Module 2 – Frontend – HTML

HTML Forms:

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

➢ HTML forms are used to collect user input data and send it to a web server for processing. They are essential for creating interactive websites that require user interaction such as login, registration, search, feedback, and more.

➢ **Key Elements of HTML Forms:**

1. <input> Element

Used to create various types of user input fields such as:

- type="text" → Single-line text input
- type="password" → Password field
- type="checkbox" / type="radio" → Multiple/single choice
- type="email", type="file", etc. → Specialized input formats

Example:

html

CopyEdit

```
<input type="text" name="username" placeholder="Enter username">
```

2. <textarea> Element

Used for multi-line text input (like comments or descriptions).

Example:

html

CopyEdit

```
<textarea name="message" rows="4" cols="50"></textarea>
```

---

3. <select> Element

Creates a drop-down list of options for users to choose from. Options are defined using <option> tags.

Example:

html

CopyEdit

```
<select name="country">
  <option value="india">India</option>
  <option value="usa">USA</option>
</select>
```

4. <button> Element

Used to submit the form or trigger JavaScript actions. It can be of type submit, reset, or button.

Example:

html

CopyEdit

```
<button type="submit">Submit</button>
```

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

- ➢ In HTML forms, the GET and POST methods define how data is sent to the server when the form is submitted.

1. GET Method:

- Data is sent via the URL (as query parameters).

- **Example:**
  https://example.com/search?query=html
- Visible to users (in browser address bar).
- Limited data size (URL length limit).
- Not secure for sensitive data (passwords, personal info).
- Can be bookmarked and cached.
- Used for retrieving data, not modifying it.

  ➢ **Use GET when:**
- The action is safe and idempotent (doesn't change server state).
- You want the URL to be shareable/bookmarkable (e.g., search forms).

🔐 2. POST Method:

➢ Data is sent in the request body, not visible in URL.

➢ More secure than GET for sending sensitive data.

➢ No size limitation (within server/browser limits).

➢ Cannot be bookmarked or cached.

➢ Used to submit or update data on the server**.**

❖ **Use POST when:**

- The action modifies server data (e.g., registration, login, order).
- You are sending large amounts of data.
- You are submitting private or sensitive information.

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

➢ **Purpose of the <label> Element:**

The <label> element is used to define a caption for form elements such as <input>, <textarea>, <select>, etc. It helps users understand what information is expected in a specific field.

➢ **Why Use <label>?**

- Associates text descriptions with input fields.
- Improves form usability and accessibility, especially for users using screen readers or assistive technologies.
- Allows users to click the label to focus or select the corresponding input field.

➢ **How to Use <label>:**

There are two main ways to associate a label with an input:

1. Using the for Attribute (Explicit Association):

html

CopyEdit

```
<label for="email">Email Address:</label>

<input type="email" id="email" name="email">
```

Here, the for value must match the id of the input.

2. Wrapping the Input (Implicit Association):

html

CopyEdit

```
<label>

  Email Address:

  <input type="email" name="email">

</label>
```

➢ **Accessibility Benefits:**

- Screen readers can read the label when the input is focused, helping visually impaired users understand the form.
- Makes form navigation easier for keyboard users.
- Improves form clarity and user experience on all devices.

<u>HTML Tables:</u>

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

➢ An HTML table is used to **display data in rows and columns**, like a spreadsheet. Tables are made using specific tags that define the structure and content of the table.

➢ **Basic Structure of an HTML Table:**

html

CopyEdit

```
<table>
 <thead>
  <tr>
   <th>Header 1</th>
   <th>Header 2</th>
  </tr>
 </thead>
```

```
  <tr>
    <td>Row 1, Cell 1</td>
    <td>Row 1, Cell 2</td>
  </tr>
</table>
```

- **Explanation of Table Elements:**

1. <table>

- The container for the entire table.
- All table content must be placed inside this tag.

- *Purpose:*
  Wraps all table-related elements like rows, headers, and data cells.

2. <tr> (Table Row)

- Defines a **row** in the table.
- Can contain header cells (<th>) or data cells (<td>).

- *Purpose:*
  Groups a set of cells into a horizontal row.

3. <th> (Table Header)

- Defines a **header cell** in a table (usually the first row or column).
- Text is **bold** and **centered** by default.

- *Purpose:*
  Labels each column or row, improving readability and accessibility.

4. <td> (Table Data)

- Represents a **data cell** in the table.
- Appears within a <tr> and contains normal data.

- *Purpose:*
  Holds the actual content (numbers, text, links, etc.) of the table.

5. <thead>

- Defines the **header section** of a table.
- Groups header rows for styling or accessibility.

- *Purpose:*
  Organizes and separates the header rows from the main table body (<tbody>).

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

- colspan and rowspan are **HTML table cell attributes** that allow a single cell to span across **multiple columns or rows**, respectively.

1. colspan:

- Merges **multiple columns** into a single cell **horizontally**.
- Used in the <td> or <th> tag.

Example:

html

CopyEdit

```
<table border="1">
  <tr>
    <th colspan="2">Name and Age</th>
  </tr>
  <tr>
    <td>John</td>
    <td>25</td>
  </tr>
</table>
```

Here, the first row has one cell that spans two columns.

2. rowspan:

- Merges **multiple rows** into a single cell **vertically**.
- Also used in <td> or <th>.

Example:

```
html
CopyEdit
<table border="1">
  <tr>
    <th rowspan="2">Name</th>
    <td>John</td>
  </tr>
  <tr>
    <td>Doe</td>
  </tr>
</table>
```

Here, the "Name" cell spans two rows vertically.

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

➢ **Why Tables Should Be Used Sparingly for Layout:**

In the early days of web design, developers used HTML tables to create page layouts. However, this practice is now **discouraged** for several important reasons:

➢ **Disadvantages of Using Tables for Layout:**

1. **Not Semantic:**
   - Tables are meant for **tabular data**, not for designing page structure.
   - Using them for layout **confuses screen readers and search engines**.
2. **Poor Accessibility:**
   - Makes it difficult for users with assistive technologies (like screen readers) to understand content.
3. **Hard to Maintain:**
   - Table-based layouts are complex and **hard to edit or update**.
   - Small design changes may require editing multiple rows and columns.
4. **Not Responsive:**
   - Tables don't adapt well to different screen sizes like mobile or tablets.
   - Breaks user experience on smaller devices.
5. **Bloated Code:**
   - Requires a lot of nested HTML, which makes the page **slow and cluttered**.

➢ **Better Alternative: Use CSS for Layout**

Modern websites use **CSS (Cascading Style Sheets)** with semantic HTML elements for layout and styling.

➢ **Common CSS-Based Layout Methods:**

1. **Flexbox (display: flex)**
   - Great for one-dimensional layouts (rows or columns).
   - Easily handles alignment and spacing.

html

CopyEdit

```
<div style="display: flex;">
  <div>Left</div>
  <div>Right</div>
</div>
```

2. **CSS Grid (display: grid)**
   - Best for two-dimensional layouts (rows *and* columns).
   - Very powerful and clean for page structures.

html

CopyEdit

```
<div style="display: grid; grid-template-columns: 1fr 1fr;">
  <div>Header</div>
  <div>Content</div>
</div>
```

3. **Media Queries**
   - Help create **responsive designs** that adapt to screen sizes.