Module 2 – Frontend – HTML

HTML Basics

Theory Assignment

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

Purpose of HTML in Web Development:

1. **Defines the Structure of Web Pages**
   HTML is used to organize content on a web page using elements such as headings, paragraphs, lists, tables, links, and images.
2. **Creates the Skeleton of a Website**
   It acts as the **foundation** or **skeleton** for any website. All other technologies like CSS (for styling) and JavaScript (for interactivity) are built upon the HTML structure.
3. **Supports Multimedia Integration**
   HTML allows embedding of images, audio, video, and interactive content using tags like `<img>`, `<audio>`, `<video>`, and `<iframe>`.
4. **Provides Semantic Meaning**
   HTML5 introduces semantic elements like `<article>`, `<section>`, `<nav>`, and `<footer>` which help both browsers and developers understand the meaning and role of content.
5. **Links Web Pages Together**
   With the `<a>` (anchor) tag, HTML enables **hyperlinking**, which connects web pages to each other, forming the World Wide Web.

• Question 2: Explain the basic structure of an HTML document. Identify the mandatory tagsand their purposes.

📑 Mandatory Tags and Their Purposes:

Tag	Purpose
`<!DOCTYPE html>`	Declares the document type and HTML version (HTML5). It helps browsers understand how to render the page.

Tag	Purpose
`<html>`	The **root element** that wraps the entire HTML document.
`<head>`	Contains **metadata**, such as the page title, character encoding, links to CSS, etc.
`<title>`	Specifies the **title of the web page** (shown on browser tab).
`<body>`	Contains the **main content** of the web page — text, images, buttons, etc.

□ Additional Notes:

- All HTML tags are usually written in **pairs**: an **opening tag** (e.g., `<body>`) and a **closing tag** (e.g., `</body>`).
- The document must be **properly nested** and structured for it to work correctly in all browsers.

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

⍰ 1. Block-Level Elements:

- **Definition:**
  Block-level elements start on a **new line** and take up the **full width** of their container, stretching as far as they can horizontally.
- **Characteristics:**
  - Always begin on a new line
  - Occupy the full width of the parent element
  - Can contain **other block-level** and **inline** elements
- **Examples:**
  - `<div>` – Generic container
  - `<p>` – Paragraph
  - `<h1>` to `<h6>` – Headings
  - `<ul>`, `<ol>`, `<li>` – Lists
  - `<table>` – Table
  - `<section>`, `<article>` – Semantic containers
- **Example Code:**

```
html
CopyEdit
<p>This is a paragraph.</p>
<div>
  <h2>Block heading</h2>
  <p>Another paragraph inside a div.</p>
</div>
```

📝 2. Inline Elements:

- **Definition:**
  Inline elements **do not start on a new line** and only take up as much **width as necessary**.
- **Characteristics:**
  - Flows **inline with surrounding content**
  - Cannot contain block-level elements
  - Typically used for **formatting small chunks** of content
- **Examples:**
  - `<span>` – Generic inline container
  - `<a>` – Anchor (link)
  - `<strong>`, `<em>` – Emphasis
  - `<img>` – Image
  - `<input>` – Form input fields
- **Example Code:**

```
html
CopyEdit
<p>This is a <strong>bold</strong> word in a paragraph.</p>
<a href="#">Click here</a> to visit the page.
```

📊 Comparison Table:

Feature	Block-Level Elements	Inline Elements
Starts on a new line	Yes	No
Takes full width	Yes	No (just enough to fit)
Can contain	Block & inline elements	Only inline elements
Examples	`<div>`, `<p>`, `<h1>`	`<span>`, `<a>`, `<img>`

✅ Conclusion:

- **Block-level elements** define the **layout and structure** of a page.
- **Inline elements** are used for **formatting content** within block-level elements.
  Both types work together to create a well-organized and styled HTML page.

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

📘 What is Semantic HTML?

Semantic HTML refers to the use of **HTML elements that clearly describe their meaning and purpose** in the content. Instead of using generic tags like `<div>` or `<span>`, semantic tags like `<article>`, `<header>`, `<nav>`, and `<footer>` provide **contextual meaning** to both browsers and developers.

🎯 Role and Importance of Semantic HTML:

✅1. Improves Accessibility:

- **Screen readers and assistive technologies** use semantic tags to interpret content structure.
- Helps visually impaired users **navigate and understand** the page content more easily.

✅2. Boosts SEO (Search Engine Optimization):

- Search engines like Google use semantic tags to better **index and rank** pages.
- Clearly defined structure helps crawlers understand **what content is important**.

✅3. Enhances Code Readability & Maintenance:

- Semantic tags make the code **easier to read, understand, and maintain** for developers.
- It becomes obvious what each section of a webpage is meant to do.

✅4. Standardized Structure:

- Promotes a **consistent structure** across web pages, aiding in design and user experience.

☐ Examples of Semantic HTML Elements:

Tag	Description
`<header>`	Defines a page or section header

Tag	Description
`<nav>`	Defines navigation links
`<main>`	Represents the main content of the page
`<article>`	Represents a self-contained piece of content (e.g., a blog post)
`<section>`	Defines a thematic grouping of content
`<aside>`	Contains side content (e.g., ads, sidebars)
`<footer>`	Represents the footer of a page or section
`<figure>` and `<figcaption>`	For images with captions

□ Example Code:

```
html
CopyEdit
<!DOCTYPE html>
<html>
  <body>
    <header>
      <h1>My Blog</h1>
    </header>

    <nav>
      <a href="#home">Home</a> |
      <a href="#articles">Articles</a>
    </nav>

    <main>
      <article>
        <h2>Post Title</h2>
        <p>This is a blog post.</p>
      </article>
    </main>

    <footer>
      <p>© 2025 My Blog. All rights reserved.</p>
    </footer>
  </body>
</html>
```

□ Conclusion:

Semantic HTML plays a **crucial role** in making web pages **accessible, SEO-friendly, and developer-friendly**. It provides **meaning and structure** to content, helping both users and machines understand the page better.

HTML Forms

Theory Assignment

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

What are HTML Forms Used For?

HTML forms are used to **collect data** from users and **send it to a server** for processing. They are commonly used in:

- **Login and registration forms**
- **Search bars**
- **Surveys and feedback forms**
- **Order and payment forms**

Forms are defined using the `<form>` tag, which wraps around various **form controls** like input fields, buttons, and dropdowns.

Key Form Elements and Their Purposes:

1. `<input>`

- Used to **accept single-line user input** such as:
  - Text
  - Numbers
  - Dates
  - Passwords
  - Checkboxes, radio buttons, etc.
- Controlled by the `type` attribute.

Examples:

```
html
CopyEdit
<input type="text" name="username" placeholder="Enter your name">
<input type="password" name="password">
<input type="checkbox" name="subscribe">
```

---

2. `<textarea>`

- Used for **multi-line text input**, such as:
  - Comments

    - Feedback
    - Descriptions

Example:

```
html
CopyEdit
<textarea name="message" rows="5" cols="30" placeholder="Write your message
here"></textarea>
```

---

3. `<select>`

- Used to create a **dropdown list** of options.
- Contains nested `<option>` tags to define the list items.

Example:

```
html
CopyEdit
<select name="country">
  <option value="india">India</option>
  <option value="usa">USA</option>
  <option value="uk">UK</option>
</select>
```

---

4. `<button>`

- Used to **submit** the form or perform actions like **resetting** or triggering JavaScript functions.
- Can be of types: `submit`, `reset`, or `button`.

Example:

```
html
CopyEdit
<button type="submit">Submit</button>
<button type="reset">Clear</button>
```

□ Conclusion:

HTML forms are essential for **interacting with users** and collecting information. The elements like `<input>`, `<textarea>`, `<select>`, and `<button>` allow users to **enter and submit data** in various formats, making web applications functional and interactive.

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

🔁 GET vs POST in Form Submission

When submitting a form in HTML using the `<form>` tag, the `method` attribute specifies **how** the form data is sent to the server. The two most common methods are **GET** and **POST**.

🔍 GET Method:

- **Sends data through the URL** as query parameters.
- Example:

  ```
  arduino
  CopyEdit
  https://example.com/form?name=John&age=25
  ```

- **Characteristics:**
  - Appends form data to the **URL**.
  - **Limited in size** (around 2000 characters).
  - **Not secure** for sensitive data (visible in URL).
  - Can be **bookmarked** and **cached** by browsers.
- **When to use GET:**
  - For **non-sensitive data**.
  - When the form is used for **search**, **filtering**, or **navigation**.
  - When you want the result page to be **shareable or bookmarkable**.

Example:

```
html
CopyEdit
<form action="/search" method="get">
  <input type="text" name="query" placeholder="Search">
  <button type="submit">Search</button>
</form>
```

🔐 POST Method:

- **Sends data in the body** of the HTTP request.
- URL remains clean (no data in it).
- **Characteristics:**
  - **Does not display data in the URL**.
  - No size limitation on form data.
  - **More secure** for sensitive information (like passwords).
  - Not cached or bookmarked.
- **When to use POST:**
  - For **sensitive or private data** (e.g., login forms).

  - When **uploading files or large amounts of data**.
  - For **actions that change data on the server**, like submitting a form to add a new record.

Example:

```
html
CopyEdit
<form action="/login" method="post">
  <input type="text" name="username">
  <input type="password" name="password">
  <button type="submit">Login</button>
</form>
```

📊 Comparison Table:

Feature	GET	POST
Data sent in	URL (query string)	HTTP request body
Data visibility	Visible in URL	Hidden from URL
Data length	Limited	Unlimited
Security	Less secure	More secure
Bookmarkable	Yes	No
Use case	Search, filtering	Login, file upload, forms

✅ Conclusion:

- Use **GET** for simple, safe requests where data can be exposed (e.g., search forms).
- Use **POST** for secure or large data submissions (e.g., login, registration, payments). Choosing the right method ensures both **security and performance** of your web application.

• 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 in HTML is used to **define a text description for a form input field**, such as a textbox, checkbox, radio button, etc. It helps users understand **what each input field is for**.

🔗 How `<label>` Works:

There are **two ways** to use the `<label>` element:

1. **Using the `for` attribute** (linked by `id`):

```
html
CopyEdit
<label for="email">Email Address:</label>
<input type="email" id="email" name="email">
```

2. **Wrapping the input field:**

```
html
CopyEdit
<label>
  Email Address:
  <input type="email" name="email">
</label>
```

Both methods associate the label with the input field, but the `for` attribute method is more common and clearer.

♿ Accessibility Benefits:

1. **Helps Screen Readers:**
   Screen readers use the label to **announce the purpose** of the input field to visually impaired users.
2. **Larger Click Area:**
   Clicking the label also activates the input (like selecting a checkbox), which improves usability, especially on mobile devices.
3. **Improves Keyboard Navigation:**
   Keyboard users (tabbing through inputs) benefit from clear labels that **describe each form field**.
4. **Better User Experience:**
   Users understand what data is expected, reducing input errors.

✅ Conclusion:

The `<label>` element is crucial for **form usability and accessibility**. It connects form fields to meaningful descriptions, helping all users — especially those using assistive technologies — to navigate and fill out forms more easily and accurately.

HTML Tables

Theory Assignment

• Question 1: Explain the structure of an HTML table and the purpose of each of the following elements: <TABLE>, <TR>, <TH>, <TD> and <THEAD>.

Structure of an HTML Table:

An **HTML table** is used to **display data in rows and columns**. The structure is made up of several tags that define the layout and contents of the table.

Key HTML Table Elements and Their Purposes:

Tag	Full Form	Purpose
`<table>`	Table	The **container** element for all table content. It defines the start and end of the table.
`<tr>`	Table Row	Defines a **row** in the table. Each `<tr>` contains cells (`<th>` or `<td>`).
`<th>`	Table Header	Represents a **header cell** in a table row. Text is usually **bold and centered**. Used to label columns or rows.
`<td>`	Table Data	Represents a **standard data cell** in a table row. Contains actual data or content.
`<thead>`	Table Head	Groups the **header row(s)** of a table. It helps structure tables and can improve styling and accessibility.

Example of a Simple Table:

```
html
CopyEdit
<table border="1">
  <thead>
    <tr>
      <th>Name</th>
      <th>Age</th>
      <th>City</th>
    </tr>
  </thead>
  <tr>
    <td>John</td>
    <td>25</td>
    <td>New York</td>
  </tr>
  <tr>
    <td>Emma</td>
    <td>30</td>
    <td>London</td>
  </tr>
</table>
```

📘 Explanation of Example:

- `<table>`: Starts the table.
- `<thead>`: Groups the header row.
- `<tr>`: Defines each row.
- `<th>`: Defines header cells: **Name**, **Age**, **City**.
- `<td>`: Defines data cells: "John", "25", "New York", etc.

✅ Conclusion:

HTML tables are structured using a combination of tags that define the table, rows, headers, and data cells. Each element—`<table>`, `<tr>`, `<th>`, `<td>`, and `<thead>`—has a specific role in organizing and displaying tabular data clearly and accessibly.

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

🔄 Definition and Purpose:

Both `**colspan**` and `**rowspan**` are **HTML table attributes** used within `<td>` or `<th>` elements to make a cell **span across multiple columns or rows**.

📐 Difference Between `colspan` and `rowspan`:

Attribute	Purpose	Affects
`colspan`	Merges a cell **horizontally** across two or more **columns**	Columns
`rowspan`	Merges a cell **vertically** across two or more **rows**	Rows

---

🏛 1. Example of `colspan`:

```
html
CopyEdit
<table border="1">
```

```
  <tr>
    <th colspan="2">Name & Age</th>
    <th>City</th>
  </tr>
  <tr>
    <td>John</td>
    <td>25</td>
    <td>New York</td>
  </tr>
</table>
```

 Explanation:

- The cell **"Name & Age"** spans **2 columns** — it takes up the space of both "Name" and "Age" columns.

📊 2. Example of `rowspan`:

```
html
CopyEdit
<table border="1">
  <tr>
    <th rowspan="2">Name</th>
    <th>Age</th>
    <th>City</th>
  </tr>
  <tr>
    <td>25</td>
    <td>New York</td>
  </tr>
</table>
```

 Explanation:

- The cell **"Name"** spans **2 rows**, aligning vertically with both the "Age" and "City" rows.

✅ Conclusion:

- Use **`colspan`** to make a cell stretch **across multiple columns** (left to right).
- Use **`rowspan`** to make a cell stretch **across multiple rows** (top to bottom).
  These attributes are helpful in formatting tables for **grouped data** or creating **complex table layouts**.

• 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, HTML tables were commonly used to create page layouts. However, this practice is now discouraged for several important reasons:

⊘ Problems with Using Tables for Layout:

1. **Not Semantic:**
   - Tables are meant to display **tabular data**, not to structure pages.
   - Using them for layout confuses both **developers** and **assistive technologies**.
2. **Poor Accessibility:**
   - Screen readers expect tables to contain data. Layout tables can make **navigation difficult** for users with disabilities.
3. **Difficult to Maintain:**
   - Layouts using nested tables are often **complicated and hard to update** or redesign.
4. **Lack of Flexibility:**
   - Tables are **not responsive** by default and don't adapt well to different screen sizes like mobile devices.
5. **Slower Page Load:**
   - Complex table layouts can cause **browsers to render more slowly** than modern layout methods.

✓ Better Alternative: Use CSS with Semantic HTML

Modern web development uses **CSS (Cascading Style Sheets)** for layout design, combined with **semantic HTML** elements like `<div>`, `<section>`, `<article>`, etc.

🔧 Modern Layout Techniques with CSS:

Technique	Description
Flexbox	For 1D layouts — arranging items in a row or column easily.
Grid	For 2D layouts — creating complex, responsive designs.
Media Queries	Used to make layouts responsive across screen sizes.

□ Example Using CSS Flexbox (Instead of Table for Layout):

```
html
CopyEdit
<style>
  .container {
    display: flex;
    justify-content: space-between;
  }
</style>

<div class="container">
  <div>Column 1</div>
  <div>Column 2</div>
  <div>Column 3</div>
</div>
```

✓ Conclusion:

Tables should only be used for displaying tabular data.
For layouts, use **CSS with semantic HTML** — this results in more **accessible**, **maintainable**, and **responsive** websites that follow modern web standards.