HTML in Full Stack

❖ HTML Basics

1) Define HTML. What is the purpose of HTML in web development?

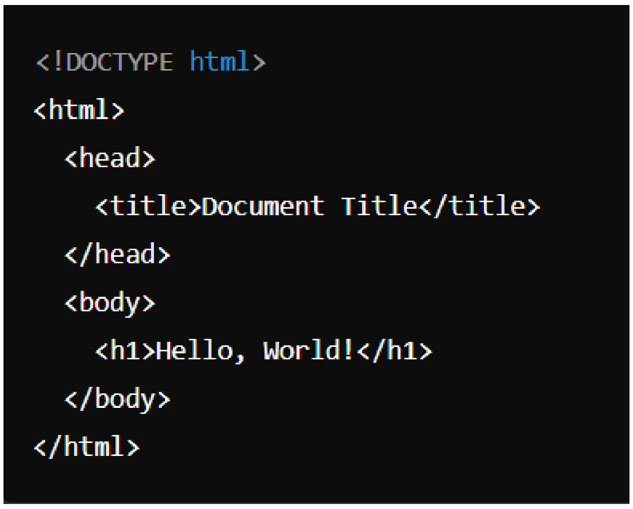
Ans. HTML (Hypertext Markup Language) is the standard language used to create and structure content on the web. It defines the structure of web pages using elements like headings, paragraphs, links, images, and more.

Purpose in Web Development:

HTML provides the foundation for web pages by organizing content into a readable and navigable structure. It works alongside CSS (for styling) and JavaScript (for interactivity) to build complete, functional websites.

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

Ans. The basic structure of an HTML document includes the following mandatory tags:



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

Mandatory Tags and Their Purposes:

1. **<!DOCTYPE html>**: Declares the document type and version of HTML.
2. **<html>**: The root element that contains all other HTML elements.
3. **<head>**: Contains metadata like the title, character encoding, and links to stylesheets or scripts.
4. **<title>**: Specifies the title of the document (shown in the browser tab).
5. **<body>**: Contains the main content of the webpage visible to use

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

Ans. Block-level elements and inline elements define how content is displayed on a webpage:

Block-Level Elements

- **Definition**: Occupy the entire width of their container, starting on a new line.
- **Purpose**: Create structure and separate sections of content.
- **Examples**:
  - <div>: General container for content.
  - <p>: Represents a paragraph.
  - <h1> to <h6>: Headings.
  - <ul> and <ol>: Lists.
  - <section> and <article>: Define sections.

Inline Elements

- **Definition**: Occupy only as much width as necessary, remaining in the same line as surrounding content.
- **Purpose**: Style or interact with specific parts of text or inline content.
- **Examples**:

    - <span>: General inline container.
    - <a>: Hyperlink.
    - <strong>: Bold text.
    - <em>: Italicized text.
    - <img>: Inline image.

Block-level elements are for structure, while inline elements are for styling or emphasizing content within a structure.

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

Ans. Role of Semantic HTML:

Semantic HTML uses meaningful tags that clearly describe the content and its purpose. Examples include <header>, <footer>, <article>, and <nav>.

Importance

1. **Accessibility**:
    - Improves navigation for screen readers and assistive technologies, making content more accessible to users with disabilities.
    - Example: <nav> helps users locate navigation menus.
2. **SEO (Search Engine Optimization)**:
    - Search engines understand the structure and relevance of content better, improving rankings.
    - Example: <article> signals a distinct piece of content.

Examples of Semantic Elements

- <header>: Defines the header section of a page.
- <footer>: Represents the footer section.

- <main>: Highlights the primary content.
- <section>: Groups related content.
- <aside>: Contains side content, like ads or related links.

❖ <u>HTML Forms</u>

1) **What are HTML forms used for? Describe the purpose of the input, text-area, select, and button elements.**

Ans. HTML Forms:

HTML forms are used to collect user input and send it to a server for processing, such as login details, survey responses, or search queries.

Key Elements and Their Purposes

1. **<input>**:
   - Collects user data like text, passwords, emails, etc.
   - Types include text, password, email, checkbox, radio, etc.
2. **<text-area>**:
   - Allows users to input multi-line text, such as comments or messages.
3. **<select>**:
   - Creates a dropdown menu for users to select an option.
   - Used with <option> to define choices.
4. **<button>**:
   - Triggers actions like submitting the form or running scripts.
   - Can display text, images, or icons.

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

Ans. Difference Between GET and POST Methods:

1. **GET Method**:
   - Sends data as a query string appended to the URL (e.g., example.com?name=Karan).
   - **Characteristics**:
     - Data is visible in the URL.
     - Limited data size (depends on URL length).
     - Cacheable and bookmarkable.
   - **Use Case**:
     - When retrieving non-sensitive data, like search queries.
2. **POST Method**:
   - Sends data in the request body, not visible in the URL.
   - **Characteristics**:
     - Secure for sensitive data (e.g., passwords).
     - No size limitations.
     - Not cacheable or bookmarkable.
   - **Use Case**:
     - When submitting sensitive or large amounts of data, like login forms or file uploads.

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

Ans. Purpose of the <label> Element:

The <label> element is used to define a text label for form controls like <input> or <textarea>. It links descriptive text to form elements, improving usability.

How It Improves Accessibility

1. **Screen Readers**:
   - Allows assistive technologies to read the label, helping visually impaired users understand the purpose of form fields.
2. **Click Association**:
   - Clicking the label focuses the associated form control, making it easier to interact with.

Example

```
<label for="username">Username:</label>
<input type="text" id="username" name="username">
```

❖ <u>HTML Tables</u>

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

Ans. Structure of an HTML Table:

An HTML table organizes data into rows and columns.

Elements and Their Purposes:

1. **<table>**:
   - The container element for the entire table.
   - Defines the structure and layout of tabular data.

2. **<tr>** (Table Row):

   - Represents a single row within the table.
   - Groups cells (<th> and <td>) horizontally.

3. **<th>** (Table Header):

   - Represents a header cell in a table.
   - Typically, bold and centered by default.
   - Used for column or row labels.

4. **<td>** (Table Data):

   - Represents a standard data cell.
   - Contains the actual content of the table.

5. **<thead>** (Table Head):

   - Groups header rows (<tr> with <th> cells).
   - Makes the header semantically distinct and reusable.

Example

```
<table>
  <thead>
    <tr>
      <th>Name</th>
      <th>Age</th>
    </tr>
  </thead>
  <tr>
    <td>Karan</td>
    <td>20</td>
  </tr>
</table>
```

2) What is the difference between colspan and rowspan in tables? Provide examples.

Ans. D**ifference Between colspan and rowspan:**

1. **colspan**:

   - Merges multiple columns into a single cell.
   - Used when a cell needs to span across multiple columns.

Example:

```
<table>
  <tr>
    <th colspan="2">Header</th>
  </tr>
  <tr>
    <td>Cell 1</td>
    <td>Cell 2</td>
  </tr>
</table>
```

2. **rowspan**:

- Merges multiple rows into a single cell.
- Used when a cell needs to span across multiple rows.

Example:

```
<table>
  <tr>
    <td rowspan="2">Row 1 & 2</td>
    <td>Row 1, Cell 2</td>
  </tr>
  <tr>
    <td>Row 2, Cell 2</td>
  </tr>
</table>
```

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

Ans. Why Tables Should Be Used Sparingly for Layout Purposes:

1. **Semantic Issues**:
    - Tables are meant for tabular data, not layout, leading to confusing and non-meaningful HTML.
2. **Accessibility**:
    - Screen readers struggle to interpret layout tables, making navigation difficult for visually impaired users.
3. **Maintenance**:
    - Tables make layout harder to manage and update compared to modern techniques.
4. **Performance**:
    - Tables load slower as browsers must process the entire table before rendering.

Better Alternative: -

- **CSS (Cascading Style Sheets)**:
    - Use **CSS Grid** and **Flexbox** for layout design.
    - They offer more flexibility, responsiveness, and cleaner, semantic HTML.

Example of CSS Layout:

```
<div style="display: flex;">
  <div>Box 1</div>
  <div>Box 2</div>
</div>
```