file_exists($file)){
        // Reads and outputs the entire file
        readfile($file) or die("ERROR: Cannot open the file.");
        } else{
         echo "ERROR: File does not exist.";
        }
?>
```

3. Another way to read the whole contents of a file without needing to open it is with t he **file_get_contents()** function. This function accepts the name and path to a file, an d reads the entire file into a string variable. Here's an example:

```
<?php
        $file = "data.txt";
                // Check the existence of file
         if(file_exists($file)){
                // Reading the entire file into a string
                $content = file_get_contents($file) or die("ERROR: Cannot open the file.");
                // Display the file content echo $content;
        } else{
                 echo "ERROR: File does not exist.";
         }
?>
```

4. One more method of reading the whole data from a file is the PHP's file() function. It does a
similar job to file_get_contents() function, but it returns the file contents as an array o f lines,
rather than a single string.

```
<?php
 $file = "data.txt";
// Check the existence of file
        if(file_exists($file)){
                        // Reading the entire file into an array
                         $arr = file($file) or die("ERROR: Cannot open the file.");
                  foreach($arr as $line){
                 echo $line;
                 }
          } else{
         echo "ERROR: File does not exist.";
          } ?>
```

Writing the Files Using PHP write

1. Similarly, you can write data to a file or append to an existing file using the PHP fwrite() function. The basic syntax of this function can be given with:

Syntax:fwrite(*file handle*, *string*)

```
<?php
    $file = "note.txt";
     // String of data to be written
     $data = "The quick brown fox jumps over the lazy dog.";
     // Open the file for writing
     $handle = fopen($file, "w") or die("ERROR: Cannot open the file.");
     // Write data to the file
    fwrite($handle, $data) or die ("ERROR: Cannot write the file.");
     // Closing the file handle
    fclose($handle);
    echo "Data written to the file successfully.";
 ?>
```

Writing the Files Using PHP write

2. An alternative way is using the file_put_contents() function. It is counterpart of **file_get_contents()** function and provides an easy method of writing the data to a file without needing to open it.This function accepts the name and path to a file together with the data to be written to the file.

If the file specified in the file_put_contents() function already exists, PHP will overwrite it by default. If you would like to preserve the file's contents you can pass the special FILE_APPEND flag as a third parameter to the file_put_contents() function. It will simply append the new data to the file instead of overwitting it.

```
<?php
    $file = "note.txt";
    // String of data to be written
     $data = "The quick brown fox jumps over the lazy dog.";
     // Write data to the file
    file_put_contents($file, $data) or die("ERROR: Cannot write the file.");
    file_put_contents($file, $data, FILE_APPEND) or die("ERROR: Cannot write the file.");
    echo "Data written to the file successfully.";
?>
```

Renaming Files with PHP rename()

You can rename a file or directory using the PHP's rename() function, like this:

```
<?php
        $file = "file.txt";
        // Check the existence of file
            if(file_exists($file)){
                // Attempt to rename the file
                if(rename($file, "newfile.txt")){
                echo "File renamed successfully.";
                } else{
                echo "ERROR: File cannot be renamed.";
                 }
              } else{
             echo "ERROR: File does not exist.";
 }
?>
```

Removing Files with PHP unlink()

You can delete files or directories using the PHP's unlink() function, like this:

```
<?php
    $file = "note.txt";
     // Check the existence of file
    if(file_exists($file)){
        // Attempt to delete the file
        if(unlink($file)){
        echo "File removed successfully.";
         } else{
        echo "ERROR: File cannot be removed.";
        }
     } else{
     echo "ERROR: File does not exist.";
    }
?>
```

Creating a New Directory

You can create a new and empty directory by calling the PHP mkdir() function with th e path and name of the directory to be created, as shown in the example below:

```
<?php
    // The directory path
    $dir = "testdir";
    // Check the existence of directory
    if(!file_exists($dir)){
        // Attempt to create directory
        if(mkdir($dir)){
         echo "Directory created successfully.";
         } else{
        echo "ERROR: Directory could not be created.";
         }
    } else{
     echo "ERROR: Directory already exists.";
    }
?>
```

Copying Files from One Location to Another

You can copy a file from one location to another by calling PHP copy() function with t he file's source and destination paths as arguments. If the destination file already ex ists it'll be overwritten. Here's an example which creates a copy of "example.txt" file i nside backup folder.

```
<?php
// Source file path
 $file = "example.txt";
 // Destination file path $newfile = "backup/example.txt";
// Check the existence of file
 if(file_exists($file)){
        // Attempt to copy file
        if(copy($file, $newfile)){
         echo "File copied successfully.";
        } else{
         echo "ERROR: File could not be copied.";
        }
} else{
 echo "ERROR: File does not exist.";
} ?>
```

PHP Filesystem Functions

Function	Description
fgetc()	Reads a single character at a time.
fgets()	Reads a single line at a time.
fgetcsv()	Reads a line of comma-separated values.
filetype()	Returns the type of the file.
feof()	Checks whether the end of the file has been reached.
is_file()	Checks whether the file is a regular file.
is_dir()	Checks whether the file is a directory.
is_executable()	Checks whether the file is executable.
realpath()	Returns canonicalized absolute pathname.
mkdir()	Make Directory
rmdir()	Removes an empty directory. …& more

Topics

01 File Upload

02 File Download

Uploading Files with PHP

In this section we will learn how to upload files on remote server using a Simple
HTML form and PHP. You can upload any kind of file like images, videos, ZIP files,
Microsoft Office documents, PDFs, as well as executables files and a wi de range of
other file types.

File Uploaded with 2 Steps:

- Step 1: Creating an HTML form to upload the file
- Step 2: Processing the uploaded file

Step 1: Creating an HTML form to upload the file

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>File Upload Form</title>
</head>
<body>
<form action="upload_action.php" method="post" enctype="multipart/form-data“ >
    <div class="row">
        <input type="file" name="image" required>
        <input type="submit" name="submit" value="Upload">
    </div>
    <p>
     <strong>Note:</strong> Only .jpg, .jpeg, .gif, .png formats allowed to a max size of 5 MB.
    </p>
</form>
</body>
</html>
```

Step 2: Processing the uploaded file

```
<?php
$target_dir = "uploads/";
   echo $target_file = $target_dir . basename($_FILES["image"]["name"]);
   $post_tmp_img = $_FILES["image"]["tmp_name"];
   $imageFileType = strtolower( pathinfo ($target_file,PATHINFO_EXTEN
SION));
   $post_imag = $_FILES["image"]["name"];
      move_uploaded_file($post_tmp_img,"uploads/$post_imag");
?>
```

Key Points

1. Form should contain **POST** and **enctype=“multipart/ formdata”**
   Example:
   <form action="upload-manager.php" method="post" enctype="multipart/form-data">
2. Once the form is submitted information about the uploaded file can be accessed via PHP superglobal array called **$_FILES**.
   For example, our upload form contains a file select field called photo (i.e. name="photo")
3. **$_FILES["photo"]["name"]** — This array value specifies the original name of th e file, including the file extension. It doesn't include the file path.
4. **$_FILES["photo"]["type"]** — This array value specifies the MIME type of the fil e.
5. **$_FILES["photo"]["size"]** — This array value specifies the file size, in bytes.
6. **$_FILES["photo"]["tmp_name"]** — This array value specifies the temporary na me including full path that is assigned to the file once it has been uploaded to the s erver.
7. **$_FILES["photo"]["error"]** — This array value specifies error or status code as sociated with the file upload, e.g. it will be 0, if there is no error.

Downloading Files with PHP

Normally, you don't necessarily need to use any server side scripting language lik e PHP to download images, zip files, pdf documents, exe files, etc. If such kind of file is stored in a public accessible folder, you can just create a hyperlink pointing to that file, and whenever a user click on the link, browser will automatically downl oads that file.

Example :

```
<a href="downloads/test.zip">Download Zip file</a>

<a href="downloads/masters.pdf">Download PDF file</a>

<a href="downloads/sample.jpg">Download Image file</a>

<a href="downloads/setup.exe">Download EXE file</a>
```

Downloading Files with PHP

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Simple Image Gallery</title>
<style type="text/css">
    .img-box{
        display: inline-block;
        text-align: center;
        margin: 0 15px;
    }
</style>
</head>
<body>
    <?php
    // Array containing sample image file names
    $images = array("kites.jpg", "balloons.jpg");

    // Loop through array to create image gallery
    foreach($images as $image){
        echo '<div class="img-box">';
            echo '<img src="/examples/images/' . $image . '" width="200" alt="' .  pathinfo($image, PATHINFO_FILENAME) .'"
>';
            echo '<p><a href="/examples/images/’ . urlencode($image) . '“ download>Download</a></p>';
        echo '</div>';
    }
    ?>
</body>
</html>
```

Today's Topics

01 Date and Time

02 Include Files

INFORMATIONS

The PHP **date()** function convert a timestamp to a more readable date and time. The computer stores dates and times in a format called UNIX Timestamp, which measures time as a number of seconds since the beginning of the Unix epoch (midnight Greenwich Mean Time on January 1, 2024 i.e. January 1, 2024 00:00:00 GMT).

Syntax : date(*format*, *timestamp*)

Note:*timestamp* is an optional parameter

Example:

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

Formatting the Dates and Times with PHP

The format parameter of the **date()** function is in fact a string that can contain multiple characters allowing you to generate a date string containing various components of the date and time, like day of the week, AM or PM, etc.

Commonly Used Format…..

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 2024)
l - (lowercase 'L') - Represents the day of the week

The parts of the date can be separated by inserting other characters, like hyphens (-), dots (.), slashes (/), or spaces to add additional visual formatting .

Formatting the Dates and Times with PHP

Similarly you can use the following characters to format the time string:

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)

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

PHP time() Function

The time() function is used to get the current time as a Unix timestamp (the number of seconds since the beginning of the Unix epoch: January 1 1970 00:00:00 GMT).

```
<?php
$timestamp = time();
 echo($timestamp);
?>
```

We can convert this timestamp to a human readable date through passing it to the pr eviously introduce date() function.

```
<?php
$timestamp = 1394003958;
echo(date("F d, Y h:i:s", $timestamp));
 ?>
```

PHP mktime() Function

The **mktime()** function is used to create the timestamp based on a specific date and time. If no date and time is provided, the timestamp for the current date and time is returned.

The syntax of the mktime() function can be given with:

mktime(*hour*, *minute*, *second*, *month*, *day*, *year*)

Example

```
<?php
// Create the timestamp for a particular date
echo mktime(15, 20, 12, 5, 10, 2014);
?>
```

The mktime() function can also be used to find a particular date in future after a specific time period. As in the following example, which displays the date which falls on after 30 month from the current date?

```
<?php
 $futureDate = mktime(0, 0, 0, date("m")+30, date("d"), date("Y"));
echo date("d/m/Y", $futureDate);
?>
```

Date From a String With strtotime()

The PHP **strtotime()** function is used to convert a human readable date string into a Unix timestamp (the number of seconds since January 1 1970 00:00:00 GMT).

Syntax

strtotime(*time, now*)

```
<?php
        $d=strtotime("10:30pm April 15 2014");
        echo "Created date is " . date("Y-m-d h:i:sa", $d);
        $d=strtotime("tomorrow");
        echo date("Y-m-d h:i:sa", $d) . "<br>";
        $d=strtotime("next Saturday");
        echo date("Y-m-d h:i:sa", $d) . "<br>";
        $d=strtotime("+3 Months");
        echo date("Y-m-d h:i:sa", $d) . "<br>";
?>
```

Including a PHP File into Another PHP File

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.

You can save a lot of time and work through including files — Just store a block of code in a separate file and include it wherever you want using the **include()** and **require()** statements instead of typing the entire block of code multiple times.

Syntax:

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

Example

```
//main.php
<?php require "my_variables.php"; ?>
<?php require "my_functions.php"; ?>
<!DOCTYPE html>
<html lang="en">
<head>
<title><?php displayTitle($home_page); ?></title>
</head>
<body>
<?php include "header.php"; ?>
<?php include "menu.php"; ?>
<h1>Welcome to Our Website!</h1>
<p>Here you will find lots of useful information.</p>
<?php include "footer.php";
?>
</body> </html>
```

Difference between require() and include():

include()	require()
The include() function does not stop the execution of the script even if any error occurs.	The require() function will stop the execution of the script when an error occurs.
The include() function does not give a fatal error.	The require() function gives a fatal error
The include() function is mostly used when the file is not required and the application should continue to execute its process when the file is not found.	The require() function is mostly used when the file is mandatory for the application.
The include() function will only produce a warning (E_WARNING) and the script will continue to execute.	The require() will produce a fatal error (E_COMPILE_ERROR) along with the warning.

include_once and require_once Statements

If you accidentally include the same file (typically functions or classes files) more tha n one time within your code using the include or require statements, it may cause co nflicts. To prevent this situation, PHP provides **include_once** and **require_once** stat ements. These statements behave in the same way as include and require statemen ts with one exception.

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

Syntax:

include_once("path/to/filename"); -Or- include_once "path/to/filename";
require_once("path/to/filename"); -Or- require_once "path/to/filename";

Example

//multiplication.php

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

//main.php

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

//main.php

```
<?php
// Including file
require_once " multiplication.php";
// Calling the function multiplySelf(2);
// Output: 4 echo "<br>";
// Including file once again
require_once " multiplication.php";
// Calling the function
multiplySelf(5); // Output: 25
?>
```

File System

Parsing Directory

File Upload

Working with Files in PHP

Since PHP is a server side programming language, it allows you to work wit h files and directories stored on the web server. In this session you will le arn how to create, access, and manipulate files on your web server using t he PHP file system functions.

Opening a File with PHP fopen() Function

To work with a file you first need to open the file. The PHP fopen() function i s used to open a file. The basic syntax of this function can be given with:

Syntax: fopen(*filename*, *mode*)

Example :

```
<?php
$handle = fopen("data.txt", "r");
?>
```

Differents Mode

Modes	What it does
r	**Open a file for read only**. File pointer starts at the beginning of the file.
r+	**Open a file for read/write**. File pointer starts at the beginning of the file.
w	**Open a file for write only**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file.
w+	**Open a file for read/write**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file.
a	**Open a file for write only**. The existing data in file is preserved. File pointer starts at the e nd of the file. Creates a new file if the file doesn't exist.
a+	**Open a file for read/write**. The existing data in file is preserved. File pointer starts at the e nd of the file. Creates a new file if the file doesn't exist.
x	**Creates a new file for write only**. Returns FALSE and an error if file already exists.
x+	**Creates a new file for read/write**. Returns FALSE and an error if file already exists.

File Exist

If you try to open a file that doesn't exist, PHP will generate a warning messa ge. So, to avoid these error messages you should always implement a simpl e check whether a file or directory exists or not before trying to access it, wit h the PHP file_exists() function.

Example :

```
<?php
  $file = "data.txt";
  // Check the existence of file
  if(file_exists($file)){
  // Attempt to open the file
  $handle = fopen($file, "r");
   } else{
  echo "ERROR: File does not exist.";
  }
  ?>
```

Reading from Files with PHP fread()

The fread() function can be used to read a specified number of characters fro m a file. The basic syntax of this function can be given with.

Syntax: fread(*file handle*, *length in bytes*)

```
<?php
$file = "data.txt";
// Check the existence of file
 if(file_exists($file)){
        // Open the file for reading
       $handle = fopen($file, "r") or die("ERROR: Cannot open the file.");
       $content _fixed= fread($handle, "20"); // Read fixed number of bytes from the file
       $content_entire = fread($handle, filesize($file)); // Read entired file
        fclose($handle); // Closing the file handle
       // Display the file content
       echo $content_ fixed;
       echo $content_entire;
} else{
echo "ERROR: File does not exist.";
}
?>
```

2. The easiest way to read the entire contents of a file in PHP is with the **readfile( )** function.

This function allows you to read the contents of a file without needing to open it. T he following example will generate the same output as above example:

```
<?php
        $file = "data.txt";
         // Check the existence of file
         if(file_exists($file)){
        // Reads and outputs the entire file
        readfile($file) or die("ERROR: Cannot open the file.");
        } else{
         echo "ERROR: File does not exist.";
        }
?>
```

3. Another way to read the whole contents of a file without needing to open it is with t he **file_get_contents()** function. This function accepts the name and path to a file, an d reads the entire file into a string variable. Here's an example:

```
<?php
        $file = "data.txt";
                // Check the existence of file
         if(file_exists($file)){
                // Reading the entire file into a string
                $content = file_get_contents($file) or die("ERROR: Cannot open the file.");
                // Display the file content echo $content;
        } else{
                 echo "ERROR: File does not exist.";
         }
?>
```

4. One more method of reading the whole data from a file is the PHP's file() function. It does a
similar job to file_get_contents() function, but it returns the file contents as an array o f lines,
rather than a single string.

```
<?php
 $file = "data.txt";
// Check the existence of file
        if(file_exists($file)){
                        // Reading the entire file into an array
                         $arr = file($file) or die("ERROR: Cannot open the file.");
                  foreach($arr as $line){
                 echo $line;
                 }
          } else{
         echo "ERROR: File does not exist.";
          } ?>
```

Writing the Files Using PHP write

1. Similarly, you can write data to a file or append to an existing file using the PHP fwrite() function. The basic syntax of this function can be given with:

Syntax:fwrite(*file handle*, *string*)

```
<?php
    $file = "note.txt";
     // String of data to be written
     $data = "The quick brown fox jumps over the lazy dog.";
     // Open the file for writing
     $handle = fopen($file, "w") or die("ERROR: Cannot open the file.");
     // Write data to the file
    fwrite($handle, $data) or die ("ERROR: Cannot write the file.");
     // Closing the file handle
    fclose($handle);
    echo "Data written to the file successfully.";
 ?>
```

Writing the Files Using PHP write

2. An alternative way is using the file_put_contents() function. It is counterpart of **file_get_contents()** function and provides an easy method of writing the data to a file without needing to open it.This function accepts the name and path to a file together with the data to be written to the file.

If the file specified in the file_put_contents() function already exists, PHP will overwrite it by default. If you would like to preserve the file's contents you can pass the special FILE_APPEND flag as a third parameter to the file_put_contents() function. It will simply append the new data to the file instead of overwitting it.

```
<?php
    $file = "note.txt";
    // String of data to be written
     $data = "The quick brown fox jumps over the lazy dog.";
     // Write data to the file
    file_put_contents($file, $data) or die("ERROR: Cannot write the file.");
    file_put_contents($file, $data, FILE_APPEND) or die("ERROR: Cannot write the file.");
    echo "Data written to the file successfully.";
?>
```

Renaming Files with PHP rename()

You can rename a file or directory using the PHP's rename() function, like this:

```
<?php
        $file = "file.txt";
        // Check the existence of file
            if(file_exists($file)){
                // Attempt to rename the file
                if(rename($file, "newfile.txt")){
                echo "File renamed successfully.";
                } else{
                echo "ERROR: File cannot be renamed.";
                 }
              } else{
             echo "ERROR: File does not exist.";
 }
?>
```

Removing Files with PHP unlink()

You can delete files or directories using the PHP's unlink() function, like this:

```
<?php
    $file = "note.txt";
     // Check the existence of file
    if(file_exists($file)){
        // Attempt to delete the file
        if(unlink($file)){
        echo "File removed successfully.";
         } else{
        echo "ERROR: File cannot be removed.";
        }
     } else{
     echo "ERROR: File does not exist.";
    }
?>
```

Creating a New Directory

You can create a new and empty directory by calling the PHP mkdir() function with th e path and name of the directory to be created, as shown in the example below:

```
<?php
    // The directory path
    $dir = "testdir";
    // Check the existence of directory
    if(!file_exists($dir)){
        // Attempt to create directory
        if(mkdir($dir)){
         echo "Directory created successfully.";
         } else{
        echo "ERROR: Directory could not be created.";
         }
    } else{
     echo "ERROR: Directory already exists.";
    }
?>
```

Copying Files from One Location to Another

You can copy a file from one location to another by calling PHP copy() function with t he file's source and destination paths as arguments. If the destination file already ex ists it'll be overwritten. Here's an example which creates a copy of "example.txt" file i nside backup folder.

```
<?php
// Source file path
 $file = "example.txt";
 // Destination file path $newfile = "backup/example.txt";
// Check the existence of file
 if(file_exists($file)){
        // Attempt to copy file
        if(copy($file, $newfile)){
         echo "File copied successfully.";
        } else{
         echo "ERROR: File could not be copied.";
        }
} else{
 echo "ERROR: File does not exist.";
} ?>
```

PHP Filesystem Functions

Function	Description
fgetc()	Reads a single character at a time.
fgets()	Reads a single line at a time.
fgetcsv()	Reads a line of comma-separated values.
filetype()	Returns the type of the file.
feof()	Checks whether the end of the file has been reached.
is_file()	Checks whether the file is a regular file.
is_dir()	Checks whether the file is a directory.
is_executable()	Checks whether the file is executable.
realpath()	Returns canonicalized absolute pathname.
mkdir()	Make Directory
rmdir()	Removes an empty directory. …& more

Topics

01 File Upload

02 File Download

Uploading Files with PHP

In this section we will learn how to upload files on remote server using a Simple
HTML form and PHP. You can upload any kind of file like images, videos, ZIP files,
Microsoft Office documents, PDFs, as well as executables files and a wi de range of
other file types.

File Uploaded with 2 Steps:

- Step 1: Creating an HTML form to upload the file
- Step 2: Processing the uploaded file

Step 1: Creating an HTML form to upload the file

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>File Upload Form</title>
</head>
<body>
<form action="upload_action.php" method="post" enctype="multipart/form-data“ >
    <div class="row">
        <input type="file" name="image" required>
        <input type="submit" name="submit" value="Upload">
    </div>
    <p>
     <strong>Note:</strong> Only .jpg, .jpeg, .gif, .png formats allowed to a max size of 5 MB.
    </p>
</form>
</body>
</html>
```

Step 2: Processing the uploaded file

```
<?php
$target_dir = "uploads/";
   echo $target_file = $target_dir . basename($_FILES["image"]["name"]);
   $post_tmp_img = $_FILES["image"]["tmp_name"];
   $imageFileType = strtolower( pathinfo ($target_file,PATHINFO_EXTEN
SION));
   $post_imag = $_FILES["image"]["name"];
      move_uploaded_file($post_tmp_img,"uploads/$post_imag");
?>
```

Key Points

1. Form should contain **POST** and **enctype=“multipart/ formdata”**
   Example:
   <form action="upload-manager.php" method="post" enctype="multipart/form-data">
2. Once the form is submitted information about the uploaded file can be accessed via PHP superglobal array called **$_FILES**.
   For example, our upload form contains a file select field called photo (i.e. name="photo")
3. **$_FILES["photo"]["name"]** — This array value specifies the original name of th e file, including the file extension. It doesn't include the file path.
4. **$_FILES["photo"]["type"]** — This array value specifies the MIME type of the fil e.
5. **$_FILES["photo"]["size"]** — This array value specifies the file size, in bytes.
6. **$_FILES["photo"]["tmp_name"]** — This array value specifies the temporary na me including full path that is assigned to the file once it has been uploaded to the s erver.
7. **$_FILES["photo"]["error"]** — This array value specifies error or status code as sociated with the file upload, e.g. it will be 0, if there is no error.

Downloading Files with PHP

Normally, you don't necessarily need to use any server side scripting language lik e PHP to download images, zip files, pdf documents, exe files, etc. If such kind of file is stored in a public accessible folder, you can just create a hyperlink pointing to that file, and whenever a user click on the link, browser will automatically downl oads that file.

Example :

```
<a href="downloads/test.zip">Download Zip file</a>

<a href="downloads/masters.pdf">Download PDF file</a>

<a href="downloads/sample.jpg">Download Image file</a>

<a href="downloads/setup.exe">Download EXE file</a>
```

Downloading Files with PHP

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Simple Image Gallery</title>
<style type="text/css">
    .img-box{
        display: inline-block;
        text-align: center;
        margin: 0 15px;
    }
</style>
</head>
<body>
    <?php
    // Array containing sample image file names
    $images = array("kites.jpg", "balloons.jpg");

    // Loop through array to create image gallery
    foreach($images as $image){
        echo '<div class="img-box">';
            echo '<img src="/examples/images/' . $image . '" width="200" alt="' .  pathinfo($image, PATHINFO_FILENAME) .'"
>';
            echo '<p><a href="/examples/images/’ . urlencode($image) . '“ download>Download</a></p>';
        echo '</div>';
    }
    ?>
</body>
</html>
```


Thank you