Module-1_Core_PHP

PHP Syntax

Que. Discuss the structure of PHP script and how to embed PHP in HTML.

- PHP script is always surrounded by <?php and ?>.
- Any logic or PHP script should be present between the starting and ending PHP tags.
- This is to make the system understand the start and end of the script.
- PHP can be easily embedded in an HTML using the PHP start and end tags.
- After PHP is embedded the file should be renamed as a '.php' file as file extension.

Que. What are the rules for naming variables in PHP?

- All PHP variables start with the `$` sign.
- First letter after `$` should be an alphabet or underscore.
- First letter should not be a number.
- Name of the variable may contain alphanumeric characters and underscore only.
- In PHP, variable names are case-sensitive so variable `$num` and `$NUM` are different.
- e.g., `$tommy`, `$page1`, `$p_24`, `$SPECIAL`, `$_999`

- **Lab Exercise : 01.**

Write a PHP script to print "Hello, World!" on a web page.

- File name : M1_LE_01.php

PHP variables

Que. Explain the concept of variables in PHP and their scope.

- In PHP, variables are containers used to store data values. They can hold different data types such as strings, integers, arrays, and objects.
- Variables always start with a dollar sign ($).
- The variable name must start with a letter or an underscore `_`.
- The variable name can only contain alphanumeric characters and underscores (`A-z, 0-9, _`).
- Variables in PHP are case-sensitive.
- We can directly assign a value to a variable and its datatype will be assigned automatically based on the provided value.

```
<?php
$NEW = 'Hello People';
$new = 'hello world';
$a = 25;
$b = 11.5;
 ?>
```

- Scope of a variable determines where in code variables can be accessed or modified. Variables in PHP can have three different scope: local, global and static.
- **Local Scope :** Variable declared inside a function have local scope and cannot be accessed outside of it. Local variables are destroyed once the function execution is completed.
- **Global Scope:** Variable declared outside any function are global variables. They can be accessed by functions with the help of `global` keyword or `$GLOBALS` array.

- **Static Scope :** Static variables retain their values across function calls. They need to be initialized once and retain their values even after function execution is completed.
- **Super global:** Super global variables are pre-defined in PHP and are accessible from any part of the script, regardless of scope.

- ## Lab Exercise : 02.

Create a PHP script to declare and initialize different types of variables (integer, float, string, boolean). Display them using echo.

- File name : M1_LE_02.php

Super Global variables

Que. What are super global variables in PHP? List atleast five super global arrays and their use.

- Super global variables in PHP are built-in, pre-defined arrays that are accessible in all scopes of a script (global, local, and static).
- These variables are automatically populated by the PHP runtime and are primarily used for accessing data such as user inputs, session information, server details, and file uploads.

1. **$_GET** - contains data sent via the URL query string using `get` method. It is used to retrieve data passed in a URL, commonly used in search forms and links

2. **$_POST** - Contains data sent via the HTTP POST method. It is used to handle form submissions securely, especially for sensitive data.

3. **$_SESSION** - Stores the session variables for the current user. These persist across pages during a session. It is used to store and maintain user login information and preferences for the session duration.

4. **$_COOKIE** - Contains data stored in cookies which are saved on user's browser. It is used to remember user preferences such as language or theme settings.

5. **$_SERVER** - Contains server and execution environment information such as headers, paths and script locations. It is used to access metadata about the request or server configuration.

Lab Exercise : 03.

Create a form that takes a user's name and email. Use the $_POST super global to display the entered data.

- File name : M1_LE_03.php

SQL Tables and SQL Queries

- ## Lab Exercise : 04.

Create multiple tables and perform queries using : SELECT, UPDATE, DELETE, INSERT, WHERE, LIKE, GROUP BY, HAVING, LIMIT, OFFSET, Subqueries, AND, OR, NOT

- File name M1_LE_04.php

Conditions, Events, and Flows

Que. Explain how conditional statements work in PHP

- Conditional statements in PHP allow you to execute certain blocks of code based on specific conditions.
- They are used to control the flow of a program by checking whether a condition is true or false.

Type of Conditional statements in PHP

1. if statement :

- if statement executes a code only if the specified condition is true.

```
$age = 20;
if ($age >= 18) {
    echo "You are eligible to drive.";
}
```

2. if-else statement :

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

```
$age = 16;
if ($age >= 18) {
    echo "You are eligible to drive.";
} else {
    echo "You are not eligible to drive.";
}
```

3. if-elseif-else statement :

- The `if-elseif-else` statement is used to test multiple conditions in sequence.

```
$marks = 75;
if ($marks >= 90) {
    echo "Grade: A+";
} elseif ($marks >= 75) {
    echo "Grade: A";
} elseif ($marks >= 50) {
    echo "Grade: B";
} else {
    echo "Grade: F";
}
```

4. switch statement :

- The `switch` statement is used to perform different actions based on the value of a variable. It is an alternative to multiple `if-elseif` conditions.

```
$day = "Monday";
switch ($day) {
    case "Monday":
        echo "Start of the work week!";
        break;
    case "Friday":
        echo "Weekend is near!";
        break;
    default:
        echo "It's a regular day.";
}
```

5. Ternary Operator :

- The ternary operator is a shorthand for the `if-else` statement.

```
$age = 20;
echo ($age >= 18) ? "Eligible to drive." : "Not eligible to drive.";
```

6. Null Coalescing (`??`) Operator :

- The null coalescing operator is used to check if a variable is set and not null. It is a shorthand for `isset()`.

```
$username = $_GET['username'] ?? "Guest";
echo "Welcome, $username!";
```

If Condition and If-else If

LAB EXERCISE 05:

Write a PHP program to determine if a number is even or odd using if conditions.

- File name M1_LE_05.php

Practical Example: Calculator and Day Finder

Lab Exercise : 06.

Simple Calculator : Create a calculator using if-else conditions that take two inputs and an operator (+, -, *, /).

- File name : M1_LE_06.php

Lab Exercise : 07.

Day Finder : Write a script that finds the current day. If it is Sunday, print "Happy Sunday".

- File name : M1_LE_07.php

Switch Case and Ternary Operator

Lab Exercise : 08.

Restaurant Food Category Program: Use a switch case to display the category (Starter/Main Course/Dessert) and dish based on user selection.

- File name : M1_LE_08.php

Lab Exercise : 09.

Ternary Operator Example: Write a script using the ternary operator to display a message if the age is greater than 18.

- File name : M1_LE_09.php

Lab Exercise : 10.

Color Selector: Write a program to display the name of a color based on user input (red, green, blue).

- File name : M1_LE_10.php

Loops: Do-While, For Each, For Loop

Que. Discuss the difference between for loop, for each loop, and do-while loop in PHP.

- In PHP, there are three common types of loops used to execute code repeatedly: `for` loop, `foreach` loop, and `do-while` loop.

1. for loop : The for loop is typically used when the number of iterations is known before entering the loop. It's ideal for iterating over a specific range or count. In for loop the initialization, condition and increment/decrement is stated in the same line.

```
// Print numbers from 1 to 5
for ($i = 1; $i <= 5; $i++) {
   echo $i . "<br>";
}
```

2. foreach loop : The `foreach` loop is specifically used for iterating over arrays. It simplifies the process of iterating through each element in an array without needing an index counter. It can also be used with key value pairs.

```
// Print each color in the array
$colors = ['Red', 'Green', 'Blue'];
foreach ($colors as $color) {
   echo $color . "<br>";
}
```

3. do-while loop : The `do-while` loop is similar to the `while` loop, but it guarantees that the code block will run at least once, because the condition is evaluated after the loop's code is executed.

```
// Print numbers from 1 to 5 using do-while loop
$i = 1;
do {
   echo $i . "<br>";
   $i++;
} while ($i <= 5);
```

Lab Exercise : 11.

For Loop : Write a script that displays numbers from 1 to 10 on a single line.

- File name : M1_LE_11.php

```
// Print numbers from 1 to 10
for ($i = 1; $i <= 10; $i++) {
   echo $i . ", ";
}
```

Lab Exercise : 12.

For Loop (Addition): Add all integers from 0 to 30 and display the total.

- File name : M1_LE_12.php

Lab Exercise : 13.

Chessboard Pattern: Use a nested loop to create a chessboard pattern (8x8 grid).

- File name : M1_LE_13.php

Lab Exercise : 14.

Various Patterns: Generate different patterns using loops.

- File name : M1_LE_14.php

PHP Array and Array Functions

Que. Define arrays in PHP. What are the different types of arrays?

Array is a data structure that stores multiple values in a single variable.
Arrays allow store, organize and manipulate related data easily and efficiently.
In PHP, arrays can dynamically resize and hold mixed data types.

Array can be defined in PHP using `array()` function and through `[]` syntax.

```
$animals = array("cat", "dog", "cow", "buffalo");
$flowers = ["rose", "lily", "tulip", "marigold"];
```

There are 3 types of arrays in PHP.

1. Indexed Arrays
2. Associative Arrays
3. Multi-dimensional Arrays

1.Indexed Arrays:

These arrays use numeric indices for storing values. The numbering starts from 0.

```
$animals = ["cat", "dog", "cow", "buffalo"];
echo $animals[0];  // Output : cat
```

2.Associative Arrays:

These arrays use keys (strings) instead of numeric indices to store value.

```
$car = [
   "name" : "Octavia",
   "brand" : "Skoda",
   "horsepower" : "188 bhp",
   "style" : "Sedan"
]
echo $car["name"];  // Output : Octavia
```

3.Multidimensional Arrays:

These arrays contain one or more arrays as their elements.
These are mostly used to create a matrix structure or a nested structure.

```
$foodChart = [
   ["Tea", "Coffee", "Milk"],
   ["Muffins", "Cookies", "Pastries"],
]
echo $foodChart[2][1]; // Output : Muffins
```

It is possible to combine indexed and associative arrays within the same structure.

Lab Exercise : 15.

Display the value of an array.

- File name M1_LE_15.php

Lab Exercise : 16.

Find and display the number of odd and even elements in an array.

- File name M1_LE_16.php

Lab Exercise : 17.

Create an associative array for user details (name, email, age) and display them.

- File name : M1_LE_17.php

Lab Exercise : 18.

Write a script to shift all zero values to the bottom of an array.

- File name : M1_LE_18.php

PHP Date-Time Function

Lab Exercise : 19.

Write a script to display the current date and time in different formats.

- File name : M1_LE_19.php

Header Function

Que. What is the header function in PHP and how is it used?

- The `header()` function in PHP is used to send raw HTTP headers to the browser before any output is sent.
- It allows you to control the headers that are sent to the client, enabling you to perform tasks like redirecting the user, setting content types, managing cache control, or customizing HTTP responses.
- No HTML or whitespace precede the header() function.
- `exit` function is used after header() function after redirection or file download.

Uses of header() function :

1. Redirecting to another page

```
header("Location: https://www.google.com");
exit;
```

2. Setting Content-type

```
header("Content-type: application/json");
```

3. Sending custom HTTP response codes

```
header("HTTP/1.1 404 Not Found");
echo "Page not found!";
```

4. File Downloads

```
header("Content-Description: File Transfer");
header("Content-Type: application/octet-stream");
header("Content-Disposition: attachment; filename=\"example.txt\"");
header("Content-Length: " . filesize("example.txt"));
readfile("example.txt");
exit;
```

Lab Exercise : 20

Redirect users to another page using the header() function.

- File name : M1_LE_20.php

Include and Require

Que. Explain the difference between include and require in PHP.

- The `include` and `require` statements are both used to include and execute contents of one PHP file into another.
- By using a PHP script through `include`, we get a warning message if the file is missing but the script is still executed.
- By using a PHP script through `require`, we get a fatal error message if the file is missing and stops execution of the script.
- The main difference is the way both statement handle errors.
- `include` statement is used when the file is optional or non-critical. `require` statement is used when the file is essential for the script to work properly.
- `include`: Non-critical. Issues a warning if the file is missing and continues execution.
- `require`: Critical. Issues a fatal error if the file is missing and halts execution.

Lab Exercise : 21

Calculator : Create a calculator using user-defined functions.

- File name : < M1_LE_21.php

Lab Exercise : 22.

Factorial: Write a function that finds the factorial of a number using recursion.

- File name : M1_LE_22.php

Lab Exercise : 23.

String Reverse: Reverse a string without using built-in functions.

- File name : M1_LE_23.php

Lab Exercise : 24

Download File: Create a button that allows users to download a file

- File name : M1_LE_24.php

PHP Expressions, Operations, and String Functions

Que. Explain what PHP expressions are and give examples of arithmetic and logical operations.

- In PHP, an expression is any valid combination of variables, operators, and values that evaluates to a single value.
- Expressions are used to perform operations, manipulate data, and return results.
- There are various types of expressions used in PHP :

1. Arithmetic Expressions
2. String Expressions
3. Comparison Expressions
4. Logical Expressions
5. Assignment Expressions
6. Increment/Decrement Expressions
7. Array Expressions
8. Ternary Expressions
9. Null Coalescing Expression

- Examples of arithmetic operations

```
$a = 10, $b = 20;
$add = $a + $b;
$sub = $a - $b;
$mul = $a * $b;
$div = $b / $a;
$a += $b;
```

- Examples of logical operations

```
$is_odd = true;
$is_prime = false;
$check1 = $is_odd && $is_prime;
$check2 = $is_odd || $is_prime;
```

Lab Exercise : 25

Write a script to perform various string operations like concatenation, substring extraction, and string length determination.

- File name : M1_LE_25.php

Extra LAB EXERCISES for Core PHP

1. PHP Syntax

Lab Exercise : 26

Write a PHP script that demonstrates the use of single-line (//), multi-line (/* */), and inline (#) comments.

- File name : M1_LE_26.php

Lab Exercise : 27

Embedding HTML and PHP: Create a web page that uses PHP to dynamically generate HTML content (e.g., a table with user information using PHP).

- File name : M1_LE_27.php

Lab Exercise : 28

Output Statements: Experiment with echo, print, and var_dump. Write a script that outputs different types of data using these functions.

- File name : M1_LE_28.php

2. PHP Variables

Lab Exercise : 29

Type Casting: Write a script that declares variables of different types and converts them into other types (e.g., integer to float, string to integer). Display the type and value before and after the conversion.

- File name : < M1_LE_29.php

Lab Exercise : 30

Variable Variables: Demonstrate the use of variable variables in PHP. Write a script where a variable name is stored in another variable, and then use it to print the value.

- File name : M1_LE_30.php

Lab Exercise : 31

Global and Local Scope: Write a script that shows how global and local variables work. Use the global keyword inside a function to access a global variable.

- File name : M1_LE_31.php

3. Super Global Variables

Lab Exercise : 32

$_GET` and $_POST : Create two separate forms: one that uses the $_GET method and one that uses $_POST. Display the difference in the URL and how data is passed.

- File name : M1_LE_32.php

Lab Exercise : 33

$_SERVER : Write a script to display various details of the server environment using $_SERVER (like PHP_SELF, SERVER_NAME, HTTP_USER_AGENT, etc.).

- File name : < M1_LE_33.php

Lab Exercise : 34

$_FILES : Create a form that allows users to upload a file. Handle the uploaded file using the $_FILES super global and display information about the file.

- File name : < M1_LE_34.php

4. Practical Example: Multiple Tables and SQL Queries

Lab Exercise : 35

Complex Joins : Create a PHP script that connects two or more tables using INNER JOIN, LEFT JOIN, and RIGHT JOIN. Display data from these tables based on specific conditions.

- File name : < M1_LE_35.php

Lab Exercise : 36

Prepared Statements : Implement SQL queries using prepared statements with placeholders to prevent SQL injection in SELECT, INSERT, UPDATE, and DELETE.

- File name M1_LE_36.php

Lab Exercise : 37

Transaction Management : Write a PHP script that uses SQL transactions to insert data into multiple tables, ensuring data integrity in case of an error.

- File name : < M1_LE_37.php

5. Conditions, Events, and Flows

Lab Exercise : 38

Nested Conditions : Write a script that uses nested if-else conditions to categorize a number as positive, negative, or zero, and also check if it's an even or odd number.

- File name : M1_LE_38.php

Lab Exercise : 39

Switch Case with Multiple Cases: Write a script that accepts a grade (A, B, C, D, F) and displays a message using a switch statement. Handle multiple cases that fall under the same logic (e.g., A and B show "Excellent").

- File name : M1_LE_39.php

6. If Condition and If-Else If

Lab Exercise : 40

Grading System: Write a PHP program that accepts a student's marks and outputs their grade using if-else conditions (A, B, C, D, Fail based on score).

- File name : M1_LE_40.php

Lab Exercise : 41

Temperature Converter: Write a script that takes temperature in Celsius or Fahrenheit as input and converts it to the other format using if conditions.

- File name : < M1_LE_41.php

7. Practical Example: Calculator and Day Finder

Lab Exercise : 42.

Enhanced Calculator: Modify the calculator to handle more complex operations such as exponentiation (^), modulus (%), and square root (√).

- File name : M1_LE_42.php

Lab Exercise : 43

Date Finder with Time Zone: Write a script that finds the current day and prints "Happy Sunday" if it's Sunday, but also adjusts for different time zones.

- File name : M1_LE_43.php

8. Switch Case and Ternary Operator

Lab Exercise : 44

Month Display: Create a program using switch case that takes a number (1-12) and displays the corresponding month.

- File name : M1_LE_44.php

Lab Exercise : 45

Discount Calculation (Ternary Operator): Write a script that calculates and displays the discount on a product based on a user-defined price. If the price is above 500, give a 10% discount; otherwise, no discount (use the ternary operator).

- File name : M1_LE_45.php

9. Loops: Do-While, For Each, For Loop

Lab Exercise : 46

FizzBuzz Program: Write a program using a for loop that prints numbers from 1 to 100. But for multiples of 3, print "Fizz" instead of the number, for multiples of 5 print "Buzz", and for multiples of both 3 and 5 print "FizzBuzz".

- File name : < M1_LE_46.php

Lab Exercise : 47

Multiplication Table: Write a PHP script using a nested for loop to generate a multiplication table from 1 to 10.

- File name : M1_LE_47.php

Lab Exercise : 48

Reverse Number Sequence: Write a script using a do-while loop that displays numbers from 10 to 1.

- File name M1_LE_48.php

10. PHP Array and Array Functions

Lab Exercise : 49

Sorting Arrays: Write a script that demonstrates the use of sort(), rsort(), asort(), and ksort() functions to sort arrays.

- File name M1_LE_49.php

Lab Exercise : 50

Multi-dimensional Array: Create a multi-dimensional array to store information about products (name, price, and stock). Write a script to display the information in a tabular format.

- File name : M1_LE_50.php

Lab Exercise : 51

Array Merge and Diff: Write a PHP script that merges two arrays and finds the difference between them using array_merge() and array_diff().

- File name : M1_LE_51.php

11. PHP Date-Time Function

Lab Exercise : 52

Time Difference: Write a script that calculates the time difference between two dates (e.g., "today" and "next birthday").

- File name : M1_LE_52.php

Lab Exercise : 53

Custom Date Formats: Create a script that displays the current date in different formats (e.g., Y-m-d, d/m/Y, l, F jS Y).

- File name : M1_LE_53.php

12. Header Function

Lab Exercise : 54

Page Redirect Based on Condition: Write a script that checks if a user is logged in (use a boolean variable). If not, use the header() function to redirect them to a login page.

- File name : M1_LE_54.php

Lab Exercise : 55

Content-Type Header: Write a script that sets the Content-Type header to return a plain text file or a JSON response.

- File name : M1_LE_55.php

13. Include and Require

Lab Exercise : 56

Template System: Write a PHP script that includes header, navigation, and footer files in multiple web pages to create a basic template system.

File name : M1_LE_56.php

Lab Exercise : 57

File Not Found Handling: Use require to include a critical file. If the file doesn't exist, display a custom error message instead of the default PHP error.

File name M1_LE_57.php

14. Practical Example: Calculator, Factorial, String Reverse

Lab Exercise : 58

Enhanced Factorial: Write a recursive and non-recursive function to calculate the factorial of a number. Compare their performance for large numbers.

- File name : M1_LE_58.php

Lab Exercise : 59

Palindrome Checker: Create a function that checks if a given string is a palindrome.

- File name : M1_LE_59.php

Lab Exercise : 60

File Upload: Create a form that allows users to upload a file. Upon submission, download the file using a button click and display the file’s details (name, type, size).

- File name : M1_LE_60.php

HTML, CSS and JS in PHP

1. HTML Basics

What is HTML? Explain its structure.

- HTML (HyperText Markup Language) is the standard language used to create and design web pages.
- It provides the structure and layout of a web page by defining elements like headings, paragraphs, links, images, and more.
- HTML is not a programming language but a markup language that uses tags to structure content.
- Basic structure of an HTML document

```
<!DOCTYPE html>
<html>
  <head>
    <title>Document Title</title>
  </head>
  <body>
    <h1>Welcome to HTML</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```

- <!DOCTYPE html> : Declares the document type and version of HTML (HTML5 in this case) and tells the browser how to interpret the HTML code.
- <html> : The root element of an HTML document. All HTML content is contained within this tag.
- <head> : Contains metadata (information about the page) that is not displayed directly on the webpage.
- Common elements inside <head>:
  1. <title> : Specifies the title of the webpage (displayed in the browser tab).
  2. <meta> : Provides metadata, like character encoding, viewport settings, or descriptions for search engines.
  3. <link> : Links external resources like CSS stylesheets.
  4. <script>: Includes or links to JavaScript files.
- <body>: Contains all the visible content of the web page, such as text, images, links, videos, and other elements.
- Example elements in the <body>:
  1. <h1> to <h6> : Headings of various levels (from largest to smallest).
  2. : Paragraphs.
  3. <a> : Hyperlinks.
  4. <img>: Images.
  5. and <ol>: Lists (unordered and ordered).

Describe the purpose of HTML tags and provide examples of commonly used tags.

- HTML tags are the building blocks of an HTML document.
- Tags define the structure and content of a webpage by specifying how elements should be displayed in a web browser.
- Tags are written in angle brackets (< >) and typically come in pairs: an opening tag and a closing tag (e.g., and). Some tags are self-closing (e.g., <img />).

Key Purposes of HTML Tags:

- Content Structuring: Organize content into headings, paragraphs, lists, tables, etc.
- Semantic Meaning: Provide meaning to the content, e.g., `<header>` for page headers, `<article>` for individual articles.
- Media Embedding: Insert images, videos, and audio into a webpage.
- Hyperlinking: Link to other pages or sections within the same page using `<a>` tags.
- Form Creation: Enable user input with forms, buttons, checkboxes, and other input types.

Commonly Used HTML Tags

1. Structural Tags

- <html>: The root element that contains all HTML content.
- <head>: Contains metadata, styles, and scripts.
- <body>: Contains the visible content of the webpage.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

2. Heading Tags

- Used to define headings of different levels. <h1>: Largest heading. <h6>: Smallest heading.

```
<h1>Main Heading</h1>
<h2>Subheading</h2>
<h3>Smaller Subheading</h3>
```

3. Paragraph and Text Formatting Tags

- <p>: Defines a paragraph.
- : Bold text for importance.
- : Italicized text for emphasis.
-
: Inserts a line break (self-closing).

```
<p>
  This is a paragraph with <strong>bold</strong> and <em>italicized</em> text.
</p>
<p>Line break below:<br />Second line.</p>
```

4. Link and Image Tags

- <a>: Creates hyperlinks.
- <img>: Displays an image (self-closing).

```
<a href="https://example.com">Visit Example</a>
<img src="image.jpg" alt="Description of image" />
```

5. List Tags

- : Unordered list (bulleted).
- : Ordered list (numbered).
- : List item.

```
<ul>
  <li>Apple</li>
  <li>Banana</li>
</ul>
<ol>
  <li>First Step</li>
```

```
  <li>Second Step</li>
</ol>
```

6. Table Tags

- <table>: Creates a table.
- <tr>: Table row.
- <td>: Table cell (data).
- <th>: Table header.

```
<table>
  <tr>
    <th>Name</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>John</td>
    <td>30</td>
  </tr>
</table>
```

7. Form Tags

- Used to collect user input.
- <form>: Form container.
- <input>: Input field (self-closing).
- <button>: Submit button.
- <textarea>: Multi-line text input.

```
<form action="submit.php" method="POST">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name" required />
  <button type="submit">Submit</button>
</form>
```

8. Media Tags

- <video>: Embeds a video.
- <audio>: Embeds audio.
- <iframe>: Embeds another webpage.

```
<video controls>
  <source src="video.mp4" type="video/mp4" />
</video>
<audio controls>
  <source src="audio.mp3" type="audio/mpeg" />
</audio>
```

What are the differences between block-level and inline elements? Give examples of each.

Differences Between Block-Level and Inline Elements

Block-level and inline elements are the two main categories of HTML elements, each with specific characteristics in terms of behavior, layout, and usage.

1. Block-Level Elements :

- **Occupy Full Width** : A block-level element always starts on a new line and occupies the full width of its container (parent element).

- **Stack Vertically** : Elements appear one below the other, creating a "block-like" structure.

- **Can Contain Other Elements** : Block-level elements can contain other block-level or inline elements.

- **Formatting** : Suitable for structuring content (e.g., headings, paragraphs, sections).
    - Common Block-Level Elements :
  - : Generic container for grouping content.
  - <p> : Paragraphs of text.
  - <h1> to <h6> : Headings of various levels.
  - <section> : Semantic section of a document.
  - <article> : Independent, self-contained content.
  - and <ol> : Lists (unordered and ordered).
  - <table> : Tables.

```
<div>
  <h1>This is a heading</h1>
  <p>This is a paragraph.</p>
  <section>
    <p>Content inside a section.</p>
  </section>
</div>
```

2. Inline Elements :

- **Do Not Start a New Line** : Inline elements stay in the same line as surrounding content unless wrapped to the next line by the container's width.

- **Occupy Only Necessary Space**: The width of an inline element is only as wide as its content.

- **Cannot Contain Block-Level Elements**: Inline elements can contain text or other inline elements but not block-level elements.

- **Formatting**: Primarily used for styling or modifying parts of text or content.
- Common Inline Elements :
  - : Generic container for styling text.
  - <a>: Hyperlinks.
  - : Bold text.
  - : Italicized text.
  - <img>: Images (self-closing).
  - <label>: Labels for form controls.
  - <code>: Code snippets.

```
<p>This is a paragraph with <strong>bold</strong> and <em>italic</em> text.</p>
<a href="https://example.com">Visit Example</a>
<span style="color: red;">This text is red.</span>
```

Inline Elements	Block Elements
Inline elements occupy only sufficient width required.	Block Elements occupy the full width irrespective of their sufficiency.

Inline Elements	Block Elements
Inline elements don’t start in a new line.	Block elements always start in a line.
Inline elements allow other inline elements to sit behind.	Block elements doesn’t allow other elements to sit behind
Inline elements don’t have top and bottom margin	Block elements have top and bottom margin.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Block vs Inline Elements</title>
  </head>
  <body>
    <!-- Block-Level Elements -->
    <div>
      <h1>Block-Level Element</h1>
      <p>This is a paragraph inside a block-level element.</p>
    </div>

    <!-- Inline Elements -->
    <p>
      This is a paragraph with <strong>bold text</strong>,
      <em>italicized text</em>, and an <a href="#">inline link</a>.
    </p>
  </body>
</html>
```

Explain the concept of semantic HTML and why it is important.

- Semantic HTML refers to the use of HTML tags that convey the meaning and structure of the content they enclose.
- These tags describe the purpose of the content, making it easier for both developers and browsers to understand and organize a webpage.
- Semantic HTML introduces meaningful tags (e.g., `<header>`, `<article>`, `<section>`) that provide additional context compared to generic tags like `<div>` and `<span>`.

Why Semantic HTML is required

Improves Readability for Developers :
- Semantic tags make the code more intuitive and self-explanatory.
- <nav> clearly indicates a navigation menu, while <header> denotes the top section of a webpage.

Enhances Accessibility :
- Screen readers and assistive technologies can interpret semantic tags to better convey the content's structure to visually impaired users.
- Using <main> helps screen readers skip repetitive navigation sections and focus on the main content.

SEO Benefits :

- Search engines use semantic tags to better understand the hierarchy and importance of content, improving search engine rankings.
- <article> signals to search engines that the content is a standalone piece of writing.

Future-Proof Code :

- Semantic HTML aligns with web standards, making websites more adaptable to future technologies and easier to maintain.

Consistency Across Browsers :

- Browsers are designed to recognize and style semantic elements consistently, providing a better user experience.

Examples of Semantic HTML Tags

1. Structural Tags :
   - `<header>`: Represents the header of a webpage or a section.
   - `<nav>`: Defines a navigation menu.
   - `<main>`: Specifies the main content of a document.
   - `<footer>`: Denotes the footer of a webpage or section.
   - `<section>`: Represents a thematic grouping of content.
   - `<article>`: Contains independent, self-contained content like blog posts or news articles.
   - `<aside>`: Contains content indirectly related to the main content (e.g., sidebars or ads).
2. Text-Level Tags :
   - `<strong>`: Indicates important text (usually bolded).
   - `<em>`: Denotes emphasized text (usually italicized).
   - `<blockquote>`: Represents a block of quoted text.
   - `<cite>`: Indicates a citation or reference.
3. Media Tags :
   - `<figure>`: Groups media elements (e.g., images, charts) with captions.
   - `<figcaption>`: Provides a caption for a `<figure>`.

2. CSS Fundamentals

What is CSS? How does it differ from HTML?

- CSS (Cascading Style Sheets) is a stylesheet language used to control the presentation and layout of HTML documents.
- It allows developers to define the look and feel of a webpage, including colors, fonts, spacing, layout, and responsiveness, separately from the structure and content defined by HTML.
- Features of CSS :
  - Separation of Content and Design : HTML defines the structure and content, while CSS defines the styling and presentation.
  - Flexibility : CSS allows applying styles to multiple elements, pages, or an entire website from a single file.
  - Cascading Nature : Stylesheets cascade, meaning rules can be applied in a layered manner, with later rules overriding earlier ones if necessary.

HTML	CSS
HTML is a markup language used to define a structure of a web page.	CSS is a style sheet language used to style the web pages by using different styling features.
It consists of tags inside which text is enclosed.	It consists of selectors and declaration blocks.
HTML doesn’t have further types.	CSS can be internal or external depending upon the requirement.
We cannot use HTML inside a CSS sheet.	We can use CSS inside an HTML document.

HTML	CSS
HTML is not used for presentation and visualization.	CSS is used for presentation and visualization.
HTML has comparatively less backup and support.	CSS has comparatively higher backup and support.
HTML doesn’t allow animations and transitions.	CSS allows animation and transitions which helps to improve the UI.
HTML files are saved with ***.htm*** or ***.html*** extension.	CSS files are saved with ***.css*** extension.

Explain the three ways to apply CSS to a web page.

- Three ways (types) to apply CSS to a web page are :
  1. Inline CSS : Applied directly to an HTML element using the `style` attribute.

  This is red text.
  2. Internal CSS : Defined within a `<style>` tag in the `<head>` section of an HTML document.

  <style>
  p {
    color: blue;
  }
  </style>
  3. Stored in an external `.css` file and linked using the `<link>` tag.
  <link rel="stylesheet" href="styles.css">
- There are advantages of External CSS over inline HTML styling :
  - A single CSS file can apply consistent styles across multiple pages.
  - Styles can be reused across different projects.
  - Keeps HTML structure free of inline styles, improving readability and maintainability.

What are CSS selectors? List and describe the different types of selectors.

- CSS selectors are patterns or rules used to "select" HTML elements that you want to style.
- They target elements in an HTML document based on attributes like tags, classes, IDs, or their relationships to other elements.
- Once a selector matches an element, you can apply styles to that element using CSS properties.
- Types of selectors :
  - Universal Selector (`*`)
    - Selects all elements on the page.
    - Syntax : `* { property : value }`
    - Example :

    * {
      margin: 0;
      padding: 0;
    }
  - Type Selector (Element Selector)
    - Selects all elements of a specific type (tag name).

- Syntax : `element { property : value }`
- Example :

```
p {
   color: blue;
}
```

What is the box model in CSS? Explain its components.

- The CSS Box Model is a fundamental concept in web design that describes how HTML elements are treated as rectangular boxes.
- It defines the structure and layout of elements on a webpage, including their size, padding, borders, and margins.
- When you style an element in CSS, you manipulate its box model properties to control how it is displayed and how it interacts with other elements.

Components of the Box Model
- The box model consists of four layers, from the innermost to the outermost:
- **Content** : The innermost part of the box where the actual text, image, or other content is displayed.
- Size can be controlled with the width and height properties.
- **Padding** : The space between the content and the border.
- Increases the distance between the content and the edges of the box.
  padding: 10px;
- **Border** : The edge surrounding the padding (or content, if padding is not set).
- Can be styled with properties like border-width, border-style, and border-color.
  border: 2px solid black;

- **Margin** : The outermost space that separates the box from neighboring elements.
- Does not have a background color or border.
  margin: 20px;
- Box Model example

```
div {
   width: 200px; /* Content width */
   height: 100px; /* Content height */
   padding: 10px; /* Space around content */
   border: 2px solid black; /* Border around padding */
   margin: 20px; /* Space outside the border */
}
```

3. Responsive Web Design

What is responsive web design? Why is it important?

- **Responsive Web Design** (RWD) is a design approach aimed at creating websites that adapt seamlessly to different screen sizes, resolutions, and devices.
- A responsive website adjusts its layout, content, and functionality based on the user's device (e.g., desktop, tablet, smartphone) to ensure an optimal viewing experience.
- Responsive web design is typically achieved using a combination of flexible layouts, media queries, and responsive media (e.g., images, videos).

Key Features of Responsive Web Design
- Fluid Grids : Use percentage-based widths instead of fixed units like pixels to create layouts that scale fluidly.
- Flexible Images : Ensure that images resize dynamically to fit within their containers using properties like max-width: 100%.
- Media Queries : CSS feature that applies specific styles based on the screen size or device characteristics.

```
@media (max-width: 768px) {
    body {
        font-size: 14px;
    }
}
```
- Mobile-First Design : Start designing for smaller screens and gradually enhance the design for larger devices.

Why is Responsive Web Design Important?

- Improves User Experience : A responsive website ensures a seamless and visually appealing experience, regardless of the device being used.
- Increases Mobile Traffic : With the rise of mobile usage, responsive design is critical to reach and retain mobile users.
- Better SEO : Search engines like Google prioritize mobile-friendly websites in search rankings. A responsive design boosts SEO and visibility.
- Cost-Effective : Instead of maintaining separate websites for desktop and mobile, a single responsive site reduces development and maintenance costs.
- Adaptability : Ensures compatibility with future devices, screen sizes, and resolutions.
- Faster Loading Times : By optimizing elements for smaller devices, responsive design can improve load times, enhancing user satisfaction.

Techniques Used in Responsive Design

1. Flexible Layouts : Use relative units like percentages (%) or viewport widths (vw, vh) to make the layout adapt to different screen sizes.
```
.container {
    width: 90%;
    max-width: 1200px;
    margin: 0 auto;
}
```
2. Media Queries : Apply different styles based on the screen's dimensions or capabilities.
```
@media (max-width: 600px) {
    .menu {
        display: none;
    }
}
```
3. Responsive Media : Resize images and videos dynamically to prevent them from overflowing their containers.
```
img {
    max-width: 100%;
    height: auto;
}
```
4. CSS Frameworks : Frameworks like Bootstrap and Tailwind CSS provide pre-built classes for responsive design, making it faster to implement.

Explain the use of media queries in CSS. Provide an example.

- Media Queries are a feature in CSS that allow developers to apply specific styles to a webpage based on the characteristics of the user's device, such as screen size, resolution, orientation, or color scheme.
- They enable responsive web design by tailoring styles for different devices, ensuring an optimal user experience.
- Syntax : A media query is written using the `@media` rule, followed by a condition (media feature) and a block of CSS rules to apply if the condition is met.

```
/* Basic Syntax */
@media (media-feature) {
    /* CSS styles to apply */
}

/* Example*/
@media (max-width: 768px) {
```

```
  body {
    font-size: 14px;
  }
}
```

- These are some commonly used media features :
1. max-width and min-width : Target screen widths.
`@media (max-width: 600px)` applies styles to screens smaller than or equal to 600px.
2. max-height and min-height : Target screen heights.
`@media (min-height: 800px)` applies styles to screens taller than 800px.
3. Orientation : Target screen orientation.
`@media (orientation: landscape)` applies styles when the device is in landscape mode.
4. aspect-ratio : Targets screens with a specific aspect ratio.
`@media (aspect-ratio: 16/9)` applies styles to 16:9 screens.
5. prefers-color-scheme : Detects light or dark mode preferences.
`@media (prefers-color-scheme: dark)` applies styles for dark mode.

What are the benefits of using a mobile-first approach in web design?

- The mobile-first approach in web design is a strategy where the design and development process starts with creating a website optimized for smaller screens (e.g., smartphones) and gradually expands to include larger screens (e.g., tablets, desktops).
- This approach prioritizes the user experience on mobile devices, which are now the most commonly used devices for browsing the web.
- It is implemented using responsive design techniques like CSS media queries, starting with styles for small screens and adding styles for larger screens.

Benefits of Using a Mobile-First Approach

- **Improved User Experience (UX)** : Mobile-first ensures the website is user-friendly on small screens, where space is limited.
- Key content and functionality are prioritized, creating a streamlined and intuitive experience.

- **Better Performance** : By focusing on small screens first, developers build lightweight, fast-loading pages with only essential elements.
- Additional resources for larger screens are loaded conditionally, improving overall performance.

- **Catering to the Majority of Users** : Mobile devices account for a significant portion of web traffic globally. Designing for mobile first ensures your website meets the needs of the majority.

- **Simplifies Progressive Enhancement** : The mobile-first approach builds a strong foundation by addressing constraints like limited screen space and bandwidth.
- Additional features and styles can then be progressively added for larger devices without sacrificing usability.

- **Improved Search Engine Optimization (SEO)** : Search engines like Google prioritize mobile-friendly websites in search rankings.
- A mobile-first approach ensures compliance with mobile usability requirements, improving visibility in search results.

- **Encourages Prioritization of Content** : Limited screen space on mobile devices forces developers to focus on the most critical content.
- This leads to cleaner designs and ensures the content hierarchy is clear.

- **Cost and Time Efficiency** : Designing for mobile first can save time and resources since additional styles for larger screens are added incrementally.
- Reduces the risk of creating unnecessary features that don't add value to smaller screen users.

- **Future-Proof Design** : The mobile-first approach ensures compatibility with the growing variety of small-screen devices like smartphones, tablets, and wearables.
- As mobile usage continues to grow, mobile-first design ensures longevity and relevance.

- **Better Accessibility** : Mobile-first designs often emphasize accessibility features like readable font sizes, touch-friendly buttons, and simple navigation, which benefit all users.

- **Easier Maintenance** : A mobile-first approach results in cleaner, modular code, making it easier to maintain and update the website in the future.

4. PHP Integration

How can PHP be used to dynamically generate HTML content? Provide examples.

- PHP is a server-side scripting language designed to embed dynamic content within HTML pages.

- It allows developers to create HTML content based on dynamic data (e.g., user input, database queries, or API responses) and send the resulting HTML to the client.

- Benefits of using PHP for dynamic HTML :
  1. **Personalized Content** : Tailor pages based on user preferences or input.

  2. **Database Integration** : Fetch data from a database and display it as HTML.

  3. **Reusability** : Use loops and functions to avoid hardcoding repetitive HTML structures.

  4. **Interactivity** : Adjust the content dynamically based on user actions, query parameters, or conditions.

- Example : display greeting according to time.

```
<?php
// Set timezone
date_default_timezone_set("Asia/Kolkata");

// Get the current hour
$hour = date("H");

// Determine the greeting
if ($hour < 12) {
    $greeting = "Good Morning!";
} elseif ($hour < 18) {
    $greeting = "Good Afternoon!";
} else {
    $greeting = "Good Evening!";
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Dynamic Greeting</title>
</head>
<body>
    <h1><?php echo $greeting; ?></h1>
```

```
    <p>The current time is <?php echo date("h:i A"); ?>.</p>
</body>
</html>
```

Explain how to include CSS files in a PHP-generated HTML page.

To include CSS files in a PHP-generated HTML page, you use standard HTML `<link>` or `<style>` tags within your PHP-generated output.

- CSS is used to style dynamically generated content in PHP.
- Link the CSS file in the `<head>` section of your HTML page:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Include CSS in PHP</title>
    <!-- Link to an external CSS file -->
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
    <h1>Welcome to My Website</h1>
    <p>This is styled using an external CSS file.</p>
</body>
</html>
```

- Ensure the CSS file is in the same directory as your PHP file or an appropriate relative path (e.g., `css/styles.css`).
- If necessary, dynamically generate the CSS using PHP (e.g., `styles.php`) instead of a static `.css` file.
- You can include CSS inside the `<style>` tag in the `<head>` section of a PHP script.

```
<?php
// Dynamic data
$bgColor = "#f0f8ff"; // AliceBlue
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Inline CSS in PHP</title>
    <style>
        body {
            background-color: <?php echo $bgColor; ?>;
            font-family: Arial, sans-serif;
        }
        h1 {
            color: #333;
        }
        p {
            color: #555;
        }
    </style>
</head>
```

```
<body>
    <h1>Welcome!</h1>
    <p>This page uses inline CSS defined in PHP.</p>
</body>
</html>
```

- This method is useful for small amounts of CSS or when styles need to be generated dynamically.
- You can include different CSS files or inline styles based on a condition (e.g., user preferences, themes, or pages).
- A PHP file can generate CSS dynamically and be linked as an external stylesheet.
- PHP can be embedded inside HTML for inline styles, but this is not recommended for larger applications.

What are the advantages of using PHP to manage HTML forms?

The advantages of using PHP to manage HTML forms include:

- **Dynamic Handling**: PHP allows processing form data dynamically, making web applications interactive.
- **Data Validation** : Forms can be validated on the server-side to ensure input accuracy and security.
- **Security** : PHP can sanitize and escape user inputs to prevent security vulnerabilities like SQL injection and XSS attacks.
- **Session & Cookie Management:** PHP enables handling user sessions and cookies to maintain login states and preferences.
- **Database Integration** : Form data can be stored and retrieved from databases using PHP.
- **Cross-Browser Compatibility** : Since PHP runs on the server, form handling remains consistent across different browsers.
- **File Upload Processing** : PHP allows handling file uploads efficiently with the `$_FILES` superglobal.
- **Custom Error Handling** : PHP provides error-handling mechanisms to display meaningful messages to users.

Lab Exercise : 61

1. Creating a Simple Web Page
   - Objective: Create a basic web page using HTML and style it with CSS.
     - Instructions:
       - Create an HTML file (e.g., index.html) that includes a header, a navigation bar, a main content section, and a footer.
       - Style the page using an external CSS file (e.g., styles.css).
       - Use CSS properties such as color, background-color, font-size, and padding to enhance the design.

       - File name : M1_LE_61.php

Lab Exercise : 62

2. Form Handling with PHP
   - Objective: Create a simple HTML form and process it using PHP.
     - Instructions:
       - Create an HTML form that collects user information (e.g., name, email, and message).
       - Use PHP to process the form data and display a confirmation message with the submitted information.
       - Validate user inputs and provide appropriate feedback.

       - File name : M1_LE_62.php

Lab Exercise : 63

3. Dynamic Content Generation
   - Objective: Use PHP to generate dynamic HTML content.
     - Instructions:
       - Create a PHP script (e.g., dynamic-content.php) that generates a list of items (e.g., products or blog posts) from an array.
       - Use a loop to display the items in a styled HTML list.
       - Style the list using CSS.

       - File name : M1_LE_63.php

Lab Exercise : 64

4. CSS Grid and Flexbox
- Objective: Create a responsive layout using CSS Grid or Flexbox.
- Instructions:
  - Build a grid layout for a gallery of images or a product showcase using either CSS Grid or Flexbox.
  - Ensure that the layout is responsive and adjusts based on the screen size.
  - Use media queries to change the layout for mobile devices.

- File name : M1_LE_64.php

Lab Exercise : 65

5. Styling a PHP Application
- Objective: Apply CSS styles to a PHP web application.
- Instructions:
  - Create a simple PHP application (e.g., a user registration page).
  - Use an external CSS file to style the form elements (e.g., inputs, buttons, labels).
  - Ensure that the application is visually appealing and user-friendly.

- File name : M1_LE_65.php

Lab Exercise : 66

6. Implementing a Responsive Navigation Bar
- Objective: Create a responsive navigation bar using HTML and CSS.
- Instructions:
  - Build a navigation bar using HTML `<ul>` and `<li>` elements.
  - Use CSS to style the navigation bar and make it responsive (e.g., using media queries).
  - Implement a dropdown menu for sub-navigation items.

- File name : M1_LE_66.php

Lab Exercise : 67

7. Image Gallery with Lightbox Effect
- Objective: Create an image gallery that opens images in a lightbox effect.
- Instructions:
  - Use HTML to create a gallery of images.
  - Implement CSS for styling and layout.
  - Use JavaScript or a CSS library to create a lightbox effect when images are clicked.

- File name : M1_LE_67.php