Theory Assignment

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

HTML, which stands for **HyperText Markup Language**, is the standard markup language used to create web pages. It provides the fundamental structure of a webpage, defining elements like headings, paragraphs, images, and links.

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

All HTML documents must include the following three tags:

- **`<!DOCTYPE html>`:** This declaration isn't a tag in the traditional sense, but it is a required instruction for the web browser. It tells the browser which version of HTML the page is written in, ensuring it renders the page correctly. For HTML5, the standard version, this is simply `<!DOCTYPE html>`.
- **`<html>`:** This is the root element that encloses all the content on the HTML page. Every other tag, except for `<!DOCTYPE html>`, is nested inside of this tag. It serves as the container for the entire document.
- **`<head>`:** This section contains metadata about the HTML document. Information placed here isn't visible on the webpage itself but is crucial for the browser and search engines. Common elements within `<head>` include the page title (`<title>`), links to stylesheets (`<link>`), and scripts (`<script>`), as well as character set information (`<meta charset="UTF-8">`).
- **`<body>`:** This is where all the visible content of the web page goes. Everything you see when you visit a website—text, images, videos, and links—is placed within the `<body>` tag. It's the main container for the user-facing content.

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

Block-Level Elements

A **block-level element** always starts on a new line and takes up the full width available to it. It creates a "block" on the page, pushing any following content to the next line. You can think of them as the fundamental building blocks for the layout of a webpage. They are commonly used for structuring major parts of a document.

Examples of block-level elements:

- **`<div>`**: A generic container for flow content.
- **`<p>`**: Defines a paragraph of text.
- **`<h1>` to `<h6>`**: Defines headings of different sizes.

- **`<ul>` and `<ol>`**: Unordered and ordered lists.
- **`<footer>`**: Defines a footer for a document or section.

Inline Elements

An **inline element** does not start on a new line and only takes up as much width as necessary. They flow with the surrounding text, and you can place multiple inline elements on the same line. They are typically used to format smaller portions of text within a block-level element.

Examples of inline elements:

- **`<span>`**: A generic container for a small portion of text or a group of inline elements.
- **`<a>`**: Defines a hyperlink.
- **`<strong>` and `<em>`**: Used to bold and italicize text, respectively.
- **`<img>`**: Embeds an image.
- **`<br>`**: Inserts a single line break.

Key Differences Summarized

Feature	Block-Level Elements	Inline Elements
Line Break	Always starts on a new line.	Does not start on a new line.
Width	Occupies the full available width.	Occupies only the necessary width.
Placement	Can contain other block-level and inline elements.	Can only contain data and other inline elements.
Layout	Used for major structural layout.	Used for styling and formatting text within a block.

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

Semantic HTML is the practice of using HTML tags to define the **meaning** and **structure** of a webpage, rather than just its appearance. Instead of using generic tags like `<div>` or `<span>` for every part of a page, semantic tags—like `<header>`, `<article>`, or `<footer>`—provide context that both developers and machines can understand. It's about choosing the right element for the right job, ensuring the code is not only rendered correctly but is also logically organized.

Importance for Accessibility and SEO

Using semantic HTML is crucial for creating a more inclusive and discoverable web.

- **Accessibility**: Assistive technologies like **screen readers** rely on the semantic structure of a webpage to interpret its content for users with visual impairments. For example, a screen reader can identify an `<article>` tag and announce it as a self-contained story or post, allowing a user to jump directly to it. Without semantic tags, the entire page would be read as a jumbled block of text, making navigation nearly impossible. Semantic tags provide a clear hierarchy and a "roadmap" for these devices, significantly improving the user experience.
- **SEO (Search Engine Optimization)**: Search engine crawlers, like Googlebot, use semantic tags to better understand the content and hierarchy of a webpage. Tags like `<h1>` to `<h6>` indicate the importance of different headings, helping search engines determine the main topic of a page and its sub-topics. This contextual information helps them index the page more accurately and rank it higher for relevant search queries. Essentially, semantic HTML provides search engines with extra clues about your content, which can lead to better visibility.

Examples of Semantic Elements

Here are some common semantic HTML tags and their purposes:

- **`<header>`**: Defines a header for a document or section. It typically contains introductory content, a logo, or a navigation menu.
- **`<nav>`**: Represents a section of a page that contains navigation links.
- **`<main>`**: Specifies the main content of the document. A page should only have one `<main>` element.
- **`<article>`**: Encloses a self-contained piece of content that could be independently distributed or reused, such as a blog post, a news story, or a forum comment.

- **`<section>`**: Groups related content together thematically, usually with a heading.
- **`<aside>`**: Represents content that is tangentially related to the content around it, often used for sidebars.
- **`<footer>`**: Defines a footer for a document or a section. It often contains information like authorship, copyright details, or contact information.

Lab Assignment

Task:

Create a simple HTML webpage that includes:

A header (<header>), footer (<footer>), main section (<main>), and aside section (<aside>).

A paragraph with some basic text.

A list (both ordered and unordered).

A link that opens in a new tab.

```html
<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>Simple Webpage</title>

</head>

<body>



   <header>

       <h1>Website</h1>

       <p>This is the header section. It's great for site titles and navigation.</p>

   </header>
```

```html
<main>

    <h3>Lists</h3>
    <h4>Categories</h4>
    <ul>
        <li>Cars</li>
        <li>Models</li>
        <li>Prices</li>
    </ul>

    <h4>An Ordered List</h4>
    <ol>
        <li>First step</li>
        <li>Second step</li>
        <li>Third step</li>
    </ol>
</main>

<aside>
    <h3>Content</h3>
</aside>

<footer>
    <p>
        Services & Support

    </p>

        Contact Information:
           [Your Address]

           [Phone Number]

           [Email Address]


       <p>    Legal:

           Privacy Policy

           Terms of Use

       </p>

   </footer>


</body>

</html>
```

HTML Forms
Theory Assignment

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 data from users**. They act as a container for various form controls, allowing users to input information, make selections, and submit that data to a server for processing. This is how websites handle everything from user registrations and login screens to search queries, contact forms, and online order submissions.

Key HTML Form Elements

`input`

The `<input>` element is the most versatile form control. Its purpose changes based on its **`type` attribute**. It's a single-line element used for a wide variety of data types, including:

- **Text:** `type="text"` for names, addresses, etc.
- **Password:** `type="password"` to hide characters as they're typed.
- **Email:** `type="email"` for email addresses, with built-in validation.
- **Checkbox:** `type="checkbox"` for selecting one or more options from a list.
- **Radio:** `type="radio"` for selecting only one option from a group.
- **File:** `type="file"` for uploading files.
- **Number:** `type="number"` for numerical input.

`textarea`

The `<textarea>` element is used for **multi-line text input**. Unlike the `input` element, it's designed for longer blocks of text, such as comments, messages, or detailed feedback. It's often used for "Comments" or "Description" fields.

`select`

The `<select>` element is used to **create a drop-down list** of options. Users can click on the list to reveal a set of choices,and they can select only one (unless the `multiple` attribute is specified). The individual options within the drop-down are defined using the `<option>` element. It's useful for providing a predefined list of choices, like a list of countries or car models.

`button`

The `<button>` element is used to **create a clickable button**. The most common use is to submit the form data to the server, which is achieved by setting its `type` attribute to **`submit`**. Buttons can also be used to reset a form with `type="reset"` or simply as a generic

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

GET Method

The **GET** method appends form data to the URL as a query string (e.g., `www.example.com/search?q=cars`).

- **Data Visibility:** The data is visible in the URL, making it unsuitable for sending sensitive information like passwords.
- **Bookmarking:** The URL with the form data can be bookmarked and shared.

- **Limitations:** It has a limit on the amount of data that can be sent, typically around 2048 characters, because of browser and server limitations on URL length.
- **Use Case:** Use GET for retrieving data from the server, such as for search queries or filters. It's considered an **idempotent** method, meaning that making the same request multiple times will have the same effect as making it once.

POST Method

The **POST** method sends form data within the body of the HTTP request.

- **Data Visibility:** The data is not visible in the URL. This makes it a secure choice for transmitting sensitive information.
- **Bookmarking:** The request cannot be bookmarked.
- **Limitations:** There are no practical limits on the amount of data that can be sent, as the data is sent in the request body, not the URL.
- **Use Case:** Use POST for sending data to the server that changes the state of the server, such as creating a new user, submitting a blog post, or processing a payment. It is **not idempotent** because sending the same request multiple times could create multiple resources (e.g., multiple user accounts).

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 is a crucial component for associating descriptive text with a form control, such as an `<input>`, `<select>`, or `<textarea>`. Its primary purpose is to provide a clear and explicit link between the text label and the input field it describes.

Purpose of the `<label>` Element

The main purposes of the `<label>` element are:

- **Clarity for Sighted Users:** It provides a visible caption or title for the input field, making it immediately clear what information the user is expected to enter.
- **Enhanced Usability:** A major benefit is that a user can click on the label text itself to focus the associated input field. This is particularly helpful for small input fields like checkboxes and radio buttons, as it expands the clickable area, making it easier for users—especially on touch devices—to interact with the form.

How it Improves Accessibility

The `<label>` element is fundamental to creating accessible forms for everyone, especially for users with disabilities. Here's how it improves accessibility:

- **Screen Reader Association:** When a `<label>` is properly associated with an input, screen readers can announce the label's text when the user's focus is on the input field. Without a label, the screen reader would only announce the input type (e.g., "text field" or "checkbox"), leaving the user with no context about what to enter.
- **Keyboard Navigation:** For users who navigate a website using a keyboard, the `<label>` element ensures that the input field can be focused by pressing the tab key. This makes the form navigable and usable for those who cannot use a mouse.
- **Assistive Technology Support:** The programmatic connection created by the `<label>` element is the foundation for various assistive technologies to understand the purpose of form controls. This allows users with cognitive or motor impairments to interact with the form effectively.

Lab Assignment

Task:
Create a contact form with the following fields:

- ○ Full name (text input)
- ○ Email (email input)
- ○ Phone number (tel input)
- ○ Subject (dropdown menu)
- ○ Message (textarea)
- ○ Submit button

Additional Requirements:

Use appropriate form validation using required, minlength, maxlength, and pattern.

Link form labels with their corresponding inputs using the for attribute.

```
<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>Form</title>

</head>
```

```html
<body>



   <h2>Contact form</h2>

   <table>



   <form action="#" method="post">

       <label for="fullName">Full Name:</label>

       <input type="text" id="fullName" name="fullName" required minlength="3"
maxlength="100">


       <br><br>


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

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


       <br><br>


       <label for="Phone No">Phone No:</label>

       <input type="Phone No" id="Phone No" name="Phone No" required
pattern="[0-9]{10}">


       <br><br>


       <label for="Subject">Subject:</label>

       <select id="subject" name="subject">

           <option value="" disabled selected>Select a subject</option>

           <option value="general">Inquery</option>
```

```
            <option value="support">Technical Support</option>

            <option value="other">Other</option>

        </select>


        <br><br>




        <label for="message">Message:</label>

        <textarea id="message" name="message"></textarea>


        <br><br>


        <button type="submit">Send</button>


    </form>

    </table>


</body>

</html>
```

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>.

HTML Table Structure

An HTML table is a way to organize data into rows and columns, similar to a spreadsheet. The entire structure is built using a series of nested elements that define the table, its rows, headers, and individual data cells.

Here is a breakdown of the purpose of each element:

- **`<table>`**: This is the root element that defines the entire table. All other table-related elements (`<tr>`, `<th>`, `<td>`, etc.) must be placed inside the opening `<table>` and closing `</table>` tags.
- **`<tr>`**: This element stands for "table row." It is used to define a single row within the table. Each `<tr>` tag can contain one or more header cells (`<th>`) or data cells (`<td>`).
- **`<th>`**: This element stands for "table header." It is used to define a header cell in a table. The content inside a `<th>`tag is typically bold and centered by default in most browsers to indicate that it is a heading for a column or row. These cells are essential for accessibility, as they provide context for the data in the corresponding column or row.
- **`<td>`**: This element stands for "table data." It is used to define a standard data cell in a table. The content inside a `<td>` tag is the actual data you want to display, such as a name, number, or other information.
- **`<thead>`**: This element stands for "table head." It is a container for all the header rows (`<tr>` elements containing `<th>` cells) in a table. Its purpose is to semantically group the header content of the table, which is useful for screen readers and other assistive technologies. It also allows browsers to handle the body and footer of the table differently, for example, by allowing the body to scroll independently of the header.

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

`colspan`

The `colspan` attribute merges cells horizontally. It allows a single cell to span across multiple columns in a table. The value of `colspan` is a number that specifies how many columns the cell should occupy.

Example: Imagine a table showing a student's grades. If you want a single header for "Math Subjects" that covers both "Algebra" and "Geometry," you would use `colspan="2"`.

`rowspan`

The `rowspan` attribute merges cells vertically. It allows a single cell to span across multiple rows in a table. The value of `rowspan` is a number that specifies how many rows the cell should occupy.

Example: Using the same student grades table, if you want to show a student's name only once but have it span multiple rows for different subjects, you would use `rowspan`.

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

Using tables for page layout was a common practice in the early days of the web, but it is now considered outdated and is strongly discouraged. The main reason is that tables were designed for presenting tabular data, not for structuring an entire web page.

Why Tables Should Be Used Sparingly for Layout

- **Accessibility Issues**: Screen readers and other assistive technologies interpret tables as a collection of rows and cells. When a table is used for layout, it can disrupt the logical reading order, making it confusing for users with visual impairments to navigate the page.
- **Lack of Flexibility**: Table-based layouts are rigid and difficult to adapt to different screen sizes. This makes them unsuitable for responsive web design, where a website needs to look good on everything from a large desktop monitor to a small mobile phone.
- **Code Clutter**: Using tables for layout often results in a lot of extra, complex, and unnecessary HTML code. This makes the code harder to read, maintain, and debug.
- **Search Engine Optimization (SEO)**: Search engines may have difficulty parsing and ranking content buried within complex table structures, which can negatively impact a website's visibility.

A Better Alternative: CSS for Layout

The modern and widely accepted alternative for creating web page layouts is to use **Cascading Style Sheets (CSS)**, in combination with semantic HTML elements like `<div>`, `<header>`, `<main>`, `<footer>`, `<nav>`, and `<section>`.

Lab Assignment

Task:
Create a product catalog table that includes the following columns:

Product Name

Product Image (use placeholder image URLs)

Price

Description

Availability (in stock, out of stock)
Additional Requirements:

Use thead for the table header.

Add a border and some basic styling using inline CSS.

Use colspan or rowspan to merge cells where applicable.

```html
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Document</title>
</head>
<body>
   <table style="border: 2px solid #333; border-collapse: collapse; width: 100%;
text-align: center; font-family: Arial, sans-serif;">
   <thead style="background-color: #f2f2f2;">
       <tr>
           <th style="border: 1px solid #333; padding: 8px;">Product Name</th>

           <th style="border: 1px solid #333; padding: 8px;">Price</th>
           <th style="border: 1px solid #333; padding: 8px;">Description</th>
           <th style="border: 1px solid #333; padding: 8px;">Availability</th>
       </tr>
   </thead>
   <tbody>
       <tr>
           <td style="border: 1px solid #333; padding: 8px;">Wireless Headphones</td>

           <td style="border: 1px solid #333; padding: 8px;">$59.99</td>
           <td style="border: 1px solid #333; padding: 8px;">Noise-cancelling,
over-ear design with 20 hours battery life.</td>
```

```
            <td style="border: 1px solid #333; padding: 8px;">In Stock</td>

        </tr>

        <tr>

            <td style="border: 1px solid #333; padding: 8px;">Bluetooth Earbuds</td>

            <td style="border: 1px solid #333; padding: 8px;">$39.99</td>

            <td style="border: 1px solid #333; padding: 8px;">Compact, water-resistant
earbuds with charging case.</td>

            <td style="border: 1px solid #333; padding: 8px;">Out of Stock</td>

        </tr>

        <td style="border: 1px solid #333; padding: 8px;">Smartwatch</td>


            <td style="border: 1px solid #333; padding: 8px;">$159.99</td>

            <td style="border: 1px solid #333; padding: 8px;">20 hours battery
life.</td>

            <td style="border: 1px solid #333; padding: 8px;">In Stock</td>

        </tr>


        <tr>

        <td style="border: 1px solid #333; padding: 8px;">Gamingmouse</td>


            <td style="border: 1px solid #333; padding: 8px;">$59.99</td>

            <td style="border: 1px solid #333; padding: 8px;">good accurate speed</td>

            <td style="border: 1px solid #333; padding: 8px;">In Stock</td>

        </tr>

    </tbody>

</table>

</body>

</html>
```