## HTML Basics

## Question 1: Define HTML. What is the purpose of HTML in web development?

**ANS:** HTML, which stands for Hypertext Markup Language, is a fundamental component of web development. It serves as the backbone of web pages, allowing developers to structure and present content on the internet. HTML is a markup language that uses a series of tags to define the structure and formatting of a web page.

**Purpose of HTML in web development:**

* It allows Developers to create Structured Document by defining Headings, Paragraphs, lists, Tables, images and other elements. By using these tags, we convey the meaning of content, making it more Accessible and understable for both humans and machine.
* HTML Elements such as divs, spans and sections, to help organize and group content, making it easier to style and layout.
* HTML enables the creation of hyperlinks, connecting web pages and allowing users to navigate between them.
* HTML provides a way to create forms, input fields, and buttons, enabling user interaction and data collection.
* HTML elements, such as headings and meta tags, help search engines understand the content and relevance of web pages.
* HTML ensures that web pages can be displayed consistently across different browsers and devices.
* HTML allows web pages to be viewed on various operating systems, devices, and screen sizes.

**Question 2**: **Explain the basic structure of an HTML document. Identify the mandatory tags and their purposes.**

**ANS:**  The basic structure of an HTML document consists of several key elements, which are represented by tags. Here's a breakdown of the mandatory tags and their purposes:

**Basic Structure:**

<! DOCTYPE html>

<html>

<head>

<! -- metadata and links to external resources -->

</head>

<body>

<! -- content of the HTML document -->

</body>

</html>

**Mandatory Tags:**

**1. <! DOCTYPE html>**

- Declares the document type and version of HTML.

- Must be the first line of the HTML document.

**2. <html>**

- Represents the root element of the HTML document.

- Contains all the other elements.

**3. <head>**

- Provides metadata about the document, such as character encoding, title, and links to external stylesheets or scripts.

- Must be the first child element of the <html> element.

**4. <title>**

- Specifies the title of the document, which appears in the browser's title bar and in search engine results.

- Must be a child element of the <head> element.

**5. <body>**

- Contains the content of the HTML document.

- Must be the second child element of the <html> element, after the <head> element.

By including these mandatory tags, you'll create a well-structured HTML document that provides a solid foundation for building web pages.

**Question 3**: What is the difference between block-level elements and inline elements in HTML? Provide examples of each.

**ANS:** In HTML, elements can be categorized into two main types: block-level elements and inline elements. The main difference between these two types of elements lies in their display behaviour and how they interact with surrounding elements.

**Block-Level Elements**

Block-level elements occupy the full width of their parent container, creating a new block of content. They typically start on a new line and can contain other block-level elements or inline elements. Examples of block-level elements include:

1. Structural Elements

- <div>: A generic container element.

- <p>: A paragraph element.

- <h1>-<h6>: Heading elements.

- <ul>, <ol>, <dl>: List elements.

- <table>: A table element.

2. Semantic Elements

- <header>: A header element.

- <footer>: A footer element.

- <nav>: A navigation element.

- <main>: A main element.

- <section>: A section element.

**Inline Elements**

Inline elements, on the other hand, occupy only the space needed for their content and do not create a new block of content. They typically do not start on a new line and can only contain other inline elements or text. Examples of inline elements include:

**1. Text-Related Elements**

- <span>: A generic inline container element.

- <a>: An anchor element (hyperlink).

- <strong>, <b>: Strong or bold text elements.

- <em>, <i>: Emphasized or italic text elements.

**2. Media-Related Elements**

- <img>: An image element.

In summary, block-level elements create a new block of content and can contain other block-level or inline elements, while inline elements occupy only the space needed for their content and can only contain other inline elements or text.

**Question 4: Discuss the role of semantic HTML. Why is it important for accessibility and SEO? Provide examples of semantic elements.**

**ANS:** Semantic HTML plays a crucial role in modern web development by providing meaning to the structure of web pages. It involves using HTML elements that describe the content they contain, making it easier for browsers, search engines, and assistive technologies to understand the context and significance of the content.

**Importance for Accessibility:**

Semantic HTML is essential for accessibility because it:

1. Improves screen reader experience: Screen readers can interpret semantic elements, providing a more accurate and meaningful experience for users with visual impairments.
2. Enhances keyboard navigation: Semantic elements help users navigate web pages using keyboards, improving accessibility for individuals with mobility or dexterity impairments.
3. Supports assistive technologies: Semantic HTML enables assistive technologies, such as voice control software, to better understand web page content and provide more effective support.

**Importance for SEO**

Semantic HTML is also important for Search Engine Optimization (SEO) because:

1. Improves search engine understanding: Search engines can better comprehend the structure and content of web pages, leading to more accurate indexing and ranking.
2. Enhances keyword relevance: Semantic elements help search engines understand the context and relevance of keywords, improving the page's visibility for related searches.
3. Supports rich snippets: Semantic HTML enables the creation of rich snippets, which provide users with more detailed information about web pages in search engine results pages (SERPs).

**Examples of Semantic Elements**

Here are some examples of semantic elements:

1. Header elements: <header>, <nav>, <h1>-<h6>
2. Footer element: <footer>
3. Sectioning elements: <section>
4. Table elements: <table>
5. Form elements: <form>, <label>, <input>, <textarea>, <select>
6. Interactive elements: <button>

By using semantic HTML elements, developers can create web pages that are more accessible, easier to maintain, and better optimized for search engines.

**HTML Forms**

**Question 1: What are HTML forms used for? Describe the purpose of the input, textarea, select, and button elements.**

**ANS:** HTML forms collect user input and send it to a server for processing, used for tasks like registration, login, and feedback.

**Common Uses:**

* **Registration:** Collecting user details for creating accounts.
* **Contact Forms:** Allowing users to send messages or inquiries.
* **Surveys:** Gathering feedback or data from users.
* **Shopping:** Entering shipping and payment information.
* **Search:** Allowing users to enter search terms.

**Purpose of the input, textarea, select, and button elements:**

* The **<input>** tag specifies an input field where the user can enter data.
* The purpose of the **<textarea>** element in an HTML form is to provide a multi-line text input field, allowing users to enter a larger amount of text, such as comments, reviews, or messages, compared to a single-line input field.
* The **<select>** element is used to create a drop-down list. The <select> element is most often used in a form, to collect user input.
* In an HTML form, a button, typically implemented using the **<button>**or <input type="button"> elements, serves as an interactive element that, when clicked, triggers an action, such as submitting the form, login etc.

**Question 2:** Explain the difference between the GET and POST methods in form submission. When should each be used?

**ANS: Difference between the GET and POST methods in form submission:**

|  |  |  |
| --- | --- | --- |
|  | GET | POST |
| Back button/Reload | Harmless | Data will be Re-submitted |
| Bookmarked | Can be bookmarked | Cannot be bookmarked |
| Cached | Can be cached | Cannot be cached |
| Encoding type | url encoded | url encoded |
| History | Parameters Remain in Browser history | Parameters are not saved in Browser history |
| Restrictions on data length | Yes, when sending data, the GET method adds the data to URL and the length of URL is limited(2048 characters) | No Restrictions |
| Restriction on Data Type | Only ASCII characters allowed | No Restrictions. Binary data is also allowed |
| Security | Less secure when sending password or other sensitive information | Safer than GET Because the parameters are not stored in browser history |
| Visibility | Data is visible to everyone in URL | Data is not displayed in the URL |

Use GET when you want the request to be bookmarkable, shareable, and cached. Use POST when dealing with sensitive data, file uploads, or creating/updating resources.

**Question 3: What is the purpose of the label element in a form, and how does it improve accessibility?**

**ANS:** The primary purpose of the <label> element is to associate a descriptive text label with a form control, making it clear what information is expected in that field.

**Accessibility Benefits:**

* **Screen Reader Users:** Screen readers announce the label text when a user focuses on the associated form control, helping users understand what to enter.
* **Improved Form Navigation:** Labels help users understand the purpose of each form field, making it easier to navigate and complete the form.

**HTML Tables**

**Question 1**: Explain the structure of an HTML table and the purpose of each of the following elements: <table>, <tr>, <th>, <td>, and <thead>.

**ANS:** The structure of an **HTML table** is organized using specific tags that define rows, columns, and headers. Here's a breakdown of the main elements:

**Basic HTML Table Structure:**

<html>

<table>

<thead>

<tr>

<th>Header 1</th>

<th>Header 2</th>

</tr>

</thead>

<tbody>

<tr>

<td>Row 1, Cell 1</td>

<td>Row 1, Cell 2</td>

</tr>

<tr>

<td>Row 2, Cell 1</td>

<td>Row 2, Cell 2</td>

</tr>

</tbody>

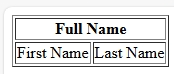
</table>

* **Table:** It Acts as the container for all the table content. It wraps the entire table structure including rows, headers, and data.
* **<tr>(Table Row):** Defines a row in the table. Inside <table>, it groups cells (<th> or <td>) horizontally.
* **<th> (Table Header):** Represents a header cell — usually used for column or row titles. Text is bold and centred by default. Often used in the <thead> section.
* **<td> (Table Data):** Represents a **standard data cell** in a table. Used inside <tr> to display actual content.
* **<thead> (Table Head):** Groups the **header row(s)** of the table. Helps structure tables and improve accessibility and styling.

**Question 2: What is the difference between colspan and rowspan in tables? Provide examples**.

ANS: **colspan:** It Merges **multiple columns** into a single cell. Used when you want a cell to span across **two or more columns** horizontally.

**Example of Colspan:**



<table border="1">

<tr>

<th colspan="2">Full Name</th>

</tr>

<tr>

<td>First Name</td>

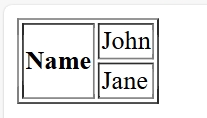
<td>Last Name</td>

</tr>

</table>

**Rowspan:** Merges Multiple Rows into single cell. It Used when you want a cell to span across two or more rows vertically.

**Example of Rowspan:**



<table border="1">

<tr>

<th rowspan="2">Name</th>

<td>John</td>

</tr>

<tr>

<td>Jane</td>

</tr>

</table>

**Question 3: Why should tables be used sparingly for layout purposes? What is a better alternative?**

**ANS:** While HTML tables **can** be used to create page layouts, it’s **not recommended** anymore. Here's why:

***Problems with Using Tables for Layout:***

1. **Poor Accessibility**: Screen readers and assistive technologies may misinterpret layout tables as data tables.
2. **Hard to Maintain:** Changing layout requires modifying lots of nested table code. very messy and time-consuming.
3. **Slow Loading**: Tables load all at once, which can slow down the rendering of your webpage.
4. **Not Responsive**: Tables don’t adapt well to small screens like phones or tablets.
5. **Bad for SEO**: Search engines prefer semantic, clean HTML. Tables used for layout confuse structure.

### Better Alternative: **CSS with Semantic HTML**

### Instead of tables, use:

### <div>, <section>, <header>, etc. for structure with CSS Layout.

### 