### Module 6 – HTML, CSS and JS in PHP

### HTML Basics

1. **What is HTML? Explain its structure.**

* HTML, short for HyperText Markup Language, is the foundation of every webpage. It defines the structure and content of web pages, telling **browsers how to display text, images, links, and other elements. The basic** structure of an HTML document follows a specific format with essential elements.
* **structure:**

<!DOCTYPE html>

<html>

<head>

<title> </title>

</head>

<body>

<h1> Heading</h1>

<p> paragraph </p>

</body>

</html>

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

* HTML tags are the **building blocks of webpages that tell browsers how to display content**. They provide structure and organization by defining elements like headings, paragraphs, links, and images. Think of them as instructions for the browser on how to render the web page.
* **Purpose of HTML Tags:**
* **Structure:** HTML tags define the different sections and elements within a document, creating a hierarchy.
* **Meaning:** They assign meaning to content, allowing browsers and other tools to understand what a specific piece of content represents (e.g., a heading, a paragraph, a link).
* **Display:** They dictate how content should be displayed on the web page, including formatting, layout, and appearance.
* **Commonly Used HTML Tags:**

1. <html>: The root element of an HTML page, enclosing all other elements.
2. <head>: Contains metadata about the document, such as the title, character set, and links to stylesheets or scripts.
3. <body>: Contains the visible content of the web page.
4. <title>: Defines the title of the document, which is displayed in the browser tab or title bar.
5. <h1> - <h6>: Defines heading levels.
6. <p>: Defines a paragraph.
7. <a>: Defines a hyperlink.
8. <img>: Defines an image.
9. <ul>: Defines an unordered list.
10. <ol>: Defines an ordered list.
11. <li>: Defines a list item.
12. <b>: Displays text in bold.
13. <i>: Displays text in italics.
14. **What are the differences between block-level and inline elements? Give examples of each.**

* Block-level elements are HTML elements that always start on a new line and take up the full width available on the page, while inline elements flow within the same line as surrounding content.
  1. **Block-Level Elements:**
* Starts on a new line: Block elements begin a new line on the page.
* Takes up full width: They fill the available width of their parent container.
* Margins: They can have margins, and margins between adjacent block elements may collapse.
* Examples: Paragraph (<p>), heading (<h1>), division (<div>), list (<ul>), <li>, <img>.
* Can contain other elements: Block elements can contain both inline and block elements.
  1. **Inline Elements:**
* Does not start on a new line: Inline elements flow within the same line as other elements.
* Only takes up necessary width: They only occupy the width needed to display their content.
* Cannot have margins: Inline elements do not have margins.
* Examples: Span (<span>), link (<a>), image (<img>), button (<button>), <em>, <strong>.
* Cannot contain a block element: Inline elements generally cannot contain block-level elements.

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

* Semantic HTML involves using HTML tags that accurately describe the meaning and purpose of web content, rather than just its presentation. This approach makes web pages more accessible, easier to understand for both users and search engines, and ultimately enhances the overall user experience.
* **Why is semantic HTML important?**
* **Improved Accessibility:**

Semantic HTML helps assistive technologies like screen readers understand the structure and context of a page, making it easier for people with disabilities to navigate and use the web.

* **Enhanced Search Engine Optimization (SEO):**

Search engines use crawlers to index web pages, and semantic HTML helps them better understand the content and context of a page. This can lead to better search rankings, making it easier for users to find your content.

* **Better User Experience:**

By providing clear structure and meaning, semantic HTML can make a webpage more intuitive and easier for users to navigate.

### **CSS Fundamentals**

* 1. **What is CSS? How does it differ from HTML?**
* CSS stands for Cascading Style Sheets
* CSS describes how HTML elements are to be displayed on screen, paper, or in other media
* CSS saves a lot of work. It can control the layout of multiple web pages all at once
* External stylesheets are stored in CSS files
* **Explain the three ways to apply CSS to a web page.**
* css have 3 types
  1. inline use in direct in tag <p style="PROPERTY:VALUE">
  2. internal in head/body part <style> ... </style>
  3. external create new file like mystyle.css

then link in all pages in head load

<head>

<link href="mystyle.css" rel="stylesheet">

</head>

* 1. **What are CSS Selectors?** **List and describe the different types of selectors.**
* CSS **selectors** are patterns used to **select and style HTML elements**. They allow you to target specific elements (or groups of elements) in the DOM and apply styles to them.
* **Types of CSS Selectors:**

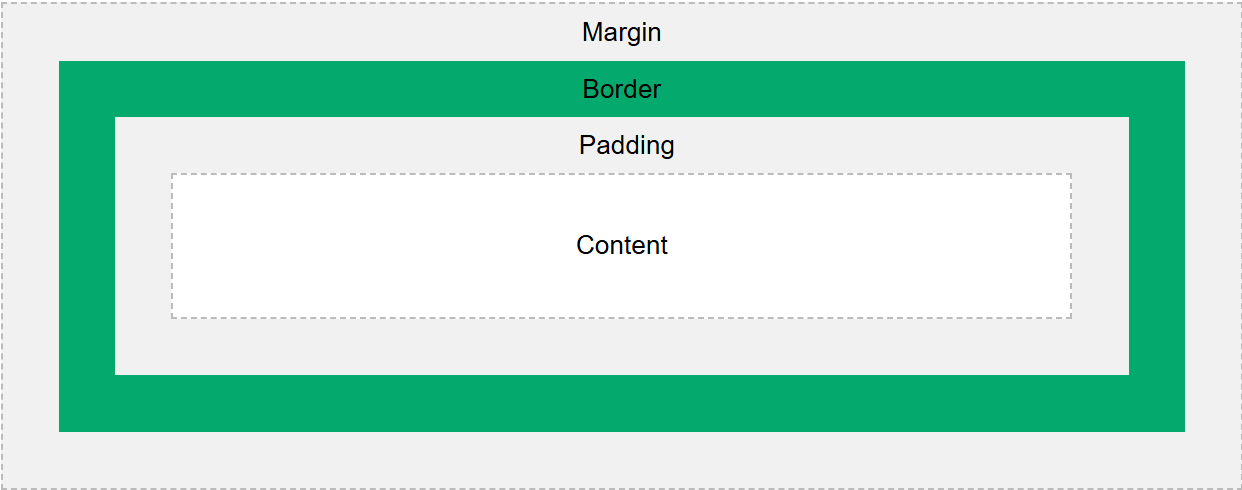
| **Selector Type** | **Syntax** | **Description** |
| --- | --- | --- |
| **Universal Selector** | \* | Selects **all** elements. |
| **Type Selector** | p, h1, div | Selects elements by **tag name**. |
| **Class Selector** | .classname | Selects elements with a **specific class** attribute. |
| **ID Selector** | #idname | Selects an element with a specific **ID** attribute. |
| **Group Selector** | h1, p, div | Applies the same styles to **multiple selectors**. |
| **Descendant Selector** | div p | Selects <p> elements **inside** <div>. |
| **Child Selector** | ul > li | Selects <li> that are **direct children** of <ul>. |
| **Adjacent Sibling** | h1 + p | Selects the **first <p>** immediately **after <h1>**. |
| **General Sibling** | h1 ~ p | Selects **all <p> siblings** that follow an <h1>. |
| **Attribute Selector** | input[type="text"] | Selects elements with a **specific attribute and value**. |
| **Pseudo-class Selector** | a:hover, li:first-child | Targets elements in a specific **state** or position. |
| **Pseudo-element Selector** | p::first-line, div::before | Selects **part of an element**, like first line or content before it. |

* 1. **What is the box model in CSS? Explain its components.**
* The **CSS Box Model** is a fundamental concept that defines how the size of elements is calculated and how they are rendered on the page.

Each element is a rectangular box that consists of **four parts**:

* **Components of the Box Model**

1. **Content**
   * The actual content of the box (text, image, etc.).
   * You can set its dimensions using width and height.
2. **Padding**
   * Space **inside** the element, between the content and the border.
   * You can control with padding, padding-top, padding-right, etc.
3. **Border**
   * A **line around** the padding (and content).
   * Controlled using border, border-width, border-style, and border-color.
4. **Margin**
   * Space **outside** the border.
   * Creates distance between the element and others.
   * Controlled using margin, margin-top, etc.

****

### **Responsive Web Design**

* 1. **What is responsive web design? Why is it important?**
* Responsive web design ensures websites adapt seamlessly to various devices and screen sizes, providing an optimal user experience. This is crucial because it makes websites accessible and user-friendly on desktops, laptops, tablets, and mobile phones.

* **Why Responsive Design is Important:**
* Improved User Experience:
* **SEO Benefits:**
* **Cost-Effective:**
* **Increased Accessibility:**
  1. **Explain the use of media queries in CSS. Provide an example.**
* **Media Queries** are a key tool in responsive design. They let you apply CSS styles **based on conditions like screen width, height, resolution, or device type**.
* **Syntax of a Media Query:**

@media (condition) {

/\* CSS rules \*/

}

* **Example:**

**css**

**/\* Default styles (Mobile First) \*/**

**body {**

**font-size: 16px;**

**background-color: white;**

**}**

**/\* Styles for tablets and larger \*/**

**@media (min-width: 768px) {**

**body {**

**font-size: 18px;**

**background-color: lightgray; }**

**}**

**/\* Styles for desktops and larger \*/**

**@media (min-width: 1024px) {**

**body {**

**font-size: 20px;**

**background-color: lightblue;**

**}**

**}**

* In this example:

Mobile devices use the default styles.

Tablets (768px and up) get slightly larger text and a different background.

Desktops (1024px and up) get even larger text and another background color.

* 1. **What are the benefits of using a mobile-first approach in web design?**
* **Mobile-First Design** means you start designing and coding for **small screens first**, then scale up for larger screens using media queries.
* **Why Mobile-First is Beneficial:**

1. **Better Performance**  
   Loads only essential content and assets, reducing load time on slower mobile networks.
2. **Improved Usability**  
   Focuses on **core content and functionality** before adding enhancements for larger screens.
3. **Wider Accessibility**  
   Ensures your site works for the majority of users who access it on mobile.
4. **Progressive Enhancement**  
   Starts simple and adds complexity only when necessary—makes the site more robust.

4. PHP Integration

1. **How Can PHP Be Used to Dynamically Generate HTML Content?** **Provide examples.**

* PHP can **embed HTML** within its code to generate web pages that respond dynamically—such as displaying user data, reading from databases, or customizing content based on conditions.

**Example : Display a personalized greeting**

php

<?php

$user = "Alex";

echo "<h1>Welcome, $user!</h1>";

?>

**Output:**

html

<h1>Welcome, Alex!</h1>

1. **Explain How to Include CSS Files in a PHP-Generated HTML Page**

* You include CSS in PHP exactly as you would in HTML—**inside the <head> tag**, as long as you output proper HTML.

**Example: Including a CSS File**

php

Copy code

<!DOCTYPE html>

<html>

<head>

<title>My Page</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<h1>Hello from PHP!</h1>

</body>

</html>

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

* **Advantages of Using PHP to Manage HTML Forms**
  1. 🛠 **Form Processing**  
     PHP can easily collect and process form data using $\_POST or $\_GET.
  2. 🔐 **Validation and Security**  
     PHP allows server-side validation, sanitization, and protection against threats like SQL injection and XSS.
  3. 💾 **Data Handling**  
     Easily store submitted data into a database (e.g., MySQL) or file.
  4. 🔁 **Dynamic Feedback**  
     Show error/success messages or re-populate forms on submission failure.
  5. 📋 **Flexible Control Flow**  
     Use logic to show or hide parts of the form, change inputs dynamically, or redirect users.

# **5. Practical Example: Multiple Tables and SQL Queries**

LAB EXERCISE:

• Create multiple tables and perform queries using:

o SELECT, UPDATE, DELETE, INSERT

o WHERE, LIKE, GROUP BY, HAVING

o LIMIT, OFFSET, Subqueries, AND, OR, NOT, IN