HTML in Full Stack

1. HTML Basics

❖ Theory Assignment

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

Ans: HTML stand for HyperText Markup language. HTML is the standard markup language for creating Web pages. HTML is used to create web pages and web applications. HTML describes the structure of a Web page. We can create a static website by HTML only. HTML consists of a series of elements. HTML consists of a series of elements. An HTML element is defined by a start tag, some content, and an end tag <tagname> Content </tagname>.

❖ Purpose of HTML in Web Development:

- **Content Structure:**
  - HTML provides the basic structure of a webpage, organizing elements like headings, paragraphs, lists, images, links, and more.
- **Webpage Layout:**
  - HTML helps create the layout of web pages by using containers such as , <header>, <footer>, <section>, etc., which help in organizing the content effectively.
- **Linking:**
  - HTML enables the creation of hyperlinks using the <a> tag, allowing navigation between different pages and external resources.
- **Embedding Multimedia:**
  - It allows embedding images, videos, and audio files into web pages using tags like <img>, <video>, and <audio>.
- **User Interaction:**
  - HTML supports forms (using <form>, <input>, <button>, etc.) that enable users to interact with websites, like submitting data or signing up.
- **Semantics:**

- HTML includes semantic tags like <article>, <header>, and <footer>, helping search engines and other user agents understand the structure and meaning of content.

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 a series of elements (tags) that define the content, structure, and metadata of the webpage. HTML document with the mandatory tags and their purposes:

❖ Basic Structure of an HTML Document:

```
<!DOCTYPE html>
<html>
 <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
        scale=1.0">
        <title>Page Title</title>
 </head>
 <body>
        <h1>Welcome to My Website</h1>
         <p>This is a paragraph of text.</p>
 </body>
</html>
```

❖ Explanation of Mandatory Tags and Their Purposes:

1. **<!DOCTYPE html>:**
   - **Purpose:** This declaration defines the document type and version of HTML being used. For modern web pages, <!DOCTYPE html> is used to specify HTML5. It helps the browser render the page correctly.
   - **Mandatory**: Yes, This is the very first line of the HTML document.
2. **<html>:**
   - **Purpose:** The root element of the HTML document. It wraps all the content of the webpage.

- **Mandatory**: You can include attributes like lang="en" to specify the language of the document.

3. **<head>:**
   - **Purpose:** Contains meta-information about the document that is not directly visible on the web page itself essential for functionality and SEO., This can include things like the document's character encoding,scripts, links to stylesheets, and metadata .
   - **Mandatory**: **Yes, <meta charset="UTF-8">**: Specifies the character encoding of the document. UTF-8 is the most common encoding and supports a wide range of characters.
   - **<title>:** Defines the title of the document, which is displayed on the browser's title bar or tab.
4. **<body**>:
   - **Purpose:**Contains the visible content of the web page, including text,images, links, and other media.
   - **Mandatory**: Yes, Everything visible on the webpage goes inside the <body> tag.

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

Ans:

❖ **<u>Difference Between Block-level Elements And Inline Elements</u>**

Block-level Elements	Inline Elements
Block elements always start in a new line.	Inline elements don’t start in a new line
A block-level element always takes up the full width available (Left to right).	Inline elements only cover the space as bounded by the tags in the HTML element.
Block elements have top and bottom margins.	Inline elements don't have a top and bottom margin.
Create line breaks before or after the element.	Do not create line breaks before or after the element.

Examples of block elements - <p> <hr> <hr> <h1>, <table><h1>to <h6>.	Examples of inline elements - , <a>,<img><input>.

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

Ans: Semantic HTML is an approach in web development that uses special HTML tags to give clear structure and meaning to your content.

❖ Importance of Semantic HTML for Accessibility:

1. **Screen Readers & Assistive Technologies:**
   - Screen readers rely on semantic elements to interpret and describe content to users with visual impairments. For example, a <header> element indicates that it contains the main heading of a page or section, while a <nav> element signifies a navigation area. This structure helps users navigate and understand the page more easily.
2. **Keyboard Navigation:**
   - Properly structured HTML elements provide better support for users who rely on keyboard navigation. Semantic HTML helps screen readers announce important sections and headings, making it easier to jump between content.
3. **Content Structuring:**
   - Elements like <article>, <section>, <footer>, and <aside> give more context about the purpose of different parts of the page. This context improves accessibility for both assistive technologies and human users who need to focus on certain content areas.

❖ Importance of Semantic HTML for SEO:

1. **Improved Search Engine Crawling:**
   - Search engines like Google use semantic HTML to better understand the content of a page. For instance, a <main> element indicates the primary content of the page, which can help search engines prioritize and rank this content.
2. **Better Indexing:**

- Proper use of semantic tags such as <h1>, <h2>, <article>, and <section> helps search engines understand the hierarchy and structure of the content. This leads to more effective indexing and ranking.

3. **Enhanced Relevance:**
   - Search engines reward content that is properly structured and easy to parse. Using tags like for emphasizing important text or <time> for dates can help search engines determine the relevance of the content.

❖ Examples of Semantic HTML Elements:

- **<header>:**
  - Defines the header section of a document or a section. Typically contains introductory content or navigational links.
- **<nav>:**
  - Defines navigation links. This tag is important for search engines and accessibility tools to identify the navigation portion of the page.
- **<article>:**
  - Represents a self-contained piece of content that could be distributed or reused. It could be a blog post, news article, or forum post.
- **<section>:**
  - Defines sections in a document, typically for grouping content by topic. This helps to structure a page into meaningful sections.
- **<footer>:**
  - Defines the footer of a document or section, often containing contact information, copyright, or related links.
- **<aside>:**
  - Represents content that's indirectly related to the main content, often used for sidebars, quotes, or tangential information.
- **<main>:**
  - Indicates the primary content of the document. It helps search engines and assistive technologies identify the most important part of the page.

2. HTML Form:

❖ Theory Assignment

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

Ans: HTML forms are used to collect user input and send that data to a server for processing. They allow interaction between a user and a web page, such as submitting information through contact forms, login forms, search boxes, or even purchasing products.

❖ Purpose of Specific HTML Form Elements:

1. Input Element(<input>):

- The <input> tag in HTML is used to collect user input in web forms. It supports various input types such as text, password, checkboxes, radio buttons, and more.
- It is used to get input data from the form in various types such as text, password, email, etc by changing its type.
- **Example:**

```
<!DOCTYPE html>
<html>
        <head>
          <title>Input Element   Example</title>
       </head>
<body style="text-align:center;">
         <h2>Login Form h2>
   <form>
                <label for="uname">User Name:</label>
                <input type="text" name="uname"><br><br>
                <label for="pwd">Password:</label>
                <input type="password" name="pwd"><br><br>
                <input type="submit" value="Submit">
    </form>
 </body>
</html>
```

❖ Common <input> Types:

Input Type	Description
<input type=”text”>	Single-line text input
<input type=”password”>	Use to collect password information securely in HTML forms. This masks the entered text for privacy.
<input type=”checkbox”>	Toggle for selecting multiple options
<input type=”radio”>	Single selection from multiple options
<input type=”submit”>	Button to submit form data
<input type=”button”>	General-purpose button
<input type=”file”>	Input for uploading files
<input type=”number”>	Input for numerical values
<input type=”date”>	Input for selecting dates
<input type=”email”>	Input for email addresses
<input type=”color”>	Input for selecting colors
<input type=”image”>	Input using an image for form submission
<input type=”range”>	Input using an image for form submission

2. Textarea Element(<textarea>):

- The <textarea> element is used for multi-line text input, allowing users to enter longer text, such as messages, comments, or descriptions.It is used to get input long text content.
- It doesn't have a type attribute like <input>, but it has attributes like rows and cols to define its size.
- **Example:**

```
<!DOCTYPE html>
<html>
    <head>
      <title>Textarea  Example</title>
    </head>
    <body>
```

```
    <h2>Example of Textarea</h2>
    <p>Please enter your comments:</p>
   <textarea name="userComments" rows="4" cols="50">
    Enter your text here...
   </textarea>
        <br>
  </body>
</html>
```

3. Select Element(<select>):

- The <select> element is used to create a dropdown list, letting users select one or more options from a list of predefined values. It is used to create a drop-down list. the id and name attributes defines the Drop-down list
- The HTML <select> tag is used to create a drop-down list for user input,, containing <option> tags to display the available choices. It provides functionality for selecting one or multiple options from a list.
- **Example:**

```
<!Doctype html>
<html>
    <head>
        <title>Select Element Example</title>
     </head>
<body>
  <form>
    <label for="country">-----Select your Country------</label>
    <select id="country" name="countries">
      <option value="INDIA">INDIA</option>
      <option value="RUSSIA">RUSSIA</option>
      <option value="JAPAN">JAPAN</option>
    </select>
  </form>
</body>
</html>
```

4. Button Element(<button>):

- The <button> element is used to create clickable buttons, often to submit a form or trigger some action like resetting the form, opening a new window, or executing a function in JavaScript.
- It can have a type attribute to specify whether it should submit the form (type="submit"), reset the form (type="reset"), or trigger a custom function (type="button"). It defines a clickable button to control other elements or execute a functionality.
- **Example:**

```
<!Doctype html>
<html>
        <head>
                <title> Example 0f Button</title>
              <script>
                      function msg() {
                              alert(“Value Submited”);
                              }
           </script>
        </head>
<body style = "text-align:center;">
        <h1 style = "color:green;" >  Web Development </h1>
        <h2>Using Input type</h2>
   <form>
           <label for="submit">Enter the Value:</label>
           <input type = "text">
          <input type="button" id="submit" onclick ="msg()"
        value="button">
        <p> The Input button properly work</p>
   </form>
</body>
</html>
```

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 in Form

Features	GET Method	Post Method
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL.
Data Length	Limited by the URL length, which can vary by browser and server. Typically, around 2048 characters.	No inherent limit on data size, allowing for large amounts of data to be sent.
Security	Less secure due to visibility in the URL. Sensitive data can be easily exposed in browser history, server logs, etc.	More secure for transmitting sensitive data since it's not exposed in the URL.
Use Case	Ideal for simple data retrieval where the data can be bookmarked or shared.	Suited for transactions that result in a change on the server, such as updating data, submitting forms, and uploading files.
Cached	GET method requests can be cached	Post method requests are never cached
Bookmark	GET method requests can be bookmarked	POST method requests cannot be bookmarked
Restrictions	Get method restriction on datatype only ASCII characters allowed. Non-ASCII characters must be encoded.	No restrictions. Binary data is also allowed
Data in URL	GET sends form data as part of the URL and Information is visible in the browser's address bar.	POST sends form data in the body of the HTTP request and Information is not visible in the URL.
Modify Data	GET requests are only used to request data (not modify)	POST requests can be used to create and modify data.
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Encoding Type	In GET method, the Encoding type is *application/x-www-form-urlencoded*	In POST method, the encoding type is *application/x-www-form-*

		urlencoded or *multipart/form-data*. Use multipart encoding for binary data
Back/Forward Buttons	Reloading a page requested by GET does not usually require browser confirmation.	Reloading a page can cause the browser to prompt the user for confirmation to resubmit the POST request.
Impact on Server	Generally used for retrieving data without any side effects on the server.	Often causes a change in server state (Ex. database updates) or side effects.

❖ When should each be used:

1. GET Method:

- Use GET when the form is used to retrieve or search for data, such as a search query, filtering results, or navigating through different parameters on a page.
- Use GET for forms that do **not** involve sensitive information (such as login credentials or personal details).
- GET is best for small amounts of data, as the information is sent in the URL, and browsers limit the amount of data that can be sent (usually around 2048 characters).
- GET requests are only used to request data (not modify). GET method requests can be bookmarked

2. POST Method:

- Always use POST when submitting sensitive information like passwords, payment details, or personal data, as POST hides the data from the URL.
- Use POST when the form submission results in a change to the server's data (like creating a new account, updating user info, deleting an item, etc.). POST is typically used for operations that involve **side effects** on the server, such as modifying or creating resources.
- No inherent limit on data size, allowing for large amounts of data to be sent.
- No restrictions, Binary data is also allowed.

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

Ans: The **<label>** element in a form is used to provide a descriptive text label for form controls (such as text inputs, checkboxes, radio buttons, and selects). Its primary purpose is to clarify what information is expected from the user when filling out a form.

❖ Purpose of the <label> element:

1. **Describes form elements:** The <label> clearly defines what the corresponding form field is for. For example, in a registration form, you might have a label like "Email Address" next to an input field where the user should enter their email.
2. **Associates labels with form controls:** The **"for"** attribute of the **<label>** element associates it with a specific input element by matching the **"id "**of the input field. This creates a direct connection between the label and the input field, making it easier to understand the relationship.

❖ **Example:** <label for="username">Username:</label>

<input type="text" id="username" name="username">

❖ How It Improves Accessibility:

- **Screen Reader Support:** For users with visual impairments, screen readers are used to read the content of web pages aloud. When a form control has a <label>, the screen reader will announce the label text when the user navigates to that field, providing them with a clear description of what information is required. Without a label, the screen reader might not give any context for what the input is for.
- **Clickable Areas:** When a label is correctly linked to an input field, clicking on the label itself will focus on the input element. This is especially useful for people with motor impairments or those using devices like touchscreens, as it makes it easier to select the input field.
- **Keyboard Navigation**: Associating labels with form fields ensures that users who navigate via the keyboard (using the Tab key, for instance) can easily understand and interact with form fields. The labels make it clear which field is selected, improving the user experience.

3. HTML Table:

❖ Theory Assignment

1: Explain the structure of an HTML table and the purpose of each of the following elements:

Ans: HTML Tables allow you to arrange data into rows and columns on a web page, making it easy to display information like schedules, statistics, or other structured data in a clear format.

❖ Structure of an HTML Table:

- The general structure of an HTML table involves several key elements:

1. **<table>:** This is the container element that holds the entire table structure.
2. **<tr> (Table Row):** This element is used to define a row in the table. It groups the content of a row together.
3. **<th> (Table Header):** This element defines the header cell in a table. It is typically bold and centered by default. It is used for headings that describe the content of each column.
4. **<td> (Table Data)**: This element defines a cell in the table that contains data (not a header). It represents the individual pieces of content in the table.
5. **<thead>:** This element groups the header content in the table. It helps to organize the layout and can be styled separately from the rest of the table.
6. **<tbody>:** This element groups the body content of the table (the main data). It's used to wrap the rows that contain the data itself.
7. **<tfoot>**: This element groups the footer content of the table, typically used for summary information or totals.

❖ Example:

```
<!DOCTYPE html>
<html>
        <head>
            <title>Example of Table Element</title>
        </head>
    <body>
        <table>
```

```
        <tr>
            <th>Firstname</th>
            <th>Lastname</th>
            <th>Age</th>
        </tr>
        <tr>
            <td>Narendra</td>
            <td>Modi</td>
            <td>73</td>
        </tr>
        <tr>
            <td>Manan</td>
            <td>Singh</td>
            <td>29</td>
        </tr>
        <tr>
            <td>Inayat</td>
            <td>Modi</td>
            <td>26</td>
        </tr>
    </table>
</body>
</html>
```

❖ **Purpose of Each Element:**

- **<table>:** It defines the table itself, creating the overall grid for organizing data.
- **<tr>:** It creates individual rows in the table, which are used to contain header or data cells.
- **<th>:** It defines the header cells, typically used to label the columns of the table and make the table more readable.
- **<td>:** It defines the actual data in the table, representing each individual piece of information inside a row.
- **<thead>**: It groups the header rows, providing semantic structure to the table and allowing for specific styling or behavior (like sticky headers).

- **<tbody>**: It groups the main content of the table, which is the data. It makes the code more organized and easier to manage, especially when dealing with large tables.
- **<tfoot>**: It groups the footer rows, typically used for totals or summary rows at the bottom of the table.

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

Ans: <u>Difference Between Colspan And Rowspan in Tables:</u>

Colspan	Rowspan
Colspan is used to merge 2 or more cells horizontally.	Rowspan is used to merge 2 or more cells vertically.
The colspan is defined as the <th> element.	Applied within a <td> or <th> element.
It specifies the number of columns that the cells span across and shows them as a single cell.	the number of cells vertically and displays them as a single cell.
This is useful when you want a single cell to extend across multiple columns within a table.	This is useful when you want a single cell to extend vertically across multiple rows.
Example: <!DOCTYPE html> <html> <head> <title>Example of Table Colspan</title> </head> <body> <table border="1"> <tr>	Example: <!DOCTYPE html> <html> <head> <title>Example This cell spans 2 rows</title> </head> <body> <table border="1"> <tr>

<td colspan="2">This cell spans 2 columns</td> </tr> <tr> <td>Column 1</td> <td> Column 2</td> </tr> </table> </body> </html>	<td rowspan="2">This cell spans 2 Rows </td> <td>Row 2</td> </tr> </table> </body> </html>

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

Ans: Tables should be used sparingly for layout purposes because they were originally designed for displaying structured data, not for creating page layouts. When used for layout, they introduce several challenges:

- **Semantic issues:**
  - Tables are meant for tabular data, so using them for layout purposes can confuse browsers, assistive technologies, and developers. Screen readers, for example, expect tables to contain data, not layout information. Misusing tables can lead to accessibility problems.
- **Accessibility:**
  - Screen readers and assistive technologies expect tables to contain data, and they rely on specific markup to interpret the information correctly. Using tables for layout can confuse these tools and make it harder for users with disabilities to navigate the page.
- **Responsiveness:**
  - Tables are not well-suited for responsive design. When used for layout, tables can make it difficult to create flexible, mobile-friendly designs that adapt to different screen sizes.
- **Lack of flexibility:**

- Tables are rigid compared to modern layout techniques like CSS Grid and Flexbox. They don't offer the flexibility needed to build modern, dynamic, and responsive designs without significant workarounds.

❖ Better Alternatives:

- **CSS Grid:** CSS Grid is a powerful layout system that allows for flexible, two-dimensional layouts. It enables you to design complex page structures without using tables. It also allows for responsive design and is easier to maintain.
- **CSS Flexbox:** Flexbox is another layout model in CSS that provides a more flexible and efficient way to align and distribute space within a container. It's great for building responsive layouts and is easier to work with than tables.
- **CSS Positioning:** This method allows you to control the exact position of elements on the page using properties like absolute, relative, fixed, etc., although it is less commonly used for general layout nowadays compared to Grid and Flexbox.