**Module 2 – MERN stack – HTML**

* **HTML Basics**
* Define HTML. What is the purpose of HTML in web development ?

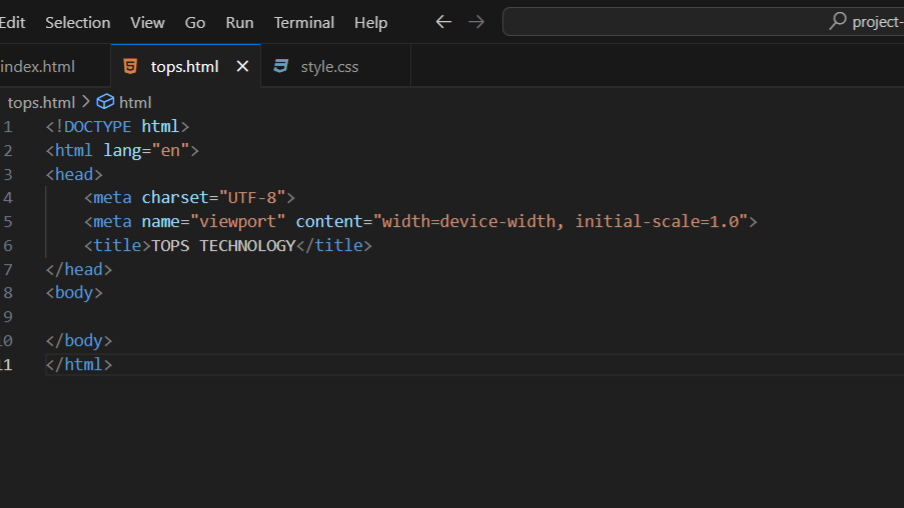
**HTML (HyperText Markup Language)** is the standard markup language used to create and structure content on the web. It organizes text, images, links, and media using tags and elements that browsers can interpret.

**Purpose of HTML in Web Development**

HTML serves as the backbone of web development by providing a standardized and structured approach to organizing and presenting content on the internet. It enables developers to create accessible and well-organized web pages, separates content from presentation, supports the integration of other web technologies, and contributes to search engine optimization.

**Key Features of HTML**

* **Semantic Structure**: HTML elements like <header>, <footer>, <article>, and <section> provide meaning to the content, enhancing accessibility and SEO.
* **Separation of Concerns**: HTML focuses on content structure, while CSS handles presentation and JavaScript manages behavior, promoting cleaner and more maintainable code.
* **Integration with Other Technologies**: HTML works seamlessly with CSS for styling and JavaScript for interactivity, enabling the creation of dynamic and responsive web applications.
* Explain the basic structure of an HTML document. Identify the mandatory tags and their purposes.



**Mandatory Tags & Their Roles**

1. **<!DOCTYPE html>**
   * Declares the document type for HTML5 and switches browsers to standards mode (avoiding quirks mode)
2. **<html lang="…">**
   * Root element of the page. The lang attribute specifies the document's language (e.g., en, hi-IN), which aids accessibility, translation, and search engines .
3. **<head>**
   * Container for metadata (not visible on the page). It includes character encoding, viewport instructions, page title, styles, scripts, and more
4. **<meta charset="UTF-8">**
   * Specifies the document’s character encoding. UTF‑8 supports international characters and emojis, essential for correct rendering
5. **<meta name="viewport" content="width=device-width, initial-scale=1.0">**
   * Ensures the layout adapts correctly across various devices (desktop, tablet, mobile) by controlling zoom and width behavior
6. **<title>**
   * Sets the page title displayed in browser tabs, history, bookmarks, and search engine results. This tag is **required** and important for SEO
7. **<body>**
   * Contains all visible content—text, images, links, sections, forms—which the browser renders on screen

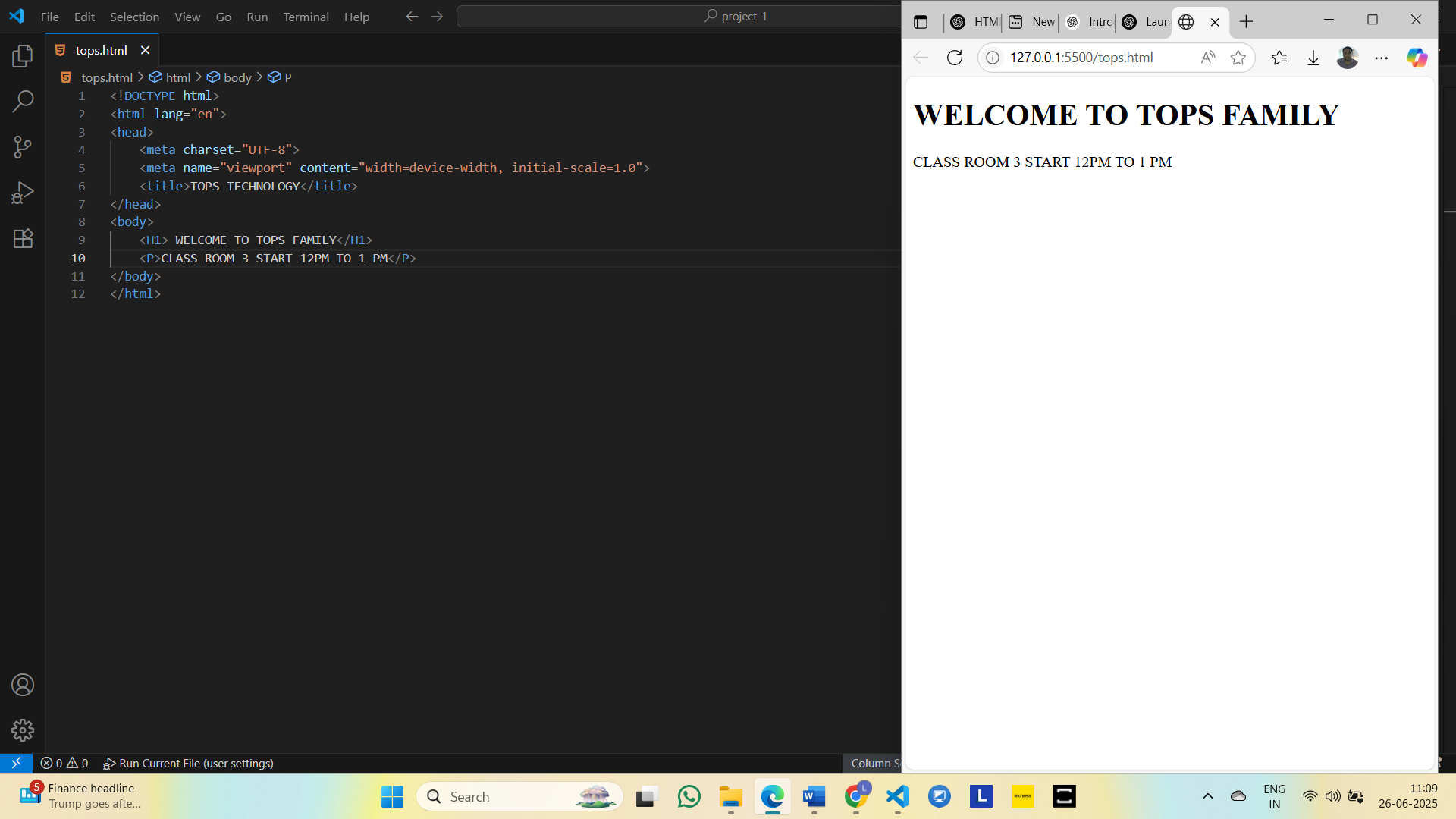
| **Tag** | **Purpose** |
| --- | --- |
| <!DOCTYPE html> | Activates standards mode |
| <html lang="…"> | Defines document root & language |
| <head> | Holds metadata and external references |
| <meta charset="UTF-8"> | Ensures proper text encoding (especially for Unicode) |
| <meta name="viewport" …> | Enables responsive design on mobile devices |
| <title> | Sets page title for UI and SEO |
| <body> | Contains visible content |

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

**Block-level Elements**

* **Behavior**:
  + Always start **on a new line**
  + **Stretch** to fill the **full width** of their parent container (unless constrained by CSS)
  + Accept **both block-level and inline** children
* **Common tags**: <div>, <p>, <h1>–<h6>, <section>, <article>, <ul>, <ol>, <blockquote>, <header>, <footer>

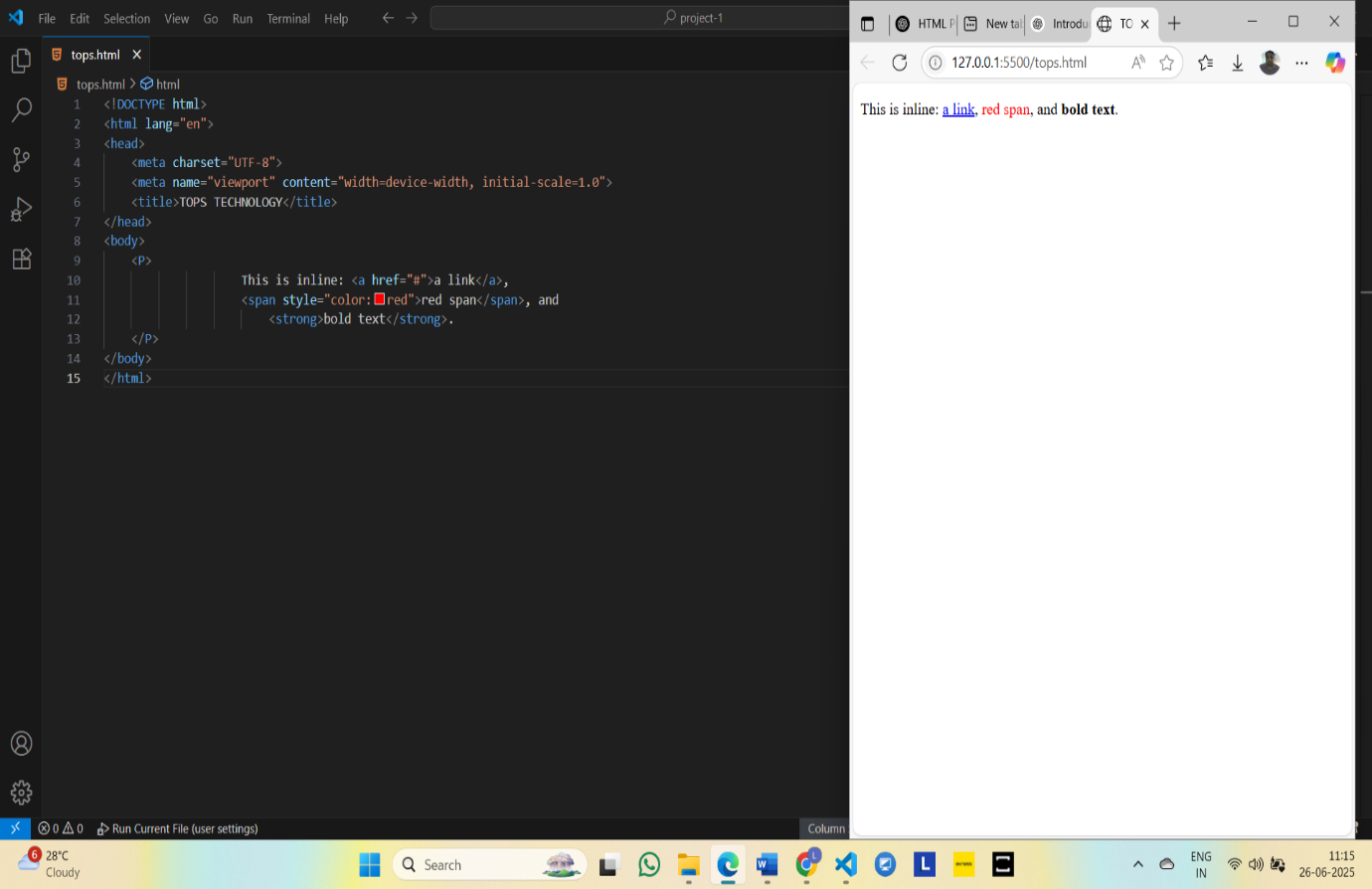
EXAMPLE:



**Inline Elements**

* **Behavior**:
  + Do **not** start on a new line; flow **within** text
  + Only occupy **as much width** as needed by their content
  + Can only contain **other inline elements** or text (not blocks)
* **Common tags**: <span>, <a>, <img>, <strong>, <em>, <code>, <label>, <input>

Example:



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

**Why Semantic HTML Matters**

1. **Accessibility**
   * Screen readers and assistive tools recognize tags like <nav>, <main>, and <header>, enabling users to skip sections or navigate intelligently
   * Interactive elements like <button> and <a> come with built-in keyboard behaviors that generic <div>s lack
2. **SEO**
   * Search engines use semantic tags to index and prioritize content. For instance, <article> signals a standalone piece, <main> highlights primary content, and headings (<h1>, <h2>) create hierarchy
   * Using descriptive anchor text inside <a> helps search engines understand link context
3. **Readability & Maintainability**
   * Code becomes easier to read, debug, and collaborate on when semantic tags clearly map to content function

**🧩 Common Semantic HTML Elements**

| **Element** | **Purpose & Benefit** |
| --- | --- |
| <header> | Page or section header; often contains logos and navs. Adds a banner landmark |
| <nav> | Wraps primary navigation links; gets navigation landmark role |
| <main> | The main content area; only one per page. Focus for SEO bots |
| <article> | Self-contained piece (e.g. blog post) ideal for indexing |
| <section> | Thematic grouping within content |
| <aside> | Sidebar or tangential content; gets complementary role |
| <footer> | Page or section footer; contains contact info, etc.; contentinfo |
| <h1>–<h6> | Headings that define content hierarchy for both users and search engines |
| <button>, <a>, <label>, <input> | Interactive and form-related elements with keyboard support |
| <figure> + <figcaption> | Images/illustrations with captions for context |
| <details> + <summary> | Popup sections like FAQs without JavaScript |
| <time>, <abbr>, <address>, <fieldset>, <dialog> | Other useful—but less common—semantic helpers |

**🧠 Real-World Illustration**

**Before (non-semantic)**:

html

<div id="header">…</div>

<div id="nav">…</div>

<div id="main\_content">…</div>

<div id="sidebar">…</div>

<div id="footer">…</div>

**After (semantic)**:

html

<header>…</header>

<nav>…</nav>

<main>

<article>…</article>

<aside>…</aside>

</main>

<footer>…</footer>

* **HTML Forms**
* What are HTML forms used for? Describe the purpose of the input, text area, select, and button elements

HTML forms are tools for collecting and sending user-entered data (like text, selections, or file uploads) to a server for processing—think login forms, search bars, feedback, or e-commerce checkouts.

**1.<input>**

* **Purpose**: Versatile single-line data entry.
* **Behavior**: Void element (no closing tag), with many types via the type attribute (e.g., text, email, password, number, checkbox, radio, file, submit).
* **Use cases**:
  + **Text input** for short inputs like names or emails (type="text" or type="email")
  + **Numeric** values (type="number")
  + **Password** fields (type="password")
  + **Checkbox/radio** for selecting options
  + **Submit/reset** buttons (type="submit"/type="reset")

2. **<textarea>**

* **Purpose**: Multi-line text input (e.g., comments, feedback).
* **Behavior**: Requires both opening and closing tags, supports rows and cols attributes or responsive CSS styling; allows line breaks.

3. **<select> (with <option>)**

* **Purpose**: Dropdown list for choosing from several predefined options.
* **Behavior**: Contains <option> elements, supports single or multiple selections via multiple, and can specify visible size via size.

4. **<button>**

* **Purpose**: Clickable button with flexible content (text, icons).
* **Behavior**: Requires type attribute (submit, reset, or button).
  + type="submit" sends form data
  + type="reset" clears inputs
* Explain the difference between the GET and POST methods in form submission. When should each be used?

**GET Method**

* **Data in URL**: Form fields are URL-encoded (e.g., ?name=Alice&age=30) and appended to the action URL
* **Use cases**: Ideal for retrieving or querying data—like search forms—where safe, cacheable, bookmarkable requests are needed
* **Characteristics**:
  + **Length-limited**: Generally up to ~2048 characters
  + **Cached & Bookmarkable**: Perfect for shareable, repeatable queries
  + **Exposed Data**: Parameters visible in browser history, logs, and address bar—so unsuitable for sensitive info
  + **Safe/Idempotent**: Should not change server state; calling it repeatedly has no side effects

**Example usage**

html

<form method="get" action="/search">

<input name="q" placeholder="Search…">

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

</form>

**🔐 POST Method**

* **Data in Body**: Parameters are sent in the HTTP request body (application/x-www-form-urlencoded or multipart/form-data)
* **Use cases**: Designed for sending sensitive info, uploading files, or performing actions with side effects (like creating or updating data)
* **Characteristics**:
  + **No Meaningful Size Limit**: Not constrained by URL length .
  + **Not Cached or Bookmarkable**: Safer for confidential or transactional data
  + **Data Hidden in URL**: Not visible in address bar or logs, reducing accidental leaks
  + **Not Safe and Not Idempotent**: May change server state and repeated submissions might create duplicates

**Example usage** (login form):

html

<form method="post" action="/login">

<input name="username">

<input type="password" name="password">

<button type="submit">Log in</button>

</form>

|  |  |  |
| --- | --- | --- |
| **Feature** | **GET Method** | **POST Method** |
| **Data visibility** | Data is visible in the URL | Data is hidden from the URL |
| **Security** | Less secure (not for sensitive data) | More secure (better for passwords, personal data) |
| **Data limit** | Limited (usually up to 2048 characters) | No practical limit (can send large data) |
| **Speed** | Faster (used for quick requests) | Slightly slower (due to extra data processing) |
| **Bookmarkable URL** | Yes – URL can be bookmarked with data | No – data is not stored in URL |
| **Use case** | Search forms, filters, non-sensitive data | Login forms, registration, secure data submission |
| **Caching** | Can be cached | Cannot be cached |
| **Idempotent (safe)?** | Yes (does not change data) | No (used to create or update data) |
| **Data sent via** | URL query string (e.g., ?name=John) | Message body of HTTP request |

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

The <label> element is used to define a caption for form inputs like textboxes, checkboxes, radio buttons, etc

|  |  |
| --- | --- |
| **Feature** | **Explanation** |
| **Improves usability** | When users click on the label, the related input field gets focused (easier to use) |
| **Better accessibility** | Screen readers read the label with the input, helping visually impaired users |
| **Clear identification** | Labels tell users what to enter in each field, avoiding confusion |

* **EX:**
* <label for="email">Email:</label> <input type="email" id="email" name="email">

**Output:**



* **HTML TABLE**
* Explain the structure of an HTML table and the purpose of each of the following elements: <table> , <tr> , <td>, <td>, <thead>.
* HTML tables allow web developers to arrange data into rows and columns.
* Each table cell is defined by a <td> and a </td> tag.
* Each table row starts with a <tr> and ends with a </tr> tag.
* Sometimes you want your cells to be table header cells. In those cases use the <th> tag instead of the <td> tag
* The <thead> tag is used to group header content in an HTML table

A screenshot of a computer

AI-generated content may be incorrect.

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

**COLSPAN** makes a cell span horizontally across multiple columns.

**ROWSPAN** makes a cell span vertically across multiple rows

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Purpose** | **Example Use Case** |
| rowspan | Merges **rows** vertically | To make a cell span across multiple **rows** |
| colspan | Merges **columns** horizontally | To make a cell span across multiple **columns** |

Example:

A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

Output:

A screenshot of a computer

AI-generated content may be incorrect.

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

Using HTML **tables for layout** (like placing elements side by side or aligning content) is discouraged for several reasons:

* Tables are meant to show **data**, like in rows and columns

(like marksheets, schedules).

* Using them to design page layout (like placing images or buttons) makes the code **messy and hard to change**.
* It becomes **difficult for blind people using screen readers** to understand the page.
* Tables **don’t work well on mobile** — the layout may break or look bad. The website may load **slower** if too many tables are used.

Better alternative:

Use **CSS** for layout. It's a better and modern way to design pages.

**Some CSS tools:**

* **Flexbox** – Easy for placing things side by side.
* **Grid** – Good for creating complex layouts.
* **Media queries** – Helps the layout change based on screen size

(mobile, tablet, desktop).

| **Issue** | **Table Layout** | **CSS Layout (Flexbox/Grid)** |
| --- | --- | --- |
| **Accessibility** | Confuses screen readers | Semantic, clear structure |
| **Responsiveness** | Rigid, breaks on mobile | Flexible, adapts to screen sizes |
| **Maintainability** | Nested HTML, hard to update | Clean markup + reusable stylesheets |
| **SEO-Friendly** | May bury important content | Supports source order independently |
| **Code Complexity** | Verbose and tangled | Leaner, modular CSS-controlled code |