THEORY ASSIGNMENT Module 2 – Mernstack – HTML

▪ HTML Basics

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

- HTML (HYPERTEXT MARKUP LANGUAGE) IS THE STANDARD MARKUP LANGUAGE USED TO CREATE AND STRUCTURE WEB PAGES. IT FORMS THE BACKBONE OF ALL WEBSITES BY DEFINING THE LAYOUT, CONTENT, AND BASIC FUNCTIONALITY OF A WEBPAGE.
- Key Features:
  - HyperText – Allows linking between pages (via <a> tags).
  - Markup Language – Uses tags to define elements (e.g., <p> for paragraphs).
  - Foundation of the Web – Works with CSS (styling) and JavaScript (interactivity).
- Purpose Of Html:
  - ❖ Builds the Structure
    - Like building a house frame before decorating Creates headings, paragraphs, lists (like a document outline)
  - ❖ Holds All the Content
    - Adds text, images, videos (like putting pictures in a photo album)
    - Makes buttons and forms (like creating boxes to type in)
  - ❖ Connects Everything
    - Creates clickable links (like bridges between pages)
    - Helps phones/computers show the site properly
  - ❖ Works with Helpers
    - CSS (makes it pretty - like paint and furniture)
    - JavaScript (makes it interactive - like light switches)

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

- An HTML document has a standard structure with mandatory tags that define its layout:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Page Title</title>
</head>
<body>
   <!-- Content goes here -->
</body>
</html>
```

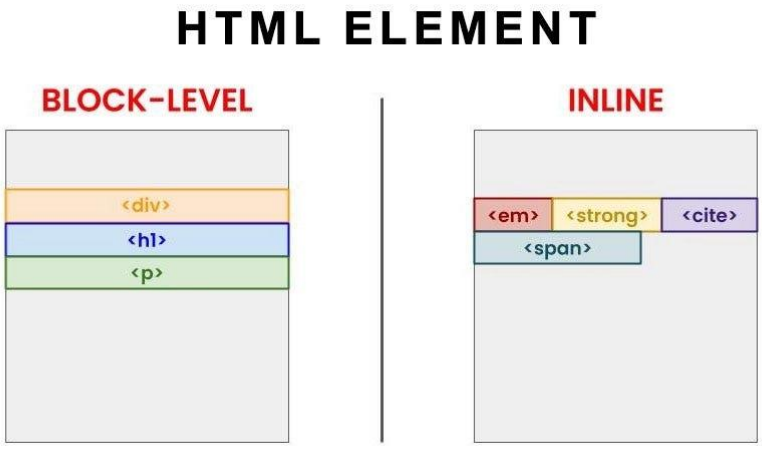
- Mandatory & Essential Tags Explained:
  - <!DOCTYPE html>
    - Must be first line; declares HTML5 document type.
  - <html lang="en">
    - Root element with language attribute (e.g., en for English).
  - <head> Section (Required)
    - <meta charset="UTF-8">
      - Ensures proper text encoding (supports all languages).
    - <meta name="viewport">
      - Makes site mobile-friendly (responsive design).
    - <title>
      - Sets the browser tab title (SEO required).
  - <body>
    - Contains all visible content (required for display).

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

- Difference between block-level elements and inline elements in html

Feature	Block-Level Elements	Inline Elements
Default Behaviour	Starts on a new line (takes full width)	Flows within text (takes only needed width)
Width	Stretches to 100% of parent width	Width matches content
Height	Respects height/width properties	Ignores height/width (unless changed to inline-block)
Margin/Padding	Supports all margins/paddings	Only horizontal margins/paddings work (vertical has no effect on layout)
Nesting Rules	Can contain both block & inline elements	Can only contain other inline elements (with exceptions like <a>)
Example Tags	, <p>, <h1>-<h6>, , , <section>	, <a>, , , <img>,

- Example:



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

- Semantic HTML refers to the use of HTML elements that clearly describe their meaning to both browsers and developers, making web content more understandable for machines and humans alike. These elements provide meaningful context about the structure of a webpage, which is crucial for accessibility and SEO.
- For accessibility, semantic HTML helps screen readers and assistive technologies interpret and navigate content more effectively, ensuring a better experience for users with disabilities.
- For SEO, search engines rely on semantic markup to better understand the content and hierarchy of a page, improving its ranking in search results.
- Examples of semantic elements include <header>, which defines the introductory section of a page; <nav>, used for navigation links; <article>, representing self-contained content like blog posts; <section>, grouping related content; and <footer>, marking the end of a page or section. By using these elements, developers create more organized, accessible, and search-engine-friendly websites.
- Additional examples of semantic elements include:
  - <figure> and <figcaption>: Used for images, diagrams, or charts with descriptive captions, improving context for both users and search engines.
  - <time>: Marks up dates and times in a machine-readable format (e.g., <time datetime="2023-10-05">October 5, 2023</time>), aiding event indexing.
  - <mark>: Highlights text for reference or notation, useful for key terms.
  - <details> and <summary>: Create expandable sections, enhancing usability without JavaScript.
  - <address>: Identifies contact information, signaling its importance to browsers.

- <time>: Marks up dates and times in a machine-readable format (e.g., <time datetime="2023-10-05">October 5, 2023</time>), aiding event indexing.
-

- **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 on websites, enabling interactions like logins, registrations, surveys, searches, and payment processes. They serve as a bridge between users and web applications by submitting data to servers for processing.
- Key Form Elements and Their Purposes:
  - <input>
    - The most versatile form element, used for various data types.
    - Examples:
      - Text fields (<input type="text">)
      - Email/URL inputs (<input type="email">, <input type="url">)
      - Checkboxes (<input type="checkbox">)
      - Radio buttons (<input type="radio">)
      - File uploads (<input type="file">)
  - <textarea>
    - Allows multi-line text input (e.g., comments, messages).
    - Supports customizable dimensions (rows and cols attributes).
  - <select>
    - Creates dropdown menus for single/multiple selections.
    - Paired with <option> tags to define choices:

```
<select name="country">
  <option value="us">United States</option>
  <option value="ca">Canada</option>
```

  - <button>
    - Triggers actions like form submission (type="submit"), resets (type="reset"), or custom scripts (type="button").
    - More flexible than <input type="button">

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

➢ Difference between GET and POST methods in Form submission

Feature	Get Method	Post Method
Data Visibility	Appears in the URL (?name=value)	Hidden in the HTTP request body
Data Length	Limited (~2048 chars)	No fixed limit
Caching	Can be cached/bookmarked	Never cached
Security	Less secure (exposed in URLs)	More secure (not in URLs)
Idempotent	Yes (repeatable safely)	No (can change server state)

➢ When to Use Each

- Use GET When:
  - Fetching Data (e.g., search queries, filters).
  - Example: Google searches (?q=term).
  - Safe Operations (no server-side changes).
  - Example: Loading a product page (/product?id=123).
  - Bookmarkable URLs (e.g., shared links).
- Use POST When:
  - Sending Sensitive Data (e.g., passwords, credit cards).
  - Example: Login forms.
  - Modifying Server Data (e.g., creating/updating records).

- Example: Submitting a contact form.

- **Large Data Uploads (e.g., file uploads, long messages).**

➢ **Example:**

```
<!-- GET -->
<form method="GET" action="/search">...</form>

<!-- POST -->
<form method="POST" action="/login">...</form>
```

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

- The label element in an HTML form provides a caption for a form control, making it easier for users to understand what information is required for each field. It improves accessibility by associating the label text with the form element, making it more understandable for assistive technology users, such as screen readers. Clicking the label also focuses the associated input, improving usability, especially for users with motor impairments.
- Example with Accessibility Benefits:
    - How it Improves Accessibility:
        - Screen Reader Compatibility:
            - Screen readers read the label text aloud when the user is focused on the associated form element, providing context and information about what is expected in the field.
        - Enhanced Understanding:
            - By providing clear and concise labels, users, including those with cognitive disabilities, can better understand what information is required for each form element.
        - Accessibility Compliance:
            - The use of label elements is crucial for meeting accessibility standards, such as WCAG (Web Content Accessibility Guidelines).
        - Increased Accessibility:
            - Proper association of labels with form controls improves usability for all users, regardless of their abilities.
    - Example:
- Code

```
<label for="firstName">First Name:</label>
<input type="text" id="firstName" name="firstName">
```

In this example, the label "First Name:" is associated with the input field with the ID "firstName". When a user focuses on the input field, a screen reader will read "First Name:". Clicking the label will also focus the input field, making it easier for users to enter information.

- **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>.

- An HTML table is structured using the <table> tag as the main container, rows (<tr>), header cells (<th>), and data cells (<td>). The <thead> tag is used to group header rows.
- <table>:
  Defines the table itself. It's the outermost tag that contains all other table elements.
- <tr>:
  Defines a row within the table. It's used to structure the table into horizontal lines.
- <th>:
  Defines a table header cell. Header cells typically contain the column or row headings.
- <td>:
  Defines a standard table cell. It's used to hold the data within the table.
- <thead>:
  Groups the header rows within the table. This is often used for styling the header and is important for larger tables that might be split across multiple pages when printed

- Example:

```
<table>
  <thead>
    <tr>
      <th>Header 1</th>
      <th>Header 2</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Data 1</td>
      <td>Data 2</td>
    </tr>
  </tbody>
</table>
```

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

- In HTML tables, colspan allows a cell to span multiple columns horizontally, while rowspan allows a cell to span multiple rows vertically. Colspan essentially merges columns, and rowspan merges rows.
- Colspan:
  Merges columns horizontally, combining multiple columns into one cell.
  Used to create wider cells that span across multiple columns, like a header that covers several columns.
- Rowspan:
  Merges rows vertically, combining multiple rows into one cell.
  Used to create taller cells that span across multiple rows, often for labels or headers that apply to several rows.
- EXAMPLE 1 Colspan:

```
<table>
 <tr>
  <td colspan="2">This cell spans 2 columns</td>  <!-- colspan=2 -->
 </tr>
 <tr>
  <td>Regular cell</td>
  <td>Regular cell</td>
 </tr>
</table>
```

- Example 2 Rowspan:

```
<table>
 <tr>
  <td rowspan="2">This cell spans 2 rows</td> <!-- rowspan=2 -->
  <td>Regular cell</td>
 </tr>
 <tr>
  <td>Regular cell</td>
 </tr>
</table>
```

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

- Tables should be used sparingly for layout because they are primarily designed for presenting tabular data, not for structuring web page layouts. Using tables for layout is discouraged due to issues with flexibility, accessibility, and semantic structure. A better alternative is to use CSS for layout, which offers more flexibility, semantic control, and accessibility benefits.
- While HTML tables are ideal for displaying structured data, they pose several drawbacks when misused for page layout: their inflexibility makes them poorly responsive across screen sizes, accessibility suffers as screen readers struggle to interpret non-data tables, SEO is impacted due to obscured content hierarchy, maintenance becomes cumbersome with nested complexity, and they violate semantic HTML principles since tables are designed for tabular data—not visual structuring. For modern web design, CSS Flexbox/Grid should replace layout tables entirely.

- **The reasons to avoid using tables for layout in HTML are listed below:**
    - **Tables Are Not Accessible:** Most search engines read the webpage as they read HTML and it becomes difficult for the search engine to render the table layout. This is the main reason why we follow the HTML5 format.
    - **Tables Are Tricky:** When you perform nesting in tables then it is difficult to maintain it. When you want to change something after some days then it will become complicated for the developer to debug the code.
    - **Tables Are Inflexible:** When you want to create the table layout with specified widths then it will become a rigid layout or not flexible and then it will take some extra time to load your page properly. The flexible layout always looks good on any device.
    - **Tables Hurt Search Engine Optimization:** Many developers create the navigation on the left-hand side and the rest of the content on the right side. If you use tables the search engine will load the content first then the navigation will start to load without navigation, the content will look not so good.
    - **Tables Don't Always Print Well:** When you try to print the table layout the printer will change the interface because the table layout

is too wide. The printer will then try to cut down some content or show extra content to the next page which will make it complicated.

- **Tables for Layout Are Invalid in HTML 4.01:** You can't create the table layout when you use HTML 4.01 because you can only be allowed to create a simple table. For example, spreadsheets or databases. Another reason is that other browsers find it tough to render through the table layout.

➢ Alternate to table for layout purpose can be CSS:

- CSS offers significant advantages over tables for modern web layouts by enabling flexible, responsive designs through features like Flexbox and Grid. Unlike rigid tables, CSS separates content from presentation, ensuring cleaner semantic markup that improves accessibility for screen readers and enhances SEO. It also simplifies maintenance, as CSS rules are easier to update than nested table structures, while providing dynamic adaptability across devices. Tables should be reserved strictly for tabular data, while CSS handles layout more efficiently and accessibly.