**HTML, CSS and JS in PHP**

**HTML**

1. What is HTML? Explain its structure.

HTML stands for Hyper Text Markup Language.

HTML is the standard markup language for creating Web pages.

HTML describes the structure of a Web page

HTML consists of a series of elements

HTML elements tell the browser how to display the content

HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

**Structure of html**

<!DOCTYPE html>  
<html>  
<head>  
 <title>Page Title</title>  
</head>  
<body>  
 <h1>My First Heading</h1>  
 <p>My first paragraph.</p>  
</body>  
</html>

* The <!DOCTYPE html> declaration defines that this document is an HTML5 document
* The <html> element is the root element of an HTML page
* The <head> element contains meta information about the HTML page
* The <title> element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab)
* The <body> element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.
* The <h1> element defines a large heading
* The <p> element defines a paragraph

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

HTML (Hyper Text Markup Language) **tags** are the building blocks of web pages. They are used to:

* **Structure** the content (headings, paragraphs, lists, etc.)
* **Embed** media (images, videos, links)
* **Define** document sections (header, footer, navigation)
* **Enhance** meaning and formatting (bold, italic, emphasized text)

Html tags come in pair like

<tagname>Content of the page</tagname>

Some tags are self closing:

<img src=”image1.jpg” alt=”sample image” />

Example of commonly used tags

| **Tag** | **Purpose** | **Example** |
| --- | --- | --- |
| <html> | Root element of an HTML page | <html> ... </html> |
| <head> | Contains metadata | <head> ... </head> |
| <title> | Sets the page title (shown in browser tab) | <title>My Page</title> |
| <body> | Contains visible content | <body> ... </body> |
| <h1> to <h6> | Headings (h1 is highest, h6 is lowest) | <h1>Welcome</h1> |
| <p> | Paragraph | <p>This is a paragraph.</p> |
| <a> | Hyperlink | <a href="https://example.com">Visit</a> |
| <img> | Image | <img src="photo.jpg" alt="Photo" /> |
| <ul> / <ol> | Unordered / Ordered list | <ul><li>Item</li></ul> |
| <li> | List item | <li>First item</li> |
| <div> | Generic block-level container | <div>Content</div> |
| <span> | Generic inline container | <span>Text</span> |
| <br> | Line break | Line 1<br>Line 2 |
| <form> | Creates a form | <form>...</form> |
| <input> | Input field | <input type="text"> |
| <button> | Button element | <button>Click</button> |

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

Block-level elements take up the entire available width, start on a new line, and are often used for larger content blocks like paragraphs or sections. Inline elements, on the other hand, only take up the space necessary for their content, do not start on a new line, and can be placed within other inline elements.

Block-level elements:

* **Example:** <div>, <p>, <h1>, <ul>, <ol>, <hr>, <header>, <footer>.
* **Characteristics:**
  + Always start a new line.
  + Take up the full width of their parent container.
  + Can have margins applied to them.
  + Can contain other block-level or inline elements.

Inline elements:

* **Example:** <span>, <a>, <em>, <strong>, <b>, <i>, <img>.
* **Characteristics:**
  + Do not start a new line.
  + Take up only as much width as their content requires.
  + Cannot have top and bottom margins applied to them.
  + Can only contain other inline elements.

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

**Semantic HTML** refers to the use of HTML tags that **clearly describe the meaning or purpose** of the content they contain. Instead of using generic tags like <div> or <span> everywhere, semantic HTML encourages using tags such as:

* <header> – Represents the header of a page or section
* <nav> – Represents a navigation menu
* <main> – Denotes the main content of a document
* <article> – Represents a self-contained piece of content
* <section> – Groups related content together
* <footer> – Represents the footer of a page or section
* <aside> – Contains content that is tangentially related to the main content

It is important because

**Improves Accessibility**

* Semantic elements help **screen readers** and other assistive technologies better understand the structure and purpose of a webpage.

**Enhances SEO (Search Engine Optimization)**

* Search engines like Google use semantic tags to better interpret the content, improving **search rankings**.

**Better Code Readability**

* Makes HTML more **organized and meaningful** for developers, improving collaboration and maintenance.

**Standardization and Best Practices**

* Follows **W3C standards**, promoting consistent web development practices.

**Improved Browser Support**

* Modern browsers can **render semantic elements more appropriately**, which can enhance layout and performance.

**CSS**

1. **What is CSS? How does it differ from HTML?**

**CSS (Cascading Style Sheets)** is a **stylesheet language** used to describe the **presentation and layout** of HTML elements on a webpage. It controls the **look and feel** of a site — including colors, fonts, spacing, alignment, and positioning.

| **Feature** | **HTML (HyperText Markup Language)** | **CSS (Cascading Style Sheets)** |
| --- | --- | --- |
| **Purpose** | Structures content | Styles content (appearance) |
| **Function** | Defines elements like headings, paragraphs, forms, images, etc. | Defines how those elements look (colors, layout, font, etc.) |
| **Type** | Markup language | Style sheet language |
| **Example** | <p>This is a paragraph</p> | p { color: blue; font-size: 16px; } |
| **Placement** | Written in .html files | Written in .css files or inside <style> tags in HTML |
| **Effect on Display** | Provides the structure, but not design | Controls visual presentation only |

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

CSS can be applied to an HTML page in **three main ways**:

* 1. Inline
  2. Internal
  3. External

**1. Inline CSS**

CSS is written **directly within an HTML tag** using the style attribute.

**✅ Example:**

<p style="color: blue; font-size: 18px;">This is a blue paragraph.</p>

**Used for:**

* Quick styling of a **single element**
* **Testing or temporary changes**

**Not ideal for:**

* Large projects
* Reusability or consistency

**2. Internal CSS**

CSS is written **inside a <style> tag** within the <head> section of the HTML document.

**✅ Example:**

<!DOCTYPE html>

<html>

<head>

<style>

h1 {

color: green;

text-align: center;

}

</style>

</head>

<body>

<h1>Welcome!</h1>

</body>

</html>

**Used for:**

* Styling a **single webpage**
* When external CSS is not needed

**Not ideal for:**

* Projects with **multiple pages**, as styles aren’t reusable across files

**3. External CSS**

CSS is written in a **separate .css file** and linked to the HTML document using the <link> tag.

**Example:**

**HTML (index.html):**

<head>

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

</head>

**CSS (styles.css):**

body {

background-color: #f0f0f0;

font-family: Arial, sans-serif;

}

**Used for:**

* Large websites with **multiple pages**
* Clean code and **better maintainability**

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

CSS selectors are used to "find" (or select) the HTML elements you want to style.

1.Universal selector : Selects all element of the page

\* {

margin: 0;

padding: 0;

}

2.Element selector : Targets element by name

p {

color: blue;

}

3.Class selector : Targets elements with a specific class attribute.

.box {

border: 1px solid black;

}

Applies to elements like <div class="box">.

4.ID selector(#) : Targets an element with a specific id attribute.

#header {

background-color: lightgray;

}

Applies only to an element like <div id="header">.

5.Grouping selector: Applies the **same style** to multiple elements.

h1, h2, p {

font-family: Arial;

}

6.Descendant Selector : Targets elements **inside** other elements.

div p {

color: green;

}

Styles <p> tags inside <div> tags.

7.Child Selector (>) : Selects elements that are a **direct child** of another.

ul > li {

list-style-type: square;

}

8. Pseudo-class Selector : Selects elements in a **specific state**.

a:hover {

color: orange;

}

Common pseudo-classes: :hover, :first-child, :nth-child(n)

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

The CSS box model is essentially a box that wraps around every HTML element. It consists of: content, padding, borders and margins. The image below illustrates the box model:

+-----------------------------+

| Margin |

| +-----------------------+ |

| | Border | |

| | +------------------+ | |

| | | Padding | | |

| | | +------------+ | | |

| | | | Content | | | |

| | | +------------+ | | |

| | +------------------+ | |

| +-----------------------+ |

+-----------------------------+

1. **Content**

This is the actual content of the element, such as text or an image.

You can control its size using properties like width and height.

**2. Padding**

* Padding is the space between the content and the border.
* It adds space **inside** the element.
* Controlled using padding, padding-top, padding-right, etc.

**3. Border**

* The border wraps around the padding (if any) and content.
* You can set its width, style, and color using border, border-width, border-style, and border-color.

**4. Margin**

* Margin is the space **outside** the border.
* It creates distance between this element and surrounding elements.
* Controlled using margin, margin-top, margin-right, etc.

**Box Sizing Note:**

* By default, the total width/height of an element is:  
  width + padding + border
* Use box-sizing: border-box; to include padding and border **inside** the width/height:

css

\* {

box-sizing: border-box;

}

Responsive Web Design

1. What is responsive web design? Why is it important?

Responsive Web Design is **designing websites that contain flexible layouts** that can scale itself according to the screen size of the device it is being viewed on.

**How to Create a Responsive Web Design?**

Creating a responsive web design means making sure your website looks good and works well on all devices, from phones to tablets to computers. Here are the steps to follow:

**Step 1: Use Fluid Grids**

* Instead of using fixed-width layouts, use fluid grid systems. This means the layout of your website will adjust smoothly based on the screen size. Fluid grids use percentages for widths, so elements resize proportionally.

.container {

width: 100%;

}

.column {

width: 33.33%; /\* 1/3 of the container \*/

}

**Step 2: Use Flexible Images**

* Make sure images and other media content can scale within their containers. Use CSS properties like max-width: 100% to ensure images adjust to the size of their containing elements, preventing them from overflowing the layout.

img {

max-width: 100%;

height: auto;

}

**Step 3: Apply Media Queries**

* Use media queries in your CSS to apply different styles for different screen sizes and orientations. Media queries allow you to change the layout, font sizes, colors, and other styles based on the width, height, and resolution of the device. For example, you can write a media query to make text larger on small screens.

@media (max-width: 768px) {

body { font-size: 14px;}

}

**Step 4: Prioritize Touchscreens**

* Ensure your website is touch-friendly. Make buttons and other interactive elements large enough for finger tapping. Place navigational elements in easily accessible areas. This makes it easier for users on mobile devices to navigate your site without frustration.

**Step 5: Test Responsiveness**

* Continuously test your website on different devices and browsers to make sure it looks and functions well everywhere. Use tools and emulators to see how your site performs on various screen sizes and make adjustments as needed.

| **Reason** | **Explanation** |
| --- | --- |
| ✅ **Improves User Experience** | Visitors can easily read and navigate on any device. |
| ✅ **Mobile-Friendly** | With increasing mobile users, it's essential for accessibility. |
| ✅ **Better SEO** | Google prioritizes mobile-responsive sites in search rankings. |
| ✅ **Cost-Effective** | One responsive site eliminates the need for separate mobile/desktop versions. |
| ✅ **Future-Proof** | Works well on all screen sizes, including new devices. |

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

**Media queries** are a feature of CSS used to apply styles based on the **device’s characteristics**, like:

* Screen **width** and **height**
* **Device type** (screen, print, etc.)
* **Orientation** (portrait or landscape)
* **Resolution** (e.g., retina displays)

They are essential for **responsive web design**, which ensures your site looks good on all devices — from phones to desktops.

**@media media-type and (condition) {**

**/\* CSS rules \*/**

**}**

media-type: Type of device (e.g., screen, print, all)

condition: The condition under which styles will apply (e.g., max-width: 768px)

Common media types

| **Media Type** | **Description** |
| --- | --- |
| screen | For computer screens, tablets, phones |
| print | For printed material or print preview |
| all | Applies to all devices |

Common media features

| **Feature** | **Description** | **Example** |
| --- | --- | --- |
| width / height | Viewport width/height | min-width: 600px |
| orientation | Portrait or landscape | orientation: portrait |
| resolution | Screen DPI | min-resolution: 300dpi |
| aspect-ratio | Width/height ratio | aspect-ratio: 16/9 |

Examples

/\* Default styles (desktop-first approach) \*/

body {

background-color: white;

font-size: 18px;

}

/\* Tablet (≤768px) \*/

@media screen and (max-width: 768px) {

body {

background-color: lightblue;

font-size: 16px;

}

}

/\* Mobile (≤480px) \*/

@media screen and (max-width: 480px) {

body {

background-color: lightgray;

font-size: 14px;

}

}

Combining Multiple Conditions

@media screen and (min-width: 600px) and (max-width: 900px) {

.container {

width: 80%;

}

}

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

Mobile-first design is an approach where you design your website for **mobile devices** first and then for desktops. This method is crucial because most people today use their **smartphones** and **tablets** to go online rather than laptops or computers. **Mobile-first designs** ensure that your website not only looks good but also works seamlessly across different devices, providing the best [**user experience**](https://www.geeksforgeeks.org/user-experience-or-ux-design/). This approach helps increase **website traffic** and ensures that your site adjusts to different screen sizes effectively.

To create a successful **mobile-first website**, consider the following tips:

* Use **large fonts** and **buttons** to make [navigation](https://www.geeksforgeeks.org/navigation-element-in-web-design/) easier on small screens.
* Keep the text **short** and to the point to improve readability.
* Use **high-quality images** to make the content visually appealing.

Benefits

**1.Improved User Experience on Mobile**

**2. Performance Optimization**

**3.** **Progressive Enhancement**