HTML & CSS Course Syllabus (18 Lectures)

✅ Structure:

- **HTML** – 6 Lectures
- **CSS** – 6 Lectures
- **Flexbox** – 3 Lectures
- **Media Queries** – 3 Lectures

Each lecture builds on the last, includes practice, and ends with either a **mini project** or **exercise**.

🔶 HTML (Lectures 1–6)

Lecture 1: Introduction to HTML & First Page

- What is HTML?
- How websites work (Browser, Server, Hosting)
- Tools setup: VS Code, Live Server
- Basic HTML boilerplate (`<!DOCTYPE>`, `<html>`, `<head>`, `<body>`)
- Your First Web Page

Lecture 2: Headings, Paragraphs, Text Formatting

- Headings (`<h1>` to `<h6>`)
- Paragraphs (`<p>`)
- Line breaks, horizontal lines (`<br>`, `<hr>`)

- Formatting tags: `<b>`, `<i>`, `<u>`, `<strong>`, `<em>`, `<mark>`, `<sup>`, `<sub>`

Lecture 3: Lists, Links, and Images

- Unordered, Ordered, Nested Lists
- Anchor tags: internal & external links
- Image tag: `src`, `alt`, width/height
- Open links in new tab (`target="_blank"`)

Lecture 4: Tables in HTML

- Table structure: `<table>`, `<thead>`, `<tbody>`, `<tfoot>`
- Rows & columns: `<tr>`, `<td>`, `<th>`
- `colspan`, `rowspan`
- Table border styling (basic)

Lecture 5: Forms & Inputs

- `<form>`, `<input>`, `<label>`, `<textarea>`, `<button>`
- Input types: text, email, password, checkbox, radio
- Grouping inputs with `<fieldset>`, `<legend>`
- Submit button and action attribute

Lecture 6: Div, Span & Semantic Tags

- `<div>` and `<span>` for grouping

- Semantic tags: `<header>`, `<footer>`, `<nav>`, `<main>`, `<section>`, `<article>`, `<aside>`
- Mini Project: **Simple Resume Page** (using only HTML)

🟩 CSS (Lectures 7–12)

Lecture 7: Intro to CSS + Applying Styles

- What is CSS?
- Inline, Internal, and External CSS
- CSS Syntax
- Selectors: element, class, id

Lecture 8: Colors, Text, and Fonts

- Color names, hex, rgba
- Font-family, font-size, text-align, font-weight
- Google Fonts
- `text-decoration`, `line-height`, `letter-spacing`

Lecture 9: Box Model Deep Dive

- `margin`, `padding`, `border`, `width`, `height`
- `box-sizing: border-box`
- Visualizing with DevTools
- Mini challenge: Styled box layout

Lecture 10: Display & Position

- `display`: block, inline, inline-block, none
- `visibility: hidden`
- `position`: static, relative, absolute, fixed
- `z-index`, stacking order

Lecture 11: Backgrounds, Borders, Buttons

- Background colors, images, gradients
- Borders: width, color, style, radius
- Styling buttons with hover effects

Lecture 12: CSS Mini Project

- Build a **Landing Page (non-responsive)**
- Includes sections: header, about, services, contact

🧲 Flexbox (Lectures 13–15)

Lecture 13: Flexbox Fundamentals

- What is Flexbox?
- `display: flex`, `flex-direction`
- `justify-content`: start, center, end, space-around, space-between
- `align-items`: center, stretch, baseline
- Practice: Centering a card layout

Lecture 14: Flexbox Deep Dive

- `flex-wrap`, `gap`
- `align-self`, `align-content`
- `flex-grow`, `flex-shrink`, `flex-basis`
- Mini challenge: 3-column pricing table

Lecture 15: Flexbox Project

- Build a **Responsive Navigation Bar**
- Add logo + nav links
- Use `space-between` and media breakpoints

📱 Media Queries (Lectures 16–18)

Lecture 16: Introduction to Responsive Design

- Why responsiveness matters
- Viewport meta tag
- `@media` syntax
- Max-width & min-width concepts

Lecture 17: Responsive Layouts with Media Queries

- Create breakpoints for mobile, tablet, desktop
- Hide/show elements at different sizes

- Make the previous Flexbox navbar fully responsive

Lecture 18: Final Responsive Project

- Create a **Fully Responsive Landing Page**
    - Header, Hero, About, Features, Contact
    - Responsive using Flexbox + Media Queries
- Wrap-up + Q&A

🌟 Lecture 1: Introduction to HTML & First Page

📌 Topics Covered

1. What is HTML?
2. How websites work (Browser, Server, Hosting)
3. Tools setup: VS Code, Live Server
4. Basic HTML boilerplate
5. Your First Web Page

🧠 1. What is HTML?

HTML stands for **HyperText Markup Language**.
It is used to **structure** web content — like headings, paragraphs, links, images, and more.

HTML is not a programming language. It's a **markup language**, which means it tells the browser **how to display content**, not how to perform logic.

📌 Example:

```
<h1>Hello World!</h1>
<p>This is a paragraph.</p>
```

🌐 2. How Websites Work (Browser, Server, Hosting)

🔁 Step-by-Step Process:

1. **Browser** (like Chrome) sends a request to a **server** (a computer where your site files are stored).
2. The **server** finds your HTML file and sends it back.
3. The **browser** reads the HTML and shows it visually on the screen.

🧱 Roles:

- **HTML** structures the content.
- **CSS** styles it.
- **JavaScript** adds interactivity.
- **Hosting** makes your website available online (like GitHub Pages, Netlify, Hostinger).

🛠️ 3. Tools Setup: VS Code + Live Server

🔧 VS Code Setup:

- Download: https://code.visualstudio.com
- Install extensions:
    - 📡 **Live Server** by Ritwick Dey

🔥 Live Server Setup:

1. Open your HTML file in VS Code.
2. Right-click → "Open with Live Server"
3. It auto-refreshes your page whenever you save!

🔤 4. Basic HTML Boilerplate

This is the default structure every HTML page must follow:

```
<!DOCTYPE html>        <!-- Declares HTML5 -->
<html>                 <!-- Root of the HTML document -->
  <head>               <!-- Meta info, title, links -->
    <title>My Website</title>
  </head>
  <body>               <!-- Visible content -->
    <h1>Hello, world!</h1>
```

```
    <p>This is my first web page.</p>
  </body>
</html>
```

💻 5. Your First Web Page

Let's create a real example now.

🔰 Code Example: `index.html`

```
<!DOCTYPE html>
<html>
  <head>
    <title>Welcome Page</title>
  </head>
  <body>
    <h1>Welcome to My Website!</h1>
    <p>My name is Swaraj. I'm learning HTML today!</p>
  </body>
</html>
```

📍 Save this as `index.html`, right-click → "Open with Live Server".

🌟 Lecture 2: Headings, Paragraphs & Text Formatting

📌 Topics Covered

1. Headings (`<h1>` to `<h6>`)
2. Paragraphs (`<p>`)
3. Line Breaks (`<br>`) and Horizontal Lines (`<hr>`)
4. Text Formatting Tags:
    - Bold (`<b>` / `<strong>`)
    - Italic (`<i>` / `<em>`)
    - Underline (`<u>`)
    - Strikethrough (`<s>` / `<del>`)
    - Superscript / Subscript (`<sup>` / `<sub>`)
5. Nesting tags properly

🧠 1. Headings (<h1> to <h6>)

HTML has **6 levels of headings**:

- `<h1>` is the largest and most important
- `<h6>` is the smallest

📌 Example:

```
<h1>This is H1</h1>
<h2>This is H2</h2>
<h3>This is H3</h3>
```

```
<h4>This is H4</h4>
<h5>This is H5</h5>
<h6>This is H6</h6>
```

🧠 2. Paragraphs (<p>)

Use <p> to define paragraphs of text.

📌 Example:

```
<p>This is my first paragraph.</p>
<p>This is another paragraph.</p>
```

🧠 3. Line Breaks (
) & Horizontal Line (<hr>)

-
 → Breaks the line (inline)
- <hr> → Adds a horizontal rule (divider)

📌 Example:

```
<p>First Line<br>Second Line</p>
<hr>
<p>Another paragraph below the line.</p>
```

🧠 4. Text Formatting Tags

Tag	Use	Code Example	Color
	Bold text	Important	🟥
	Bold + Important meaning	Warning!	🟥
	Italic text	This is italic	🟪

`<em>`	Emphasized text (italic + meaning)	`<em>Pay attention</em>`	🟪
`<u>`	Underlined text	`<u>Underlined</u>`	🟨
`<s>` / `<del>`	Strikethrough (old/deleted)	`<s>Old Price</s>` or `<del>Deleted</del>`	🟫
`<sup>`	Superscript (above line)	`X<sup>2</sup>`	🟩
`<sub>`	Subscript (below line)	`H<sub>2</sub>O`	🟩

✅ Why Use `<strong>` Instead of `<b>`?

Reason	Description (Short)
1. Accessibility	Screen readers give `<strong>` added emphasis, helping visually impaired users.
2. Semantic Meaning	`<strong>` shows the text is important; `<b>` is just visual with no meaning.
3. Developer Clarity	Easier for developers to understand what matters in the code.
4. SEO Benefit	Search engines may treat `<strong>` text as more relevant.
5. CSS Styling	Easier to style important content consistently with semantic tags.
6. Future-Proofing	Works better with AI tools, content extractors, and web standards.

💡 Summary:

- Use `<strong>` when the text is **important**.
- Use `<b>` only for **visual bolding** with **no special meaning**.

Why Use `<em>` Instead of `<i>`?

Reason	Description (Short)

1. Accessibility	Screen readers add vocal emphasis to `<em>`, making it more meaningful when read.
2. Semantic Meaning	`<em>` means the text is emphasized or stressed; `<i>` is just visual styling.
3. Developer Clarity	Makes it clear that the text should be noticed or stressed.
4. SEO Benefit	Search engines may recognize `<em>` as more relevant than plain `<i>`.
5. CSS Styling	Semantic tags like `<em>` are easier to target for consistent styling.
6. Future-Proofing	Plays better with assistive tech, tools, and modern standards.

💡 Summary:

- Use `<em>` when the text needs **emphasis or stress**.
- Use `<i>` only for **italic styling** (e.g. foreign words, names, titles) with **no special meaning**.

🧠 5. Nesting Tags Properly

Make sure tags open and close in the correct order:

✅ Correct:

```
<p><b>This is bold inside a paragraph.</b></p>
```

❌ Wrong:

```
<p><b>This is wrong.</p></b>
```

💻 Full Example

```
<!DOCTYPE html>
```

```
<html>
  <head>
    <title>Text Formatting</title>
  </head>
  <body>
    <h1>Welcome to My Page</h1>
    <p>This is a <b>bold</b> and <i>italic</i> sentence.</p>
    <p><u>Underlined text</u> and <s>strikethrough</s></p>
    <p>Mathematics: X<sup>2</sup> + Y<sub>2</sub></p>
    <hr>
    <p>New paragraph after horizontal rule.<br>Second line using
br.</p>
  </body>
</html>
```

💡 Mini Challenge

Create a page that includes:

- One `<h1>` heading for your name.
- A paragraph introducing yourself.
- Make one word **bold**, another *italic*, and underline your hobby.
- Use a `<hr>` to separate sections.
- Use `<sup>` or `<sub>` for a math or chemical expression.

📚 Homework

✅ Task 1:

Make a file called `text-practice.html` with:

- Headings from `<h1>` to `<h3>`
- At least 2 paragraphs
- Use `<b>`, `<i>`, `<u>`, `<sup>`, and `<sub>` at least once

✅ Task 2:

Use `<br>` to split lines within a paragraph and `<hr>` between sections.

🌈 Lecture 3: Lists, Links, and Images

Topics We'll Cover:

- Unordered Lists (`<ul>`)
- Ordered Lists (`<ol>`)
- Nested Lists
- Anchor Tags: Internal & External Links
- Opening Links in New Tab (`target="_blank"`)
- Image Tag (`<img>`): `src`, `alt`, `width`, `height`

1️⃣ Lists in HTML

Unordered List (`<ul>`)

- Used for bulleted lists
- List items wrapped with `<li>`

```
<ul>
  <li>Apple</li>
  <li>Banana</li>
  <li>Cherry</li>
</ul>
```

Ordered List (`<ol>`)

- Used for numbered lists

```
<ol>
  <li>Wake up</li>
  <li>Brush Teeth</li>
  <li>Have Breakfast</li>
</ol>
```

Nested Lists

- Lists inside lists to create subcategories

```
<ul>
  <li>Fruits
    <ul>
      <li>Apple</li>
      <li>Banana</li>
    </ul>
  </li>
  <li>Vegetables
    <ul>
      <li>Carrot</li>
      <li>Spinach</li>
    </ul>
  </li>
</ul>
```

2️⃣ Anchor Tags (`<a>`) – Links

Basic Anchor Tag Syntax:

```
<a href="https://example.com">Visit Example</a>
```

- `href` attribute defines the URL or path
- Internal links navigate within your site, e.g. `<a href="about.html">About Us</a>`

Open Link in New Tab

Add `target="_blank"`:

```
<a href="https://google.com" target="_blank">Google (opens in new
tab)</a>
```

3️⃣ Image Tag (`<img>`)

Important attributes:

- `src`: image source (URL or file path)
- `alt`: alternative text (important for accessibility and SEO)
- `width` and `height`: control image size

```
<img src="cat.jpg" alt="Cute cat" width="300" height="200">
```

💻 Complete Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Lists, Links & Images</title>
  </head>
  <body>

    <h1>My Favorite Things</h1>

    <h2>Fruits</h2>
    <ul>
      <li>Apple</li>
      <li>Banana</li>
      <li>Cherry</li>
    </ul>

    <h2>Morning Routine</h2>
    <ol>
      <li>Wake up</li>
      <li>Brush Teeth</li>
      <li>Have Breakfast</li>
    </ol>

    <h2>Shopping List</h2>
    <ul>
      <li>Fruits
```

```
        <ul>
          <li>Orange</li>
          <li>Grapes</li>
        </ul>
      </li>
      <li>Vegetables
        <ul>
          <li>Potato</li>
          <li>Tomato</li>
        </ul>
      </li>
    </ul>

    <h2>Links</h2>
    <p><a href="https://youtube.com" target="_blank">YouTube</a></p>
    <p><a href="about.html">About Us (Internal Link)</a></p>

    <h2>Images</h2>
    <img src="flower.jpg" alt="Beautiful Flower" width="250">
    <img src="logo.png" alt="Company Logo" height="100">

  </body>
</html>
```

🏆 Mini Challenge

- Create an HTML page with:
    - An unordered list with 3 hobbies
    - An ordered list with your daily tasks
    - A nested list with categories (e.g., sports > football, basketball)
    - 2 external links (one opens in a new tab)
    - 2 images with alt texts and different sizes

🏠 Homework

1. Create a file named `lecture3.html` that contains:
    - A nested list of your favorite foods
    - Three links: one external (opens in new tab), two internal
    - Three images with various widths or heights
2. Add comments in your code to explain each section.

🌈 Lecture 4: Tables in HTML

Topics We'll Cover:

- Table structure: `<table>`, `<thead>`, `<tbody>`, `<tfoot>`
- Table rows and columns: `<tr>`, `<td>`, `<th>`
- Cell spanning: `colspan` and `rowspan`
- Basic table border styling

1️⃣ Table Structure

- `<table>`: Container for the entire table
- `<thead>`: Defines table header section
- `<tbody>`: Defines the main body of the table
- `<tfoot>`: Defines the footer section (optional)

```
<table>
  <thead>
    <tr>
      <th>Header 1</th>
      <th>Header 2</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Row 1, Cell 1</td>
      <td>Row 1, Cell 2</td>
    </tr>
    <tr>
      <td>Row 2, Cell 1</td>
      <td>Row 2, Cell 2</td>
    </tr>
  </tbody>
  <tfoot>
```

```
    <tr>
      <td>Footer 1</td>
      <td>Footer 2</td>
    </tr>
  </tfoot>
</table>
```

2️⃣ Rows & Columns

- `<tr>`: Defines a table row
- `<td>`: Defines a table cell (data cell)
- `<th>`: Defines a header cell (bold and centered by default)

3️⃣ Colspan and Rowspan

- **`colspan`**: Makes a cell span multiple columns
- **`rowspan`**: Makes a cell span multiple rows

Example:

```
<table border="1">
  <tr>
    <th>Product</th>
    <th colspan="2">Price</th>
  </tr>
  <tr>
    <td>Apple</td>
    <td>$1</td>
    <td>$1.20</td>
  </tr>
  <tr>
    <td rowspan="2">Orange</td>
    <td>$0.80</td>
    <td>$0.90</td>
  </tr>
```

```
  <tr>
    <td colspan="2">Special Offer</td>
  </tr>
</table>
```

4️⃣ Basic Table Border Styling

Add `border` attribute for simple borders (not recommended for production, use CSS for better styling).

```
<table border="1">
  ...
</table>
```

Better practice with CSS:

```
<style>
  table {
    border-collapse: collapse;
    width: 80%;
  }
  th, td {
    border: 2px solid #333;
    padding: 8px;
    text-align: center;
  }
  thead {
    background-color: #f2f2f2;
  }
</style>
```

💻 Complete Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Tables</title>
    <style>
```

```
    table {
      border-collapse: collapse;
      width: 70%;
      margin: 20px auto;
    }
    th, td {
      border: 2px solid #555;
      padding: 10px;
      text-align: center;
    }
    thead {
      background-color: #add8e6;
    }
    tfoot {
      background-color: #f9f9f9;
      font-style: italic;
    }
  </style>
</head>
<body>

  <h1>Product Price Table</h1>

  <table>
    <thead>
      <tr>
        <th>Product</th>
        <th colspan="2">Price</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>Apple</td>
        <td>$1.00</td>
        <td>$1.20</td>
      </tr>
      <tr>
        <td rowspan="2">Orange</td>
        <td>$0.80</td>
        <td>$0.90</td>
      </tr>
      <tr>
        <td colspan="2">Special Offer</td>
      </tr>
```

```
      </tbody>
      <tfoot>
        <tr>
          <td colspan="3">Prices subject to change</td>
        </tr>
      </tfoot>
    </table>

  </body>
</html>
```

🏆 Mini Challenge

- Create a table listing 3 of your favorite movies.
- Include columns for Title, Year, and Genre.
- Use a `<thead>` for headers, `<tbody>` for content, and `<tfoot>` for a footer note.
- Use `colspan` or `rowspan` in at least one cell creatively.
- Style the table borders using CSS (no `border` attribute).

🏠 Homework

1. Build an HTML file named `lecture4.html` with a table showing a weekly schedule (days as columns, time slots as rows).
2. Use `<thead>`, `<tbody>`, and `<tfoot>`.
3. Add at least one `colspan` and `rowspan` usage.
4. Style the table with CSS to have borders and alternate header background colors.

🗒️ Lecture 5: Forms & Inputs in HTML

📚 Topics We'll Cover:

- `<form>`, `<input>`, `<label>`, `<textarea>`, `<button>`
- Input types: `text`, `email`, `password`, `checkbox`, `radio`
- Grouping fields with `<fieldset>` and `<legend>`
- `action` attribute and submit behavior

1️⃣ What is a Form?

HTML forms are used to **collect user input**. A form can contain input fields, checkboxes, buttons, text areas, etc.

```
<form>
  <!-- form inputs go here -->
</form>
```

2️⃣ Form Elements

🔹 `<input>` – Most common form element

```
<input type="text" placeholder="Your name" />
```

🔹 `<label>` – Describes input field (improves accessibility)

```
<label for="username">Username:</label>
<input type="text" id="username" name="username" />
```

🔹 `<textarea>` – For multi-line input

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

```
<textarea id="message" name="message" rows="4" cols="40"></textarea>
```

🔹 **`<button>` – Used to submit or reset**

```
<button type="submit">Submit</button>
```

3️⃣ Common Input Types

```
<input type="text" placeholder="Text input" />
<input type="email" placeholder="Email input" />
<input type="password" placeholder="Password input" />
```

✅ Checkbox & Radio

```
<label>
  <input type="checkbox" name="subscribe" /> Subscribe to newsletter
</label>

<br />

<label>
  <input type="radio" name="gender" value="male" /> Male
</label>
<label>
  <input type="radio" name="gender" value="female" /> Female
</label>
```

4️⃣ Grouping Inputs with `<fieldset>` and `<legend>`

Use these to logically group form elements.

```
<fieldset>
  <legend>Personal Info</legend>

  <label>Name:</label>
  <input type="text" name="name" />
```

```
  <br />

  <label>Email:</label>
  <input type="email" name="email" />
</fieldset>
```

5️⃣ Submit Button & `action` Attribute

- The `action` attribute defines **where to send form data**.
- The `method` can be `GET` or `POST`.

```
<form action="/submit-form" method="POST">
  <input type="text" name="name" />
  <button type="submit">Submit</button>
</form>
```

💻 Complete Example

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Form Example</title>
  <style>
    form {
      width: 300px;
      margin: 30px auto;
      padding: 20px;
      border: 2px solid #444;
      border-radius: 10px;
      background-color: #f0f8ff;
    }
    label {
      display: block;
      margin-top: 10px;
      font-weight: bold;
    }
```

```
    input, textarea {
      width: 100%;
      padding: 8px;
      margin-top: 4px;
      margin-bottom: 10px;
      border-radius: 5px;
      border: 1px solid #aaa;
    }
    button {
      background-color: #007bff;
      color: white;
      padding: 8px 15px;
      border: none;
      border-radius: 5px;
      cursor: pointer;
    }
  </style>
</head>
<body>

<form action="/submit" method="POST">
  <fieldset>
    <legend>Contact Us</legend>

    <label for="name">Name:</label>
    <input type="text" id="name" name="name" required />

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

    <label for="message">Message:</label>
    <textarea id="message" name="message" rows="4"></textarea>

    <label>
      <input type="checkbox" name="subscribe" /> Subscribe to
newsletter
    </label>

    <label>Gender:</label>
    <label><input type="radio" name="gender" value="male" />
Male</label>
    <label><input type="radio" name="gender" value="female" />
Female</label>
```

```
    <br />
    <button type="submit">Send</button>
  </fieldset>
</form>

</body>
</html>
```

🧠 Mini Challenge

🔹 Create a **Feedback Form** with the following:

- Name, Email (required), and a Message box
- Gender selection (radio buttons)
- Subscribe checkbox
- A Submit button
- Use at least one `<fieldset>` with `<legend>`
- Style the form using internal CSS

🏠 Homework

📝 Create a file `lecture5.html` and build the following:

1. A **Registration Form** with:
    - Name, Email, Password
    - Gender (radio), Skills (checkbox)
    - Bio (textarea)
2. Group relevant fields with `<fieldset>`
3. Add basic styling with border, padding, and width
4. Submit button with `type="submit"`

🧱 Lecture 6: Div, Span & Semantic Tags

📚 Topics We'll Cover:

- `<div>` and `<span>` for grouping and layout
- Semantic HTML tags: `<header>`, `<footer>`, `<nav>`, `<main>`, `<section>`, `<article>`, `<aside>`
- Mini Project: Build a **Simple Resume Page** using only HTML

1️⃣ What is `<div>`?

- **Block-level container** used to group elements.
- No default style, often used with CSS.

```
<div>
  <h2>Welcome</h2>
  <p>This is a section inside a div.</p>
</div>
```

2️⃣ What is `<span>`?

- **Inline-level container** to group text or inline elements.
- Useful for styling part of a sentence.

```
<p>This is a <span style="color: red;">red word</span> in a
sentence.</p>
```

3️⃣ 🧠 When to Use `<div>` vs `<span>`

Use Case	Element
Grouping block elements (layout)	`<div>`
Styling a word/phrase inside text	`<span>`

4️⃣ Semantic HTML Tags

Semantic tags describe **meaning** of the content clearly to browsers and developers.

🔹 `<header>` – Top section of the page or article

```
<header>
  <h1>My Website</h1>
</header>
```

🔹 `<footer>` – Bottom section of the page or article

```
<footer>
  <p>&copy; 2025 My Portfolio</p>
</footer>
```

🔹 `<nav>` – Contains navigation links

```
<nav>
  <a href="#about">About</a>
  <a href="#projects">Projects</a>
</nav>
```

🔹 `<main>` – Primary content of the page

```
<main>
  <h2>Welcome to My Portfolio</h2>
</main>
```

🔹 **`<section>` – Groups related content**

```
<section>
  <h3>Skills</h3>
  <ul>
    <li>HTML</li>
    <li>CSS</li>
  </ul>
</section>
```

🔹 **`<article>` – Independent content like blogs, news**

```
<article>
  <h2>My Journey into Web Development</h2>
  <p>It all started when...</p>
</article>
```

🔹 **`<aside>` – Sidebar or extra info**

```
<aside>
  <p>This is a tip or note.</p>
</aside>
```

🧑‍💼 Mini Project: Simple Resume Page (Only HTML)

```
<!DOCTYPE html>
<html>
<head>
  <title>My Resume</title>
</head>
<body>

<header>
  <h1>Swaraj Jadhav</h1>
  <p>Full Stack Developer</p>
```

```
</header>

<nav>
  <a href="#about">About</a> |
  <a href="#skills">Skills</a> |
  <a href="#contact">Contact</a>
</nav>

<main>
  <section id="about">
    <h2>About Me</h2>
    <p>I am a passionate developer with experience in MERN stack.</p>
  </section>

  <section id="skills">
    <h2>Skills</h2>
    <ul>
      <li>HTML</li>
      <li>CSS</li>
      <li>JavaScript</li>
    </ul>
  </section>

  <section id="contact">
    <h2>Contact</h2>
    <p>Email: example@email.com</p>
  </section>
</main>

<footer>
  <p>&copy; 2025 Swaraj Jadhav</p>
</footer>

</body>
</html>
```

🧠 Mini Challenge

- 🔹 Create a **Web Developer Resume Page** using:
  - `<header>`, `<nav>`, `<main>`, `<section>`, `<footer>`

- Your name, profession, skills, and contact info
- Style using only HTML (no CSS yet)

🏠 Homework

📝 Create a file `resume.html` and:

1. Use **semantic tags** only (`div` only if necessary)
2. Add 3 sections: **About**, **Skills**, **Projects**
3. Include a **nav bar** at the top
4. Add a **footer** with copyright
5. BONUS: Use `<article>` and `<aside>` for extra content

🎨 Lecture 7: Intro to CSS + Applying Styles

📚 Topics We'll Cover:

- What is CSS?
- Inline, Internal, and External CSS
- CSS Syntax
- Selectors: element, class, id

1️⃣ What is CSS?

CSS (Cascading Style Sheets) is used to **style** and **layout** HTML content — like colors, fonts, spacing, alignment, and responsiveness.

🧠 It **separates content (HTML)** from **presentation (CSS)**.

2️⃣ Types of Applying CSS

🔸 Inline CSS

Applied directly inside HTML elements using the `style` attribute.

```
<p style="color: blue; font-size: 18px;">Hello Inline CSS</p>
```

✅ Quick changes
❌ Not reusable

🔸 Internal CSS

Written inside `<style>` tag in the HTML `<head>`.

```html
<!DOCTYPE html>
<html>
<head>
  <style>
    h1 {
      color: green;
    }
  </style>
</head>
<body>
  <h1>This is styled using Internal CSS</h1>
</body>
</html>
```

✅ Good for small projects
❌ Doesn't separate design from content fully

🔸 External CSS

Linked via a separate `.css` file.

style.css

```css
body {
  background-color: #f4f4f4;
}
```

index.html

```
<link rel="stylesheet" href="style.css">
```

✅ Best practice
✅ Reusable and clean

3️⃣ CSS Syntax

```
selector {
  property: value;
}
```

🔹 **Example**:

```
p {
  color: red;
  font-size: 16px;
}
```

📌 Multiple properties can be added using `;` between each.

4️⃣ CSS Selectors

🔹 Element Selector

Targets HTML tags.

```
h2 {
  color: purple;
}
```

🔹 Class Selector (`.`)

```
<p class="highlight">Important text</p>

.highlight {
  background-color: yellow;
}
```

🔹 ID Selector (`#`)

```
<h1 id="main-heading">Welcome</h1>

#main-heading {
  text-align: center;
}
```

✅ Use ID only **once per page**

🎯 Summary Table

Selector Type	Example	Targets
Element	`p`	All `<p>` tags
Class	`.card`	All elements with class "card"
ID	`#hero`	Unique element with id "hero"

✏️ Mini Challenge

🔸 Create a page with:

- An inline-styled heading
- Internal CSS for a paragraph
- An external CSS file that changes the background color of the body and sets font styles

---

🏠 Homework

✅ Create a folder `lecture7/`
✅ Inside it:

1. Create `index.html` with:
    - One `<h1>` styled inline
    - One `<p>` styled via internal CSS
    - One `<div>` with class `"box"` styled in an external `style.css`
2. In `style.css`:

- Set `.box` to have width `300px`, background `lightblue`, and padding `20px`

3. Link the CSS file properly

🎨 Lecture 8: Colors, Text, and Fonts

📚 Topics We'll Cover:

- CSS Colors (names, hex, rgba)
- Font styling: `font-family`, `font-size`, `text-align`, `font-weight`
- Google Fonts
- Text styling: `text-decoration`, `line-height`, `letter-spacing`

1️⃣ Colors in CSS

You can apply colors using:

- **Color Names** (like `red`, `blue`, `green`)
- **Hex Codes** (like `#ff0000`)
- **RGBA Values** (Red, Green, Blue, Alpha)

✅ Example:

```
/* Named color */
h1 {
  color: red;
}

/* Hexadecimal */
p {
  color: #3498db;
}
```

```css
/* RGBA with transparency */
.box {
  background-color: rgba(255, 0, 0, 0.3);
}
```

2️⃣ Font Styling

🔹 font-family

```css
body {
  font-family: Arial, sans-serif;
}
```

➡ You can provide multiple fonts as fallbacks.

🔹 font-size

```css
h2 {
  font-size: 24px;
}
```

✅ You can use `px`, `em`, `rem`, `%`.

🔹 `text-align`

```
p {
  text-align: center;
}
```

Options: `left`, `center`, `right`, `justify`

🔹 `font-weight`

```
strong {
  font-weight: bold; /* or 400, 700, etc. */
}
```

3️⃣ Using Google Fonts

Go to https://fonts.google.com

Step-by-step:

1. Pick a font → click “+”
2. Copy the `<link>` tag into your HTML `<head>`
3. Use the font in CSS

🔽 Example:

In HTML:

```
<link
href="https://fonts.googleapis.com/css2?family=Poppins&display=swap"
rel="stylesheet">
```

In CSS:

```
body {
  font-family: 'Poppins', sans-serif;
}
```

4️⃣ More Text Styling

🔹 text-decoration

```
a {
  text-decoration: none; /* remove underline */
}
```

🔹 line-height

```
p {
  line-height: 1.6;
}
```

Makes paragraphs more readable.

🔹 `letter-spacing`

```
h1 {
  letter-spacing: 2px;
}
```

Adjusts space between letters.

✏️ Mini Challenge

Create a styled heading and paragraph:

- Heading color: `#1abc9c`, text-align center, Google font (`Poppins`)
- Paragraph with: font-size 18px, line-height 1.6, color `rgba(0,0,0,0.7)`
- Link styled with no underline and blue color

🏠 Homework

✅ Create a folder `lecture8/`
✅ Inside it:

1. `index.html` with:
    - A heading using Google Font (`Roboto`)
    - Two paragraphs with different text styles
    - A link styled with no underline
2. `style.css`:
    - Use `rgba` background for one box
    - Set custom `letter-spacing` and `line-height` for text

📦 Lecture 9: Box Model Deep Dive

📚 Topics We'll Cover:

- `margin`, `padding`, `border`, `width`, `height`
- `box-sizing: border-box`
- Visualizing with Chrome DevTools
- ✅ Mini Challenge

1️⃣ What is the CSS Box Model?

Every HTML element is a box, made of:

- **Content** → the actual text or image
- **Padding** → space inside the element
- **Border** → the outer line
- **Margin** → space outside the element

2️⃣ Properties

🔹 Width & Height

```
.box {
  width: 300px;
  height: 150px;
}
```

🔹 Padding

```
.box {
  padding: 20px;
}
```

➡ Adds space **inside** the border

🔹 Margin

```
.box {
  margin: 30px;
}
```

➡ Adds space **outside** the element

🔹 Border

```
.box {
  border: 2px solid #2980b9;
}
```

➡ `border: width style color`

3️⃣ box-sizing: border-box

By default, width + padding + border = total width
Using `border-box`, padding & border are **included** in the width.

```
* {
  box-sizing: border-box;
}
```

4️⃣ Visualizing with Chrome DevTools

1. Right-click element → Inspect
2. Hover to see Box Model (margin, border, padding)
3. Use “Computed” tab to view actual dimensions

🧩 Mini Challenge

✅ Create a `box` with:

- Width: `300px`, Height: `200px`
- Padding: `20px`, Margin: `40px`
- Border: `3px dashed #e74c3c`
- `box-sizing: border-box`

🏠 Homework

Create 3 boxes (`div`):

- Different padding, margin, border
- Use `box-sizing: content-box` and `border-box`
- Visualize using DevTools

🔧 Lecture 10: Display & Position

📚 Topics We'll Cover:

- `display`: `block`, `inline`, `inline-block`, `none`
- `visibility: hidden`
- `position`: `static`, `relative`, `absolute`, `fixed`
- `z-index`, stacking elements

1️⃣ display Property

🔹 block

```
div {
  display: block;
}
```

Takes full width.

🔹 inline

```
span {
  display: inline;
}
```

Takes only the content width. Can't set height/margin-top.

🔹 inline-block

```
.button {
  display: inline-block;
  padding: 10px;
}
```

Behaves like inline, but allows width/height.

🔹 none

```
.box {
  display: none;
}
```

Hides the element completely.

2️⃣ visibility: hidden

```
.secret {
  visibility: hidden;
}
```

👀 Element is invisible, but space is still reserved.

3️⃣ position Property

- **static (default)**

```
.element {
  position: static;
}
```

- **relative**

```
.element {
  position: relative;
  top: 10px;
  left: 20px;
}
```

Moves relative to its normal position.

- **absolute**

```
.element {
  position: absolute;
  top: 0;
```

```
  left: 0;
}
```

📍 Positioned relative to the **nearest positioned parent**.

🔹 fixed

```
.nav {
  position: fixed;
  top: 0;
}
```

Stays fixed on the screen while scrolling.

4️⃣ z-index

```
.box1 {
  z-index: 1;
}

.box2 {
  z-index: 2;
}
```

Higher `z-index` appears **on top**.

🧩 Mini Challenge

✅ Create 3 boxes:

- One with `relative` positioning
- One with `absolute` inside a `relative` container
- One fixed at top-left corner
- Use `z-index` to layer them

🏠 Homework

Create:

- A `header` fixed at the top
- A `button` placed with `absolute` and `relative`
- Practice hiding/showing using `display` and `visibility`

🎨 Lecture 11: Backgrounds, Borders & Buttons

📚 Topics We'll Cover:

- Background colors, images, gradients
- Borders: width, color, style, radius
- Styling buttons with hover effects

1️⃣ Background Colors

```
.section {
  background-color: lightblue;
}
```

✔ You can use color names, hex codes, `rgb()`, or `rgba()`.

2️⃣ Background Images

```
.hero {
  background-image: url("banner.jpg");
  background-size: cover;
  background-position: center;
  background-repeat: no-repeat;
}
```

3️⃣ Background Gradients

```
.box {
  background: linear-gradient(to right, #ff7e5f, #feb47b);
}
```

➡ Direction: `to right`, `to bottom`, etc.
➡ You can use multiple color stops.

4️⃣ Borders

```
.card {
  border: 2px solid #333;
  border-radius: 10px;
}
```

Border Properties:

- `border-width`
- `border-color`
- `border-style`: `solid`, `dashed`, `dotted`
- `border-radius`: for rounded corners

5️⃣ Styling Buttons

```
button {
  background-color: #3498db;
  color: white;
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}
```

6️⃣ Button Hover Effect

```
button:hover {
  background-color: #2980b9;
}
```

🧠 Hover effects improve UX!

🏠 Homework

✅ Create a styled button with:

- Gradient background
- Rounded borders

- Hover effect
- Custom font and padding

🚀 Lecture 12: CSS Mini Project – Build a Landing Page

📚 Project Brief:

- Create a **non-responsive** landing page using only **HTML + CSS**
- Sections to include:
    - Header
    - About
    - Services
    - Contact

1️⃣ Layout Plan (Sections)

```
<header>My Brand</header>
<section id="about">...</section>
<section id="services">...</section>
<section id="contact">...</section>
```

2️⃣ Sample Structure

```
<header>
  <h1>Welcome to My Landing Page</h1>
  <nav>...</nav>
```

```
</header>

<section id="about">
  <h2>About Us</h2>
  <p>We provide amazing services...</p>
</section>

<section id="services">
  <h2>Our Services</h2>
  <ul>
    <li>Web Development</li>
    <li>Design</li>
  </ul>
</section>

<section id="contact">
  <h2>Contact Us</h2>
  <p>Email: info@example.com</p>
</section>
```

3️⃣ Apply CSS Styling

```
header {
```

```
  background-color: #2c3e50;
  color: white;
  padding: 30px;
  text-align: center;
}

section {
  padding: 40px;
  border-bottom: 1px solid #ccc;
}
```

4️⃣ Bonus Styling Ideas

- Add a Google Font
- Use gradients in backgrounds
- Add hover effects on links/buttons
- Use `border-radius` and shadows for modern UI

🧩 Challenge:

Make it creative!
💡 Try different color schemes and layouts.

📦 Lecture 13: Flexbox Fundamentals

📚 Topics We'll Cover:

- What is Flexbox?
- `display: flex`, `flex-direction`
- `justify-content`: start, center, end, space-around, space-between
- `align-items`: center, stretch, baseline
- Practice: Centering a card layout

🔍 What is Flexbox?

Flexbox is a one-dimensional layout system used to align items in rows or columns.

It helps with:

- Horizontal/vertical alignment
- Spacing and distribution
- Responsive design

✅ Enabling Flexbox

```
.container {
  display: flex;
}
```

This turns `.container` into a **flex container**.

🔄 `flex-direction`

Defines the direction of the main axis (default: row)

```
.container {
  flex-direction: row;        /* default */
  flex-direction: column;
  flex-direction: row-reverse;
  flex-direction: column-reverse;
}
```

🎯 `justify-content` – Horizontal Alignment (Main Axis)

```
.container {
  justify-content: flex-start;    /* default */
  justify-content: center;
  justify-content: flex-end;
  justify-content: space-between;
  justify-content: space-around;
  justify-content: space-evenly;
}
```

🎯 `align-items` – Vertical Alignment (Cross Axis)

```
.container {
  align-items: flex-start;
  align-items: center;
  align-items: flex-end;
  align-items: stretch;     /* default */
  align-items: baseline;
}
```

💡 Practice Task: Center a Card in the Page

```
<div class="container">
  <div class="card">I am centered</div>
</div>
```

```
.container {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}
```

```
.card {
  background-color: lightblue;
  padding: 40px;
  border-radius: 10px;
}
```

🏠 Homework

✅ Create a card that is perfectly centered in the viewport using Flexbox.

🧱 Lecture 14: Flexbox Deep Dive

📚 Topics We'll Cover:

- `flex-wrap`, `gap`
- `align-self`, `align-content`
- `flex-grow`, `flex-shrink`, `flex-basis`
- Mini Challenge: 3-column Pricing Table

🔁 `flex-wrap`

```
.container {
  flex-wrap: nowrap;   /* default */
  flex-wrap: wrap;
  flex-wrap: wrap-reverse;
}
```

✅ Allows items to wrap onto multiple lines.

📏 `gap` Between Items

```
.container {
  display: flex;
  gap: 20px;
}
```

⚠️ `gap` works directly in flex (previously only in grid!).

🎯 `align-content` – Space Between Rows (like justify-content but for lines)

```
.container {
  align-content: center;
}
```

✅ Only works when there are **multiple rows**.

🎯 `align-self` – Align a Single Item Differently

```
.item:nth-child(2) {
  align-self: flex-end;
}
```

Overrides the container's `align-items`.

📦 `flex-grow`, `flex-shrink`, `flex-basis`

```
.item {
  flex-grow: 1;       /* Takes up remaining space */
  flex-shrink: 1;     /* Shrinks when space is tight */
```

```
  flex-basis: 200px;  /* Initial size before shrinking/growing */
}
```

Shorthand:

```
.item {
  flex: 1 1 200px;  /* grow shrink basis */
}
```

🧩 Mini Challenge: 3-Column Pricing Table

Create a layout like:

Basic	Pro	Premium

```
<div class="pricing">
  <div class="card">Basic</div>
  <div class="card">Pro</div>
  <div class="card">Premium</div>
</div>
```

```
.pricing {
  display: flex;
  gap: 20px;
```

```
  justify-content: center;
  align-items: stretch;
}

.card {
  flex: 1;
  padding: 30px;
  background: #eee;
  border-radius: 10px;
  text-align: center;
}
```

🏠 Homework

✅ Build a 3-column layout with `flex-grow` and `gap`.

✅ Customize one column using `align-self`.

🧭 Lecture 15: Flexbox Project – Responsive Navigation Bar

📚 Topics We'll Cover:

- Building a responsive navigation bar
- Adding logo + navigation links
- Using `space-between` for layout
- Adding media queries for small screens

🧱 Step 1: HTML Structure

```
<nav class="navbar">
  <div class="logo">MyLogo</div>
  <ul class="nav-links">
    <li><a href="#">Home</a></li>
    <li><a href="#">About</a></li>
    <li><a href="#">Services</a></li>
    <li><a href="#">Contact</a></li>
  </ul>
</nav>
```

🎨 Step 2: Basic Flexbox Layout

```
.navbar {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 20px;
  background-color: #333;
  color: white;
}

.nav-links {
  display: flex;
  list-style: none;
  gap: 20px;
}

.nav-links a {
  text-decoration: none;
  color: white;
}
```

📱 Step 3: Responsive Breakpoint Example

```
@media (max-width: 768px) {
```

```
  .nav-links {
    flex-direction: column;
    gap: 10px;
    background-color: #444;
    padding: 10px;
  }

  .navbar {
    flex-direction: column;
    align-items: flex-start;
  }
}
```

✅ This changes the layout for smaller screens.

🏠 Homework

✅ Build your own navbar with:

- Logo on the left
- Links on the right
- Responsive layout using media queries

📢 **Next Up: Intro to Responsive Design (Lecture 16)**

📱 Lecture 16: Introduction to Responsive Design

📚 Topics We'll Cover:

- Why responsive design matters
- The viewport meta tag
- Media query `@media` syntax
- `max-width` vs `min-width`

❓ Why Responsive Design?

Modern websites need to:

- Look good on all devices (mobile, tablet, desktop)
- Adjust layout based on screen size
- Improve accessibility and usability

📱 Viewport Meta Tag (Always Include in HTML)

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

✅ This makes sure your site scales properly on mobile devices.

🧠 Understanding Media Queries

Basic Syntax:

```
@media (max-width: 600px) {
  /* CSS rules for screens ≤ 600px */
}

@media (min-width: 1024px) {
  /* CSS rules for screens ≥ 1024px */
}
```

🆚 `max-width` vs `min-width`

- `max-width`: Used for mobile-first design (styles apply **up to** a certain width)
- `min-width`: Used for desktop-first design (styles apply **from** a certain width)

Example:

```
@media (max-width: 768px) {
  body {
    background-color: lightblue;
  }
}

@media (min-width: 1025px) {
  body {
    background-color: pink;
```

```
    }
}
```

🏠 Homework

✅ Add the viewport meta tag in your HTML file.

✅ Practice writing two media queries:

- One for screens less than 768px
- One for screens more than 1024px

📱 Lecture 17: Responsive Layouts with Media Queries

📚 Topics We'll Cover:

- Creating breakpoints for mobile, tablet, and desktop
- Show/hide elements based on screen size
- Making the Flexbox navbar fully responsive

📏 Step 1: Define Breakpoints

Here are commonly used screen width breakpoints:

```
/* Mobile First */
@media (max-width: 480px) { /* Small phones */ }

@media (max-width: 768px) { /* Tablets */ }

@media (min-width: 1024px) { /* Desktops */ }
```

👻 Step 2: Hide/Show Elements at Different Sizes

```
/* Hide logo on small screens */
@media (max-width: 480px) {
  .logo {
```

```
    display: none;
  }
}

/* Show a mobile menu icon only on mobile */
@media (max-width: 768px) {
  .menu-icon {
    display: block;
  }

  .nav-links {
    display: none;
  }
}
```

🧪 Step 3: Make Navbar Fully Responsive

Continue from your previous Flexbox navbar:

- Add a hamburger icon (☰) that appears only on small screens
- Toggle nav links visibility using JavaScript or checkbox hack (optional)
- Stack items vertically on mobile

```
@media (max-width: 768px) {
  .navbar {
    flex-direction: column;
```

```
    align-items: flex-start;
  }

  .nav-links {
    flex-direction: column;
    width: 100%;
  }
}
```

🏠 Homework

✅ Add multiple breakpoints to your existing layout

✅ Hide/show specific elements as needed

✅ Make sure your navbar works perfectly on all screen sizes

🚀 Lecture 18: Final Responsive Project – Landing Page

📚 Topics We'll Cover:

- Building a **Fully Responsive Landing Page**
- Sections: Header, Hero, About, Features, Contact
- Use **Flexbox** + **Media Queries**

🧱 Step 1: HTML Structure

```
<header>MyBrand</header>

<section class="hero">
  <h1>Welcome to My Site</h1>
  <p>Your solution starts here.</p>
</section>

<section class="about">About Us</section>

<section class="features">Features</section>

<section class="contact">Contact Us</section>

<footer>© 2025 MyBrand</footer>
```

🎨 Step 2: Flexbox + Styling Example

```
.hero {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  height: 400px;
  background: linear-gradient(to right, #4facfe, #00f2fe);
  color: white;
  text-align: center;
}
```

📱 Step 3: Media Queries for Responsiveness

```
@media (max-width: 768px) {
  .hero h1 {
    font-size: 24px;
  }

  .features {
```

```
    flex-direction: column;
  }
}
```

✅ Final Outcome

- Fully structured layout
- Clean design
- Responsive across **mobile**, **tablet**, and **desktop**
- Flexbox used for section layouts
- Media queries for smooth resizing

🏁 Final Homework

✅ Build your own responsive landing page using:

- Flexbox
- Media queries
- Clear sectioning

✅ Test it on different devices (or Chrome DevTools)