Table of Contents

UNIT I:

HTML Fundamentals – Lists, Links, Tables, Forms, and Frames Lists: Ordered, Unordered, Definition, Nested, Hyperlinks and Images, Tables with formatting, HTML Forms, Frames and Layout Management

Experiments:

1. Create an HTML page demonstrating all types of lists, including nested and definition lists.
2. Design a web page using hyperlinks (<a> tag) with href and target attributes.
3. Create a gallery with thumbnail images linking to full-size versions.
4. Build a timetable using HTML tables with rowspan, colspan, and caption.
5. Design a student registration form using different input types and proper layout with tables.
6. Construct a webpage using frames to display an image, a paragraph, and a hyperlink in three distinct areas. Use the noframes tag for compatibility.

UNIT I: HTML Fundamentals – Lists, Links, Tables, Forms, and Frames

This unit introduces the foundational elements of HTML for structuring web content. You will learn how to organize information using various list types, create interactive navigation with hyperlinks, present tabular data effectively, gather user input through forms, and structure page layouts using frames.

1.1. HTML Basics Recap (Brief Review)

- **HTML:** (HyperText Markup Language) - The standard markup language for creating web pages.
- **Basic Structure:** <!DOCTYPE html>, <html>, <head>, <body>.
- **Elements, Tags, Attributes:** Understanding how they work together.
- **Text Formatting (brief):** <h1> to <h6>, <p>, <b>, <i>, <u>, , .

1.2. Lists

HTML provides different ways to organize items into lists, improving readability and semantic meaning.

1.2.1. Ordered Lists (<ol>)

- Used for items where the order matters (e.g., steps in a recipe, top 10 list).
- Each list item is defined by the <li> tag.
- **Attributes:**
  - type: Specifies the numbering style (e.g., 1 for numbers, a for lowercase letters, A for uppercase letters, i for lowercase Roman numerals, I for uppercase Roman numerals).
  - start: Specifies the starting value of the list items.
  - reversed: Specifies that the list order should be descending.

Example:

HTML

```
<ol type="1" start="5">
  <li>First item</li>
  <li>Second item</li>
  <li>Third item</li>
</ol>
```

1.2.2. Unordered Lists (<ul>)

- Used for items where the order does not matter (e.g., a list of ingredients, features).
- Each list item is defined by the <li> tag.
- **Attributes (Deprecated, use CSS for styling):**
  - type: Specifies the marker style (e.g., disc, circle, square). *It's recommended to use CSS list-style-type for styling.*

Example:

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

1.2.3. Definition Lists (<dl>)

- Used to define terms and their descriptions.
- Consists of pairs: <dt> (definition term) and <dd> (definition description).

Example:

HTML
```
<dl>
  <dt>HTML</dt>
  <dd>HyperText Markup Language</dd>
  <dt>CSS</dt>
  <dd>Cascading Style Sheets</dd>
</dl>
```

1.2.4. Nested Lists

- Lists can be nested within other list items to create hierarchical structures.

Example:

HTML
```
<ul>
  <li>Fruits
    <ul>
      <li>Apple</li>
      <li>Banana</li>
    </ul>
  </li>
  <li>Vegetables
    <ol>
      <li>Carrot</li>
      <li>Broccoli</li>
    </ol>
  </li>
</ul>
```

1.3. Hyperlinks and Images

These elements are fundamental for navigating the web and embedding visual content.

1.3.1. Hyperlinks (<a>)

- Used to create links to other web pages, files, or locations within the same page.
- The text or image between the <a> tags becomes the clickable link.
- **Attributes:**
  - href (Hypertext Reference): Specifies the URL of the linked resource.
    - **Absolute URLs:** Link to external websites (e.g., https://www.google.com).
    - **Relative URLs:** Link to pages within the same website (e.g., about.html, images/pic.jpg).
    - **Anchor Links:** Link to a specific section within the same page or another page (e.g., Go to Section). The target section needs an id attribute: <h2 id="section_id">Section Title</h2>.
  - target: Specifies where to open the linked document.

        - _self (default): Opens in the same frame.
        - _blank: Opens in a new window/tab.
        - _parent: Opens in the parent frame.
        - _top: Opens in the full body of the window.
    - title: Provides a tooltip text when hovering over the link.

Example:

HTML

```
<a href="https://www.example.com" target="_blank" title="Visit Example.com">Visit Example</a>
<a href="about.html">About Us</a>
<a href="#top_section">Back to Top</a>
```

1.3.2. Images (<img>)

- Used to embed images into a web page.
- It is an empty tag (does not have a closing tag).
- **Attributes:**
    - src (Source): Specifies the URL of the image file.
    - alt (Alternative text): Provides a text description of the image for accessibility (screen readers) and when the image cannot be displayed.
    - width: Specifies the width of the image in pixels or percentage.
    - height: Specifies the height of the image in pixels or percentage.
    - title: Provides a tooltip text when hovering over the image.

Example:

HTML

```
<img src="images/logo.png" alt="Company Logo" width="150" height="auto" title="Our Company Logo">
```

1.4. Tables (<table>)

Tables are used to display tabular data (rows and columns) in a structured format.

1.4.1. Basic Table Structure

- <table>: The container for the entire table.
- <caption>: (Optional) Provides a title for the table.
- <thead>: Groups the header content in a table (for semantic purposes).
- <tbody>: Groups the body content in a table (for semantic purposes).
- <tfoot>: Groups the footer content in a table (for semantic purposes).
- <tr> (Table Row): Defines a row in the table.
- <th> (Table Header): Defines a header cell within a row. Typically bold and centered by default.
- <td> (Table Data): Defines a standard data cell within a row.

Example:

HTML

```
<table>
  <caption>Monthly Sales Report</caption>
  <thead>
    <tr>
      <th>Month</th>
      <th>Sales</th>
      <th>Profit</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>January</td>
      <td>$1000</td>
```

```
      <td>$300</td>
    </tr>
    <tr>
      <td>February</td>
      <td>$1200</td>
      <td>$400</td>
    </tr>
  </tbody>
</table>
```

1.4.2. Table Formatting (Using Attributes - some are deprecated, prefer CSS)

While HTML attributes exist for basic styling, **CSS is the preferred method for table formatting** (borders, padding, background colors, etc.). However, for understanding, here are some HTML attributes:

- **border**: Specifies the thickness of the table border (e.g., border="1").
- **cellpadding**: Specifies the space between the cell content and its border.
- **cellspacing**: Specifies the space between cells.
- **width**: Specifies the width of the table or a column/cell.
- **height**: Specifies the height of the table or a row/cell.
- **align**: (Deprecated) Horizontal alignment for table/content (e.g., left, center, right).
- **valign**: (Deprecated) Vertical alignment for cell content (e.g., top, middle, bottom).
- **bgcolor**: (Deprecated) Background color.

1.4.3. Spanning Rows and Columns (rowspan, colspan)

- colspan: Merges a cell across multiple columns.
  - **Attribute:** colspan="number_of_columns_to_span"
- rowspan: Merges a cell across multiple rows.
  - **Attribute:** rowspan="number_of_rows_to_span"

Example:

HTML

```
<table>
  <tr>
    <th>Name</th>
    <th colspan="2">Phone</th>
  </tr>
  <tr>
    <td>John Doe</td>
    <td>555-1234</td>
    <td>555-5678</td>
  </tr>
  <tr>
    <td rowspan="2">Jane Smith</td>
    <td colspan="2">555-9876</td>
  </tr>
  <tr>
    <td colspan="2">555-4321</td>
  </tr>
</table>
```

1.5. HTML Forms (<form>)

Forms are used to collect user input, such as text, selections, and files, to send to a server.

1.5.1. Basic Form Structure

- <form>: The container for all form elements.

- **Attributes:**
  - action: Specifies the URL where the form data will be sent when submitted.
  - method: Specifies the HTTP method to use for sending data.
    - GET: Appends form data to the URL (visible in URL, limited data). Suitable for idempotent requests (e.g., search queries).
    - POST: Sends form data in the body of the HTTP request (not visible in URL, larger data). Suitable for sensitive data or data that changes state on the server (e.g., submitting a new post).
  - enctype: (For method="post") Specifies how the form data should be encoded, especially for file uploads (multipart/form-data).

Example:

HTML
```
<form action="/submit_form.php" method="post">
  </form>
```

1.5.2. Form Input Elements (<input>)

The <input> tag is versatile and has various types.

- **Attributes:**
  - type: Specifies the type of input control.
  - name: Essential for identifying the input field when data is sent to the server.
  - value: The initial value of the input or the value sent to the server.
  - id: For linking with <label> for accessibility.
  - placeholder: Hint text shown in the input field.
  - required: Makes the field mandatory.
- **Common type values:**
  - text: Single-line text input.

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

  - password: Text input, characters masked.

HTML
```
<label for="password">Password:</label>
<input type="password" id="password" name="password">
```

  - submit: Button to submit the form.

HTML
```
<input type="submit" value="Submit">
```

  - reset: Button to reset form fields to their initial values.

HTML
```
<input type="reset" value="Clear Form">
```

  - checkbox: Allows selection of zero or more options.

HTML
```
<input type="checkbox" id="newsletter" name="newsletter" value="yes">
<label for="newsletter">Subscribe to newsletter</label>
```

  - radio: Allows selection of exactly one option from a group (same name attribute for grouping).

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

  - button: A clickable button (use with JavaScript).

HTML
```
<input type="button" value="Click Me">
```

- file: For uploading files.

HTML
```
<label for="upload_file">Upload File:</label>
<input type="file" id="upload_file" name="document">
```

- hidden: An input field that is not visible to the user but is sent with the form data.

HTML
```
<input type="hidden" name="user_id" value="123">
```

- email, number, date, url, tel, range, color, etc. (HTML5 new types for better validation and user experience).

1.5.3. Other Form Elements

- <textarea>: Multi-line text input.
  - **Attributes:** rows, cols.

HTML
```
<label for="comments">Comments:</label><br>
<textarea id="comments" name="comments" rows="5" cols="40"></textarea>
```

- <select>: Drop-down list (single or multiple selection).
  - <option>: Defines an item in the list.
  - **Attributes:** multiple (for multiple selections), size (number of visible options), selected (default selected option).

HTML
```
<label for="country">Country:</label>
<select id="country" name="country">
  <option value="usa">United States</option>
  <option value="can">Canada</option>
  <option value="mex" selected>Mexico</option>
</select>
```

- <datalist>: Provides a list of pre-defined options for an <input> element.

HTML
```
<label for="browser">Choose your browser:</label>
<input list="browsers" name="browser" id="browser">
<datalist id="browsers">
  <option value="Edge">
  <option value="Firefox">
  <option value="Chrome">
</datalist>
```

- <fieldset>: Groups related elements in a form.
- <legend>: Provides a caption for the <fieldset>.

HTML
```
<fieldset>
  <legend>Personal Information:</legend>
  <label for="fname">First Name:</label>
  <input type="text" id="fname" name="fname">
  </fieldset>
```

- <label>: Associates text with a form control. Crucial for accessibility.
  - The for attribute of the label must match the id attribute of the input.

1.6. Frames & Layout Management (Deprecated - Use CSS Layouts)

Note: The use of HTML frames (<frameset>, <frame>, <noframes>) for page layout is **highly deprecated** and generally discouraged in modern web development. They have

significant usability, accessibility, and SEO issues. Modern web design uses CSS (Flexbox, Grid) for layout management.

However, since it's part of the unit material, here's a brief overview:

1.6.1. Introduction to Frames

- Frames allowed splitting a browser window into multiple independent sub-windows, each displaying a separate HTML document.
- Defined using the <frameset> and <frame> tags.
- The <body> tag is not used directly within a document that uses <frameset>.

1.6.2. <frameset> Tag

- Replaced the <body> tag in a framed document.
- **Attributes:**
  - rows: Defines the height of each row in pixels, percentage, or relative sizing (*).
  - cols: Defines the width of each column in pixels, percentage, or relative sizing (*).

Example:

HTML

```
<frameset rows="20%, 80%">
  </frameset>

<frameset cols="25%, 75%">
  </frameset>
```

1.6.3. <frame> Tag

- Defines a single frame within a <frameset>.
- **Attributes:**
  - src: Specifies the URL of the HTML document to display in the frame.
  - name: Assigns a name to the frame (important for targeting links).
  - noresize: Prevents the user from resizing the frame.
  - scrolling: Controls scrollbars (yes, no, auto).
  - frameborder: Specifies if borders should be shown around the frame.

Example (combining frameset and frame):

HTML

```
<!DOCTYPE html>
<html>
<head>
  <title>Frames Example</title>
</head>
<frameset rows="100, *">
  <frame src="header.html" name="header_frame" noresize>
  <frameset cols="200, *">
    <frame src="navbar.html" name="nav_frame">
    <frame src="content.html" name="main_content">
  </frameset>
  <noframes>
    <body>
      <p>Your browser does not support frames.</p>
    </body>
  </noframes>
</frameset>
</html>
```

1.6.4. <noframes> Tag

- Provides alternative content for browsers that do not support frames (or when JavaScript is disabled).
- Content within <noframes> is displayed instead of the frames.

1.6.5. Layout Management with Frames

- **Targeting Links:** Using the target attribute in <a> tags to open a linked document in a specific named frame (e.g., Open Page).

Why Frames are Deprecated:

- **SEO Issues:** Search engines struggled to index framed content.
- **Accessibility:** Difficult for screen readers and other assistive technologies.
- **Bookmarks:** Hard to bookmark a specific page within a frameset.
- **Usability:** Back button issues, printing issues, inconsistent URLs.
- **Development Complexity:** More difficult to manage multiple HTML files for a single logical page.

Modern Alternatives for Layout:

- **CSS Flexbox:** For one-dimensional layouts (rows or columns).
- **CSS Grid:** For two-dimensional layouts (rows and columns simultaneously).
- **<iframe> (Inline Frame):** Still used to embed *another* independent HTML document within a specific area of a parent document (e.g., embedding a YouTube video or a Google Map). This is different from frameset-based layouts.

Unit I: HTML Fundamentals – Lists, Links, Tables, Forms, and Frames

Experiment No.: 1

Title: Create an HTML page demonstrating all types of lists, including nested and definition lists.

---

1. Aim

To create an HTML page that demonstrates all types of lists in HTML, including **ordered lists**, **unordered lists**, **definition lists**, and **nested lists**.

---

2. Objectives

- To understand the structure and usage of various HTML list types.
- To apply formatting and nesting to lists.
- To use <ol>, <ul>, <dl>, <li>, <dt>, and <dd> tags effectively.

---

3. Software & Hardware Requirements

- **Software:** Any text editor (Notepad, VS Code, Sublime Text), Web browser (Chrome, Firefox, Edge).
- **Hardware:** PC or Laptop with basic configuration.

---

4. Theory

HTML supports the following types of lists:

1. **Ordered List (<ol>)**
   Displays list items in a specific order using numbers, letters, or Roman numerals.
2. **Unordered List (<ul>)**
   Displays items with bullets.
3. **Definition List (<dl>)**
   Contains terms and their corresponding descriptions.
4. **Nested List**
   A list within another list, used for hierarchical data.

Tags Used:

- : Ordered list container
- : Unordered list container
- : List item
- <dl> : Definition list container
- <dt> : Definition term
- <dd> : Definition description

---

5. Procedure

1. Open a text editor and create a new HTML file.
2. Use the basic HTML structure (<html>, <head>, <body>).
3. Add headings to indicate the type of list.
4. Create examples for:
   - Ordered list with numbering styles.
   - Unordered list with bullet styles.
   - Nested lists (combination of ordered and unordered lists).
   - Definition list with terms and descriptions.
5. Save the file as lists.html.
6. Open the file in a web browser to view the output.

---

6. Program Code

```
<!DOCTYPE html>
```

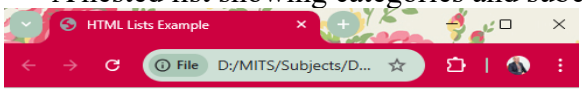
```
<html>
<head>
  <title>HTML Lists Example</title>
</head>
<body>
  <h1>HTML Lists Demonstration</h1>

  <h2>1. Ordered List</h2>
  <ol type="1">
    <li>HTML</li>
    <li>CSS</li>
    <li>JavaScript</li>
  </ol>

  <h2>2. Unordered List</h2>
  <ul type="circle">
    <li>Apple</li>
    <li>Banana</li>
    <li>Cherry</li>
  </ul>

  <h2>3. Definition List</h2>
  <dl>
    <dt>HTML</dt>
    <dd>HyperText Markup Language</dd>
    <dt>CSS</dt>
    <dd>Cascading Style Sheets</dd>
  </dl>

  <h2>4. Nested List</h2>
  <ol>
    <li>Programming Languages
      <ul>
        <li>Python</li>
        <li>Java</li>
        <li>C++</li>
      </ul>
    </li>
    <li>Web Technologies
      <ul>
        <li>HTML</li>
        <li>CSS</li>
        <li>JavaScript</li>
      </ul>
    </li>
  </ol>
</body>
</html>
```

7. Output

When the above code is opened in a browser, it will display:

- A numbered list of HTML, CSS, JavaScript.

- A bulleted list of fruits.
- A definition list explaining HTML and CSS.
- A nested list showing categories and subcategories.



HTML Lists Demonstration

1. Ordered List

1. HTML
2. CSS
3. JavaScript

2. Unordered List

- Apple
- Banana
- Cherry

3. Definition List

HTML
: HyperText Markup Language

CSS
: Cascading Style Sheets

4. Nested List

1. Programming Languages
   - Python
   - Java
   - C++
2. Web Technologies
   - HTML
   - CSS
   - JavaScript

8. Result

The HTML page was successfully created to demonstrate **ordered, unordered, definition, and nested lists**.

9. Viva Questions

1. What is the difference between <ol> and <ul> tags?
2. How can you change the numbering style of an ordered list?
3. What are <dl>, <dt>, and <dd> tags used for?
4. Give an example of when you would use a nested list in a website.
5. How do you change bullet styles in an unordered list?

Unit I: HTML Fundamentals – Lists, Links, Tables, Forms, and Frames

Experiment No.: 2

Title: Design a web page using hyperlinks (<a> tag) with href and target attributes

---

1. Aim

To design a web page that demonstrates the use of HTML hyperlinks using the <a> tag with **href** and **target** attributes to navigate between web pages or sections.

---

2. Objectives

- Understand the purpose and syntax of the <a> tag in HTML.
- Learn how to link to external websites, internal pages, and sections within a page.
- Use the **target** attribute to control how a link is opened.

---

3. Software & Hardware Requirements

- **Software:** Any text editor (Notepad, VS Code, Sublime Text), Web browser (Chrome, Firefox, Edge).
- **Hardware:** PC or Laptop with basic configuration.

---

4. Theory

A **hyperlink** is a clickable element in a webpage that directs the user to another location (same page, different page, or external website).

Syntax:

html

```
<a href="URL" target="value">Link Text</a>
```

Important Attributes:

- **href** → Specifies the destination URL or path.
- **target** → Specifies where to open the linked document:
  - _self → Opens in the same tab (default).
  - _blank → Opens in a new tab/window.
  - _parent → Opens in the parent frame.
  - _top → Opens in the full window.

---

5. Procedure

1. Open a text editor and create a new HTML file.
2. Add the basic HTML structure (<html>, <head>, <body>).
3. Create multiple hyperlinks using <a> tags:
   - Link to an **external website**.
   - Link to another **HTML page** in the same folder.
   - Link to a **specific section** within the same page using an id.
4. Apply the **target** attribute to open links in the same tab, new tab, or specific frames.
5. Save the file as hyperlinks.html.
6. Open it in a browser and test the links.

---

6. Program Code

```
<!DOCTYPE html>
<html>
<head>
  <title>Hyperlinks Example</title>
</head>
<body>
  <h1>HTML Hyperlinks Demo</h1>
```

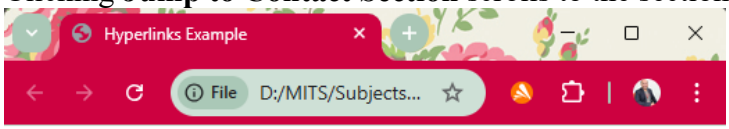
```
    <h2>1. External Link (Open in Same Tab)</h2>
    <a href="https://www.google.com" target="_self">Go to Google</a>

    <h2>2. External Link (Open in New Tab)</h2>
    <a href="https://www.wikipedia.org" target="_blank">Go to Wikipedia</a>

    <h2>3. Internal Page Link</h2>
    <a href="about.html" target="_self">Go to About Page</a>

    <h2>4. Link to a Section within the Same Page</h2>
    <a href="#contact">Jump to Contact Section</a>

    <h2>Some Content...</h2>
    <p>Scroll down to find the contact section.</p>

    <h3 id="contact">Contact Section</h3>
    <p>Email: example@example.com</p>
</body>
</html>
```

---

7. Output

When opened in a browser:

- Clicking **Go to Google** opens Google in the same tab.
- Clicking **Go to Wikipedia** opens Wikipedia in a new tab.
- Clicking **Go to About Page** opens another HTML file in the same folder.
- Clicking **Jump to Contact Section** scrolls to the section with id="contact".



HTML Hyperlinks Demo

1. External Link (Open in Same Tab)

Go to Google

2. External Link (Open in New Tab)

Go to Wikipedia

3. Internal Page Link

Go to About Page

4. Link to a Section within the Same Page

Jump to Contact Section

Some Content...

Scroll down to find the contact section.

Contact Section

Email: example@example.com

8. Result

The web page was successfully created to demonstrate the use of hyperlinks with **href** and **target** attributes.

9. Viva Questions

1. What is the purpose of the <a> tag in HTML?
2. What does the href attribute do?
3. How does the target="_blank" attribute behave?
4. How can you link to a section within the same web page?
5. What happens if you omit the target attribute?

Unit I: HTML Fundamentals – Lists, Links, Tables, Forms, and Frames

Experiment No.: 3

Title: Create a gallery with thumbnail images linking to full-size versions

---

1. Aim

To design an HTML web page that displays a gallery of thumbnail images, where each thumbnail is linked to its corresponding full-size image using hyperlinks.

---

2. Objectives

- To understand how to embed images in HTML using the <img> tag.
- To use hyperlinks (<a> tag) to link thumbnail images to larger versions.
- To structure a responsive image gallery layout.

---

3. Software & Hardware Requirements

- **Software:** Any text editor (Notepad, VS Code, Sublime Text), Web browser (Chrome, Firefox, Edge).
- **Hardware:** PC or Laptop with basic configuration.

---

4. Theory

HTML provides the <img> tag for displaying images.
We can create clickable image thumbnails using the <a> tag, where:

- The <a> tag's **href** points to the full-size image.
- The <img> tag inside <a> displays a smaller thumbnail version.

Basic Syntax:

```
<a href="fullsize.jpg" target="_blank">
  <img src="thumbnail.jpg" alt="Description">
</a>
```

Key Attributes:

- **src** → Image file path (thumbnail).
- **alt** → Alternate text if image cannot be displayed.
- **href** → Link to full-size image.
- **target="_blank"** → Opens the full-size image in a new tab.

---

5. Procedure

1. Prepare a set of images — one thumbnail and one full-size version for each.
2. Save them in an images folder for easy reference.
3. Open a text editor and create a new HTML file.
4. Create a or table to arrange thumbnails in a grid.
5. For each thumbnail, wrap the <img> tag inside an <a> tag pointing to the full-size image.
6. Add CSS (optional) for spacing and layout.
7. Save the file as gallery.html.
8. Open it in a browser to test the gallery.

---

6. Program Code

```
<!DOCTYPE html>
<html>
<head>
  <title>Image Gallery</title>
  <style>
    body {
```

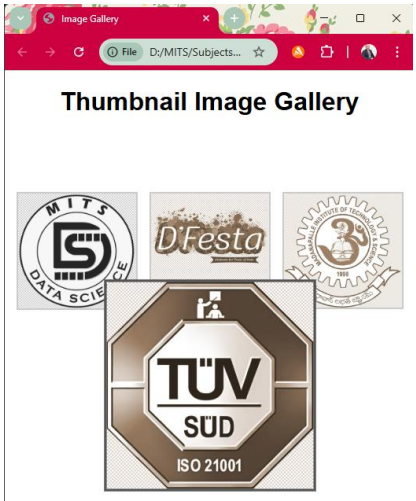
```
        font-family: Arial, sans-serif;
        text-align: center;
      }
      .gallery {
        display: flex;
        flex-wrap: wrap;
        justify-content: center;
        gap: 15px;
      }
      .gallery img {
        border: 2px solid #ccc;
        width: 150px;
        height: auto;
        transition: transform 0.2s;
      }
      .gallery img:hover {
        transform: scale(5.05);
        border-color: #555;
      }
    </style>
</head>
<body>
    <h1>Thumbnail Image Gallery</h1>
    <div class="gallery">
      <a href="images/full1.jpg" target="_blank">
        <img src="images/thumb1.jpg" alt="Image 1">
      </a>
      <a href="images/full2.jpg" target="_blank">
        <img src="images/thumb2.jpg" alt="Image 2">
      </a>
      <a href="images/full3.jpg" target="_blank">
        <img src="images/thumb3.jpg" alt="Image 3">
      </a>
      <a href="images/full4.jpg" target="_blank">
        <img src="images/thumb4.jpg" alt="Image 4">
      </a>
    </div>
</body>
</html>
```

---

7. Output

When the above code is opened in a browser:

- A gallery of clickable thumbnails appears.
- Clicking a thumbnail opens the corresponding full-size image in a new tab.



8. Result

The HTML page was successfully created to display a clickable thumbnail gallery linking to full-size images.

9. Viva Questions

1. Which HTML tag is used to insert an image?
2. How do you make an image clickable in HTML?
3. What is the purpose of the alt attribute in <img>?
4. How can you open an image in a new tab from a thumbnail?
5. What is the advantage of using thumbnail images instead of directly displaying large images?

Unit I: HTML Fundamentals – Lists, Links, Tables, Forms, and Frames

Experiment No.: 4

Title: Build a timetable using HTML tables with rowspan, colspan, and caption

---

1. Aim

To design an HTML web page displaying a timetable using tables, incorporating rowspan, colspan, and caption attributes for proper structuring.

---

2. Objectives

- To understand HTML <table> structure.
- To use table attributes like rowspan and colspan for merging cells.
- To add captions for descriptive table headings.

---

3. Software & Hardware Requirements

- **Software:** Any text editor (Notepad, VS Code, Sublime Text), Web browser (Chrome, Firefox, Edge).
- **Hardware:** PC or Laptop with basic configuration.

---

4. Theory

HTML tables allow us to display data in rows and columns.
Key elements:

- <table> → Defines the table.
- <tr> → Table row.
- <th> → Table header cell.
- <td> → Table data cell.
- <caption> → Describes the table.
- **rowspan** → Merges cells vertically.
- **colspan** → Merges cells horizontally.

Basic Syntax:
html

```
<td rowspan="2">Text</td>
<td colspan="3">Text</td>
```

---

5. Procedure

1. Open a text editor and create a new HTML file.
2. Start with <html> and <body> tags.
3. Create a <table> with <caption> for the timetable title.
4. Use <th> for headings and <td> for normal cells.
5. Apply rowspan for cells spanning multiple rows.
6. Apply colspan for cells spanning multiple columns.
7. Add some CSS for better formatting.
8. Save the file as timetable.html.
9. Open in a web browser to check the output.

---

6. Program Code

```
<!DOCTYPE html>
<html>
<head>
  <title>Class Timetable</title>
  <style>
```

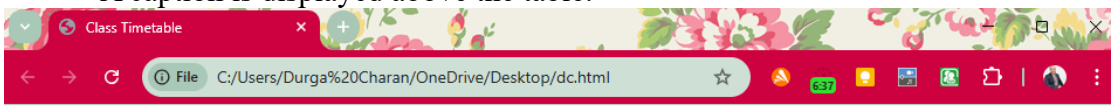
```
      table {
         border-collapse: collapse;
         width: 80%;
         margin: auto;
      }
      caption {
         font-size: 1.5em;
         font-weight: bold;
         margin-bottom: 10px;
      }
      th, td {
         border: 1px solid #333;
         padding: 10px;
         text-align: center;
      }
      th {
         background-color: #f2f2f2;
      }
   </style>
</head>
<body>
   <table>
      <caption>B.Tech CSE (Data Science)– Class Timetable</caption>
      <tr>
         <th><b>Day</b></th>
         <th><b>9:10 - 10:10</b></th>
         <th><b>10:10 - 11:10</b></th>
         <th><b>11:10 - 12:10</b></th>
         <th><b>1:00 - 2:00</b></th>
         <th><b>2:00 - 3:00</b></th>
         <th><b>3:00 - 4:00</b></th>
         <th><b>4:00 - 5:00</b></th>
      </tr>
      <tr>
         <td ><b>Monday</b></td>
         <td>FOAI(MOOC)</td>
         <td>23CSD109</td>
         <td>23CSD108</td>
          <td>OE 1</td>
         <td colspan ="3">23CSD205</td>
      </tr>
      <tr>
         <td><b>Tuesday</b></td>
         <td>23CSD107</td>
         <td>23CSD109</td>
         <td>Verbal</td>
  <td>OE 1</td>
            <td>23PHY102</td>
         <td>23ENG901</td>
         <td>23CSD108</td>
         </tr>
      <tr>
```

```
        <td><b>Wednesday</b></td>
        <td>IOS(MOOC)</td>
        <td colspan="2">TT</td>
        <td>OE 1</td>
        <td>FOAI(MOOC)</td>
        <td colspan="2">23CSD604 (SEC)</td>
      </tr>
      <tr>
        <td><b>Thursday</b></td>
        <td>23CSD108</td>
        <td rowspan ="2">23PHY102</td>
        <td>IOS(MOOC)</td>
        <td colspan="2">TT</td>
        <td>23CSD604(SEC)</td>
        <td>23ENG901</td>
      </tr>
      <tr>
        <td><b>Friday</b></td>
        <td>23CSD107</td>
        <td>APS</td>
        <td>FOAI(MOOC)</td>
        <td colspan="3">23CSD206</td>
      </tr>
      <tr>
        <td><b>Saturday</b></td>
        <td>IOS(MOOC)</td>        <td>23CSD107</td>        <td>23CSD109</td>
        <td colspan="4">23CSD701</td>
      </tr>
    </table>
</body>
</html>
```

7. Output

When viewed in a browser:

- A neatly formatted timetable appears.
- Some cells span multiple rows or columns using rowspan and colspan.
- A caption is displayed above the table.



B.Tech CSE (Data Science)– Class Timetable

Day	9:10 - 10:10	10:10 - 11:10	11:10 - 12:10	1:00 - 2:00	2:00 - 3:00	3:00 - 4:00	4:00 - 5:00
Monday	FOAI(MOOC)	23CSD109	23CSD108	OE 1	23CSD205		
Tuesday	23CSD107	23CSD109	Verbal	OE 1	23PHY102	23ENG901	23CSD108
Wednesday	IOS(MOOC)	TT		OE 1	FOAI(MOOC)	23CSD604 (SEC)	
Thursday	23CSD108	23PHY102	IOS(MOOC)	TT		23CSD604(SEC)	23ENG901
Friday	23CSD107		APS	FOAI(MOOC)	23CSD206		
Saturday	IOS(MOOC)	23CSD107	23CSD109	23CSD701			

8. Result

The HTML timetable was successfully created using rowspan, colspan, and caption.

---

9. Viva Questions

1. What is the purpose of rowspan in HTML tables?
2. What is the difference between <th> and <td>?
3. How is colspan useful in table formatting?
4. Where is the <caption> tag placed in an HTML table?
5. How can you style tables in HTML?

Unit I: HTML Fundamentals – Lists, Links, Tables, Forms, and Frames

Experiment No.: 5

Title: Design a student registration form using different input types and proper layout with tables

---

1. Aim

To design an HTML student registration form that collects personal, academic, and contact details using various HTML input types and organizes them neatly using tables.

---

2. Objectives

- To learn HTML form structure and different input types.
- To use tables for form layout.
- To apply attributes like name, value, placeholder, required, and type.

---

3. Software & Hardware Requirements

- **Software:** Any text editor (Notepad, VS Code, Sublime Text) and a web browser (Chrome, Firefox, Edge).
- **Hardware:** PC or Laptop with basic configuration.

---

4. Theory

HTML forms are used to collect user input.

Key Tags & Attributes:

- <form> → Defines a form.
- <input> → Accepts user input (types: text, number, email, date, password, radio, checkbox, submit, reset).
- <select> & <option> → Dropdown list.
- <textarea> → Multi-line text box.
- name → Identifies form data for backend processing.
- placeholder → Displays hint text inside input fields.
- required → Makes a field mandatory.

Using Tables for Layout:

Tables allow aligning labels and inputs neatly in rows and columns.

---

5. Procedure

1. Open a text editor and create a new HTML file.
2. Create a <form> element and set attributes like action and method.
3. Use a <table> to organize form fields in rows and columns.
4. Add various <input> types like text, password, email, number, date, radio, and checkbox.
5. Use <select> for dropdowns and <textarea> for multi-line inputs.
6. Add Submit and Reset buttons.
7. Save the file as student_registration.html.
8. Open it in a web browser to check the output.

---

6. Program Code

```
<!DOCTYPE html>
<html>
<head>
  <title>Student Registration Form</title>
  <style>
    table {
```

```
        border-collapse: collapse;
        margin: auto;
        background-color: #f9f9f9;
        padding: 20px;
      }
      td {
        padding: 10px;
      }
      h2 {
        text-align: center;
        color: #333;
      }
      input, select, textarea {
        width: 100%;
        padding: 6px;
      }
      input[type=radio], input[type=checkbox] {
        width: auto;
      }
      .center {
        text-align: center;
      }
 body {
        font-family: Arial, sans-serif;
        margin: 40px;
      }
      .success-message {
        color: green;
        margin-top: 15px;
        display: none; /* Hidden until shown */
      }
      input, button {
        padding: 8px;
        margin-top: 10px;
      }
  </style>
</head>
<body>
  <h2>Student Registration Form</h2>
  <form          id="myForm" action="#" method="post">
    <table border="1">
      <tr>
        <td>First Name:</td>
        <td><input type="text" name="first_name" placeholder="Enter first name"
required></td>
      </tr>
      <tr>
        <td>Last Name:</td>
        <td><input type="text" name="last_name" placeholder="Enter last name"
required></td>
      </tr>
      <tr>
```

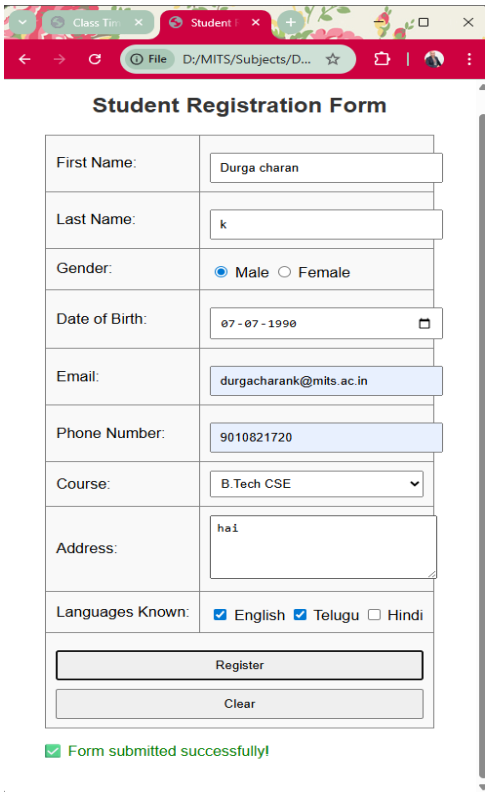
```
            <td>Gender:</td>
            <td>
              <input type="radio" name="gender" value="Male" required> Male
              <input type="radio" name="gender" value="Female"> Female
            </td>
          </tr>
          <tr>
            <td>Date of Birth:</td>
            <td><input type="date" name="dob" required></td>
          </tr>
          <tr>
            <td>Email:</td>
            <td><input type="email" name="email" placeholder="Enter email" required></td>
          </tr>
          <tr>
            <td>Phone Number:</td>
            <td><input type="tel" name="phone" pattern="[0-9]{10}" placeholder="10-digit
number" required></td>
          </tr>
          <tr>
            <td>Course:</td>
            <td>
              <select name="course" required>
                <option value="">--Select Course--</option>
                <option value="B.Tech CSE">B.Tech CSE</option>
                <option value="B.Tech ECE">B.Tech ECE</option>
                <option value="B.Tech MECH">B.Tech MECH</option>
                <option value="B.Tech CIVIL">B.Tech CIVIL</option>
              </select>
            </td>
          </tr>
          <tr>
            <td>Address:</td>
            <td><textarea name="address" rows="4" placeholder="Enter your address"
required></textarea></td>
          </tr>
          <tr>
            <td>Languages Known:</td>
            <td>
              <input type="checkbox" name="lang" value="English"> English
              <input type="checkbox" name="lang" value="Telugu"> Telugu
              <input type="checkbox" name="lang" value="Hindi"> Hindi
            </td>
          </tr>
          <tr>
            <td colspan="2" class="center">
              <input type="submit" value="Register">
              <input type="reset" value="Clear">
            </td>
          </tr>
        </table>
      </form>
```

```
<p id="successMsg" class="success-message">✅ Form submitted successfully!</p>

<script>
    const form = document.getElementById('myForm');
    const successMsg = document.getElementById('successMsg');

    form.addEventListener('submit', function(e) {
        e.preventDefault(); // Stop form from refreshing page
        successMsg.style.display = 'block'; // Show success message
        form.reset(); // Clear form fields
    });
</script>

</body>
</html>
```

7. Output

When opened in a browser, a neatly formatted **Student Registration Form** appears with:

- Various input types (text, radio, email, date, checkbox, dropdown).
- Layout organized using a table.
- Submit and Reset buttons for form actions.



8. Result

The Student Registration Form was successfully created using HTML with various input types and a table-based layout.

9. Viva Questions

1. What is the difference between method="get" and method="post" in forms?
2. Why do we use the name attribute in form inputs?

3. How does the placeholder attribute work?
4. What is the difference between <input type="radio"> and <input type="checkbox">?
5. Why do we use the required attribute in form fields?

Unit I: HTML Fundamentals – Lists, Links, Tables, Forms, and Frames

Experiment No.: 6

Title: Construct a webpage using frames to display an image, a paragraph, and a hyperlink in three distinct areas. Use the <noframes> tag for compatibility.

---

1. Aim

To create an HTML webpage using frames to display an image, a paragraph, and a hyperlink in three separate sections, ensuring compatibility for browsers that do not support frames by using the <noframes> tag.

---

2. Objectives

- To understand the <frameset> and <frame> tags in HTML.
- To learn how to divide the browser window into multiple sections.
- To implement <noframes> for non-frame supporting browsers.

---

3. Software & Hardware Requirements

- **Software:** Text editor (Notepad, VS Code, Sublime Text) and a web browser (Chrome, Firefox, Edge).
- **Hardware:** PC or Laptop with basic configuration.

---

4. Theory

HTML **frames** allow you to divide the browser window into multiple sections, each displaying a different document.

Important Tags:

- <frameset> – Defines the structure of the frames.
- <frame> – Specifies the source file for each frame.
- <noframes> – Provides alternative content for browsers that do not support frames.

Key Points:

- Frames can be split horizontally (rows) or vertically (cols).
- Each frame can load a separate HTML file.
- Frames are now outdated in HTML5; <iframe> is preferred, but frames are still studied for legacy purposes.

---

5. Procedure

1. Create **three HTML files**:
   - image.html → Contains an image.
   - paragraph.html → Contains a paragraph of text.
   - link.html → Contains a hyperlink.
2. Create a **main HTML file** (frameset.html) to define the frameset layout.
3. Use the <frameset> tag to divide the window into three areas.
4. Link each frame to the respective HTML file using the <frame> tag.
5. Add a <noframes> section for compatibility.
6. Save all files in the same folder.
7. Open frameset.html in a browser to test.

---

6. Program Code

File 1: image.html

```
<!DOCTYPE html>
<html>
<head>
   <title>Image Frame</title>
```

```
</head>
<body>
   <img src="college.jpg" alt="College Image" width="100%" height="100%">
</body>
</html>
```

File 2: paragraph.html

```
<!DOCTYPE html>
<html>
<head>
   <title>Paragraph Frame</title>
</head>
<body>
   <p>
      Welcome to our college website. Our institution offers a variety of courses
      designed to meet the needs of students in the modern professional world.
   </p>
</body>
</html>
```

File 3: link.html

```
<!DOCTYPE html>
<html>
<head>
   <title>Link Frame</title>
</head>
<body>
   <a href="https://www.example.com" target="_blank">Visit Example Website</a>
</body>
</html>
```

File 4: frameset.html

```
<!DOCTYPE html>
<html>
<head>
   <title>Frameset Example</title>
</head>
<frameset cols="33%,34%,33%">
   <frame src="image.html" name="imageFrame">
   <frame src="paragraph.html" name="textFrame">
   <frame src="link.html" name="linkFrame">
   <noframes>
      <body>
         <p>Your browser does not support frames. Please update your browser or
         <a href="image.html">click here</a> to view the image.</p>
      </body>
   </noframes>
</frameset>
</html>
```

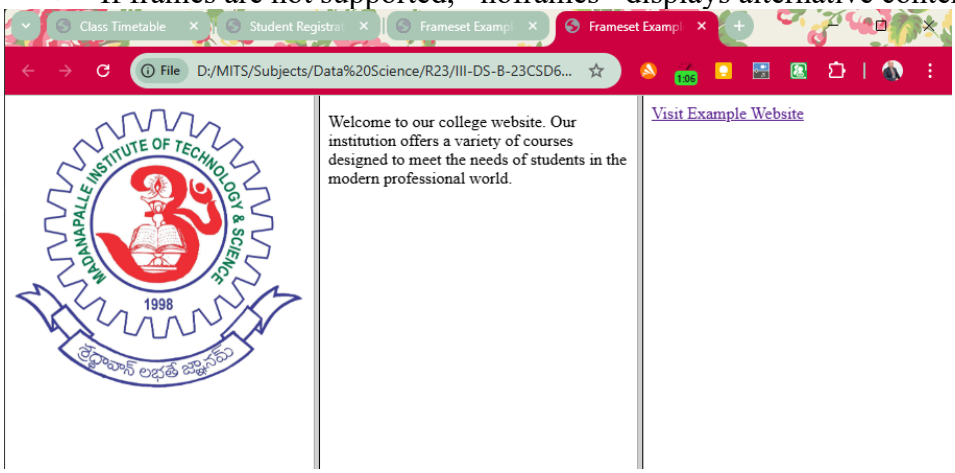
---

7. Output

When frameset.html is opened:

- The window is split into **three vertical sections**.
- **Left frame** displays an image.
- **Middle frame** displays a paragraph.

- **Right frame** displays a clickable hyperlink.
- If frames are not supported, <noframes> displays alternative content.



8. Result

A webpage with three distinct frames was successfully created, displaying an image, a paragraph, and a hyperlink with <noframes> compatibility.

9. Viva Questions

1. What is the difference between <frame> and <iframe>?
2. Why are frames not recommended in modern web design?
3. What is the use of the <noframes> tag?
4. How do you split frames horizontally instead of vertically?
5. Can you load different HTML pages in different frames? How?

UNIT II:

HTML5 and CSS Styling- HTML5 Semantic Tags: <article>, <section>, <header>, etc., Audio & Video embedding, CSS Types: Inline, Internal, External, CSS Selectors: Simple, Combinators, Pseudo-classes/elements, Attribute selectors

Experiments:

1. Develop a page using HTML5 tags like <main>, <nav>, <aside>, <figure>, etc.
2. Embed audio and video files in a web page.
3. Apply all three types of CSS to style HTML elements.
4. Demonstrate various selector types (simple, combinators, pseudo-classes/elements, attribute selectors) in a single web page.

2.1. HTML5 Semantic Tags

Semantic Tags:

- **Semantic HTML** introduces tags that clearly define their meaning in a human- and machine-readable way.
- Unlike non-semantic tags (,), semantic tags describe the purpose of the content inside them.

Advantages:

1. Improves **readability** of code.
2. Better **SEO (Search Engine Optimization)**.
3. Easier for **screen readers** and accessibility tools.
4. Encourages a **structured layout**.

Common Semantic Tags in HTML5

Tag	Meaning	Example Usage
<header>	Introductory section (title, logo, nav)	Website header with logo & menu
<footer>	Footer section (copyright, contact)	Page ending details
<article>	Self-contained content (news, blog)	Blog post, article, news story
<section>	Thematic grouping of content	A chapter or module in a course
<nav>	Navigation links	Main menu, sidebar menu
<aside>	Secondary info (ads, side notes)	Sidebar ads, tips
<main>	Central/primary content of page	Core content excluding nav/footer
<figure>	Holds image, diagram, code snippet	Image with caption
<figcaption>	Caption for figure	Caption below image
<mark>	Highlights text	Marking important keywords

Example Code:

```
<!DOCTYPE html>
<html>
<head>
  <title>Semantic Tags Example</title>
</head>
<body>
  <header>
    <h1>My Blog</h1>
    <nav>
      <a href="#">Home</a> | <a href="#">About</a>
    </nav>
  </header>
```

```
  <main>
    <article>
      <header>
        <h2>Article Title</h2>
      </header>
      <p>This is the content of the article.</p>
    </article>
  </main>

  <aside>
    <h3>Related Links</h3>
    <ul>
      <li><a href="#">Another Post</a></li>
    </ul>
  </aside>

  <footer>
    <p>&copy; 2025 My Blog</p>
  </footer>
</body>
</html>
```

2.2. Audio & Video Embedding in HTML5

Audio Embedding

- Use <audio> tag.
- Attributes:
  - controls → shows play/pause UI
  - autoplay → plays automatically
  - loop → repeats audio
  - muted → starts muted

```
<audio controls>
  <source src="music.mp3" type="audio/mpeg">
  <source src="music.ogg" type="audio/ogg">
  Your browser does not support audio tag.
</audio>
```

Video Embedding

- Use <video> tag.
- Attributes:
  - controls → shows play/pause UI
  - autoplay, loop, muted
  - poster → image before playing

```
<video width="400" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
  Your browser does not support video tag.
</video>
```

2.3. CSS Types

CSS (Cascading Style Sheets) defines how HTML elements are displayed.

Types of CSS

1. **Inline CSS**

- Written inside HTML element using style attribute.
- Example:

```
<p style="color: red; font-size: 20px;">This is inline CSS</p>
```

- ✔ Quick but ✘ not reusable.

2. **Internal CSS**
   - Written inside <style> tag within <head>.
   - Example:

```
<head>
  <style>
    p { color: blue; }
  </style>
</head>
```

   - Useful for **single page** styling.
3. **External CSS**
   - Stored in separate .css file and linked with <link>.
   - Example:

```
<head>
  <link rel="stylesheet" href="styles.css">
</head>
```

   - ✔ Reusable across multiple pages (Best practice).

2.4. CSS Selectors

Selectors define which HTML elements to style.

2.4.1. Simple Selectors

Selector	Usage	Example
Element	Selects elements by name	p { color: red; }
ID	Selects unique element	#header { background: yellow; }
Class	Selects group of elements	.btn { border: 1px solid; }
Universal (*)	Selects all elements	* { margin: 0; }
Grouping (,)	Applies to multiple	h1, h2 { color: blue; }

2.4.2. Combinators

Combinator	Meaning	Example
Descendant ( )	Selects elements inside another	div p { color: green; }
Child (>)	Direct child only	div > p { font-size: 18px; }
Adjacent sibling (+)	First immediate sibling	h1 + p { color: red; }
General sibling (~)	All siblings after	h1 ~ p { font-style: italic; }

2.4.3. Pseudo-classes

Define special states of an element.

- :hover → when mouse over
- :first-child → first child
- :last-child → last child
- :nth-child(n) → nth child

```
a:hover { color: orange; }
li:first-child { font-weight: bold; }
```

2.4.4. Pseudo-elements

Style specific parts of an element.

- ::before → insert content before
- ::after → insert content after
- ::first-letter → style first letter
- ::first-line → style first line

```
p::first-letter { font-size: 200%; color: red; }
p::after { content: " [Read more...]"; }
```

2.4.5. Attribute Selectors

Style elements based on attributes.

Selector	Example	Meaning
[attr]	input[required]	Selects elements with attribute
[attr=value]	a[target="_blank"]	Matches exact value
[attr^=value]	a[href^="https"]	Starts with value
[attr$=value]	a[href$=".pdf"]	Ends with value
[attr*=value]	a[href*="google"]	Contains value

Unit II: HTML5 and CSS Styling

Experiment No.: 7

Title: Develop a page using HTML5 tags like <main>, <nav>, <aside>, <figure>, etc.

1. Aim

To create a structured webpage using HTML5 semantic elements such as <main>, <nav>, <aside>, <figure>, <header>, <footer>, and <article> for better readability and search engine optimization.

2. Objectives

- To understand the purpose of HTML5 semantic tags.
- To implement semantic structure for improved accessibility and maintainability.
- To organize web content logically with navigation, main content, and sidebar elements.

3. Software & Hardware Requirements

- **Software:** Any text editor (Notepad, VS Code, Sublime Text) and a modern web browser (Chrome, Firefox, Edge).
- **Hardware:** Desktop or Laptop with basic configuration.

4. Theory

HTML5 introduced **semantic tags** to give meaning to webpage content beyond simple structure. They help browsers, developers, and search engines understand the purpose of different sections of a webpage.

Common HTML5 Semantic Tags:

- <header> – Represents the top section, usually containing the logo or heading.
- <nav> – Defines navigation links.
- <main> – Contains the main content of the page.
- <article> – Represents an independent, self-contained piece of content.
- <section> – Groups related content together.
- <aside> – Contains related information like sidebars, ads, or references.
- <figure> – Encapsulates images, diagrams, or illustrations.
- <figcaption> – Provides a caption for the <figure>.
- <footer> – Defines the bottom section of the page.

Advantages of Semantic Tags:

- Improves **SEO** (Search Engine Optimization).
- Enhances **accessibility** for screen readers.
- Makes code easier to read and maintain.

5. Procedure

1. Create a new HTML5 file named semantic_page.html.
2. Start with <!DOCTYPE html> to ensure HTML5 compliance.
3. Add <header> for the page title or banner.
4. Create <nav> for the menu links.
5. Use <main> to wrap the main content of the page.
6. Include an <article> and <section> for primary content blocks.
7. Add <aside> for related links or extra content.
8. Insert an image inside <figure> with a <figcaption>.
9. Finish with a <footer> for copyright and contact info.
10. Save and run the file in a browser.

6. Program Code

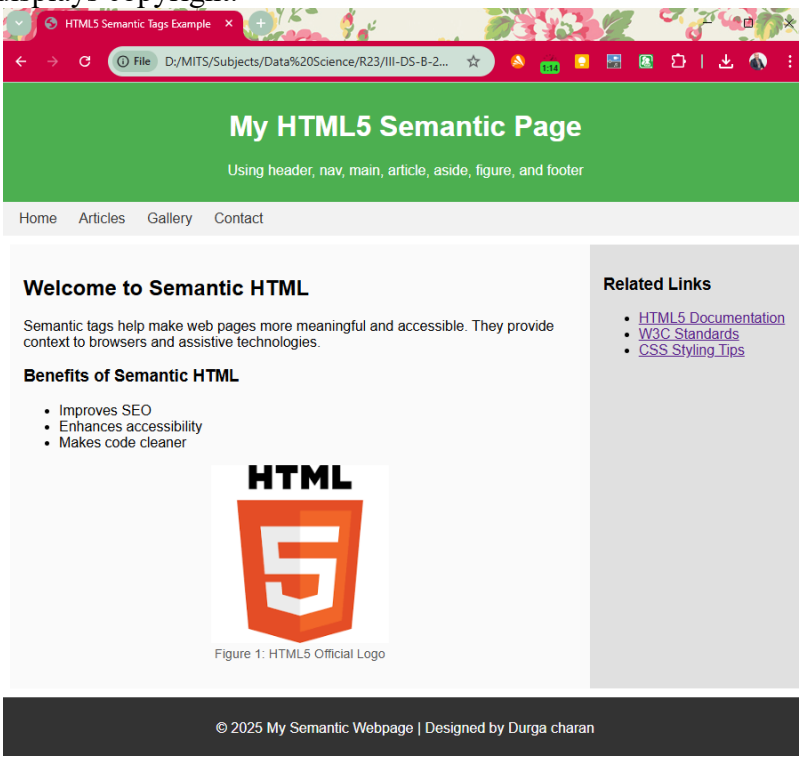
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>HTML5 Semantic Tags Example</title>
  <style>
    body { font-family: Arial, sans-serif; margin: 0; padding: 0; }
    header, nav, main, aside, footer { padding: 10px; }
    header { background-color: #4CAF50; color: white; text-align: center; }
    nav { background-color: #f2f2f2; }
    nav a { margin: 10px; text-decoration: none; color: #333; }
    main { display: flex; }
    article { flex: 3; padding: 15px; background-color: #fafafa; }
    aside { flex: 1; background-color: #e0e0e0; padding: 15px; }
    figure { text-align: center; }
    figcaption { font-size: 0.9em; color: #555; }
    footer { background-color: #333; color: white; text-align: center; padding: 10px; }
  </style>
</head>
<body>

  <header>
    <h1>My HTML5 Semantic Page</h1>
    <p>Using header, nav, main, article, aside, figure, and footer</p>
  </header>

  <nav>
    <a href="#">Home</a>
    <a href="#">Articles</a>
    <a href="#">Gallery</a>
    <a href="#">Contact</a>
  </nav>

  <main>
    <article>
      <h2>Welcome to Semantic HTML</h2>
      <p>Semantic tags help make web pages more meaningful and accessible. They
provide context to browsers and assistive technologies.</p>

      <section>
        <h3>Benefits of Semantic HTML</h3>
        <ul>
          <li>Improves SEO</li>
          <li>Enhances accessibility</li>
          <li>Makes code cleaner</li>
        </ul>
      </section>

      <figure>
        <img src=" images /html5_logo.jpg" alt="HTML5 Logo" width="200">
```

```
            <figcaption>Figure 1: HTML5 Official Logo</figcaption>
        </figure>
    </article>

    <aside>
        <h3>Related Links</h3>
        <ul>
            <li><a href="#">HTML5 Documentation</a></li>
            <li><a href="#">W3C Standards</a></li>
            <li><a href="#">CSS Styling Tips</a></li>
        </ul>
    </aside>
</main>

<footer>
    <p>&copy; 2025 My Semantic Webpage | Designed by Student</p>
</footer>

</body>
</html>
```

7. Output

When the file is opened in a browser:

- **Header** displays the title and description.
- **Nav bar** contains clickable menu links.
- **Main section** has an article with a section, a figure, and an aside sidebar.
- **Footer** displays copyright.



8. Result

The webpage was successfully created using HTML5 semantic elements <main>, <nav>, <aside>, <figure>, and others, improving structure and readability.

9. Viva Questions

1. What is the main purpose of HTML5 semantic tags?
2. Differentiate between <section> and <article>.
3. Why should we use <figure> and <figcaption> instead of just <img>?
4. How does semantic HTML improve SEO?
5. Give examples of semantic and non-semantic HTML tags.

Unit II: HTML5 and CSS Styling

Experiment No.: 8

Title: Embed audio and video files in a web page

---

1. Aim

To embed audio and video content in a webpage using HTML5 <audio> and <video> tags with various attributes for control and playback.

---

2. Objectives

- To understand HTML5 multimedia tags for audio and video.
- To learn how to use different attributes like controls, autoplay, loop, and muted.
- To make media content accessible across modern browsers.

---

3. Software & Hardware Requirements

- **Software:** Any text editor (VS Code, Sublime Text, Notepad) and a modern browser (Chrome, Firefox, Edge).
- **Hardware:** Desktop or Laptop with multimedia playback capability.

---

4. Theory

HTML5 introduced native multimedia support without relying on third-party plugins like Flash.

Audio Tag

html

```
<audio src="file.mp3" controls></audio>
```

- **Attributes:**
  - controls – Displays play, pause, volume controls.
  - autoplay – Starts playing automatically.
  - loop – Plays the audio repeatedly.
  - muted – Mutes audio initially.
  - preload – Instructs browser to preload audio (auto, metadata, none).

Video Tag

html

```

<video src="file.mp4" controls></video>
```

- **Attributes:**
  - controls – Displays video player controls.
  - autoplay – Plays video automatically on page load.
  - loop – Repeats video.
  - muted – Mutes video initially.
  - poster – Displays an image before the video starts.

Advantages:

- No need for external plugins.
- Cross-browser compatibility (with correct formats).
- Better integration with HTML5 and JavaScript APIs.

---

5. Procedure

1. Create a new HTML5 file named media_embed.html.
2. Add the <!DOCTYPE html> declaration.
3. Inside <body>, create an <audio> tag with the controls attribute.
4. Add multiple <source> tags inside <audio> to support various formats (.mp3, .ogg).
5. Create a <video> tag with the controls attribute and optional poster image.

6. Add multiple <source> tags for different formats (.mp4, .webm, .ogg).
7. Save and open the file in a browser to test playback.

---

6. Program Code

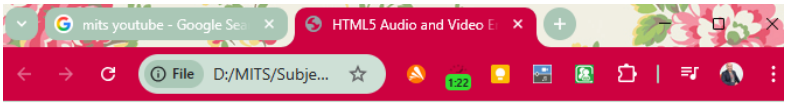
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>HTML5 Audio and Video Embedding</title>
  <style>
    body { font-family: Arial, sans-serif; text-align: center; padding: 20px; }
    h1 { color: #4CAF50; }
    audio, video { margin: 20px auto; display: block; }
  </style>
</head>
<body>

  <h1>HTML5 Multimedia Embedding</h1>

  <h2>Audio Example</h2>
  <audio controls>
    <source src="sample-audio.mp3" type="audio/mpeg">
    <source src="sample-audio.ogg" type="audio/ogg">
    Your browser does not support the audio element.
  </audio>

  <h2>Video Example</h2>
  <video width="480" height="270" controls poster="poster-image.jpg">
    <source src="sample-video.mp4" type="video/mp4">
    <source src="sample-video.webm" type="video/webm">
    <source src="sample-video.ogg" type="video/ogg">
    Your browser does not support the video element.
  </video>

</body>
</html>
```

---

7. Output

When the page is opened in a browser:

- An **audio player** appears with play/pause, volume, and seek controls.
- A **video player** appears with a poster image until played, showing playback controls.



HTML5 Multimedia Embedding

Audio Example

0:00 / 0:00

Video Example

8. Result

The webpage was successfully created to embed audio and video using HTML5 <audio> and <video> tags with multiple source formats for compatibility.

9. Viva Questions

1. What is the purpose of <source> tags in <audio> and <video>?
2. How does poster attribute work in <video>?
3. Name three common video formats supported in HTML5.
4. What is the use of preload in <audio>?
5. Why is it recommended to provide multiple file formats for multimedia?

Unit II: HTML5 and CSS Styling

Experiment No.: 9

Title: Apply all three types of CSS to style HTML elements

---

1. Aim

To demonstrate the use of all three types of CSS—Inline, Internal, and External—to style HTML elements.

---

2. Objectives

- To understand the differences between Inline, Internal, and External CSS.
- To learn how to link CSS files to HTML pages.
- To apply styles in a structured manner for better maintainability.

---

3. Software & Hardware Requirements

- **Software:** Text Editor (VS Code, Sublime Text, Notepad) and a modern web browser (Chrome, Firefox, Edge).
- **Hardware:** Desktop or Laptop with basic specifications.

---

4. Theory

CSS (Cascading Style Sheets) is used to control the look and feel of HTML elements.

Types of CSS

1. **Inline CSS**
   - Applied directly within an HTML element using the style attribute.
   - Example:

html
```
<p style="color: blue;">This is blue text</p>
```

2. **Internal CSS**
   - Defined inside a <style> tag within the <head> section of an HTML document.
   - Example:

html
```
<style>
  p { color: red; }
</style>
```

3. **External CSS**
   - Written in a separate .css file and linked using the <link> tag.
   - Example:

html
```
<link rel="stylesheet" href="styles.css">
```

Advantages of External CSS:

- Reusability across multiple pages.
- Easier maintenance and cleaner HTML code.

---

5. Procedure

1. Create an HTML file named css_demo.html.
2. Add **Inline CSS** for one element.
3. Add **Internal CSS** in the <head> section to style multiple elements.
4. Create an **External CSS** file named styles.css and link it using <link>.
5. Save and open in a browser to verify styles.

6. Program Code

HTML File: css_demo.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>CSS Types Example</title>

   <!-- Internal CSS -->
   <style>
      h1 {
         color: green;
         text-align: center;
      }
      p {
         font-size: 18px;
         font-family: Arial, sans-serif;
      }
   </style>

   <!-- External CSS -->
   <link rel="stylesheet" href="styles.css">
</head>
<body>

   <!-- Inline CSS -->
   <h1 style="background-color: yellow;">Welcome to CSS Styling</h1>

   <p>This paragraph is styled with internal CSS.</p>

   <div class="external-style">
      This section is styled using external CSS.
   </div>

</body>
</html>
```

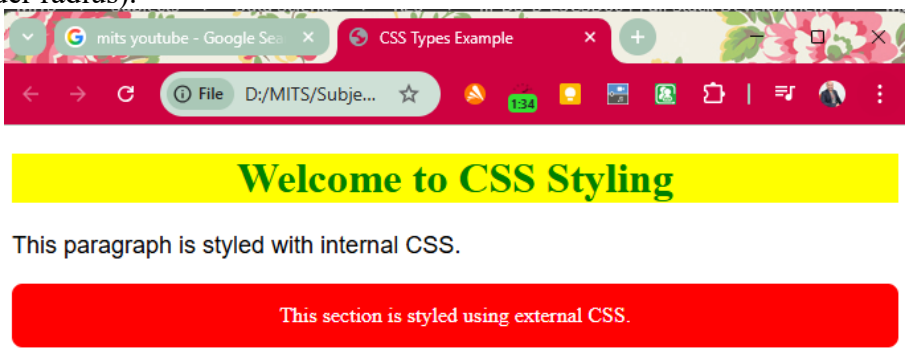
External CSS File: styles.css

```
.external-style {
   color: white;
   background-color: navy;
   padding: 15px;
   border-radius: 8px;
   text-align: center;
}
```

7. Output

- **Heading (<h1>)** has a yellow background via Inline CSS and green text from Internal CSS.
- **Paragraph (<p>)** is styled using Internal CSS (font size and family).

- **Div ( )** is styled using External CSS (white text, navy background, padding, border radius).



8. Result

A webpage was successfully created to apply Inline, Internal, and External CSS styles to HTML elements, demonstrating their differences and usage.

9. Viva Questions

1. What are the main differences between Inline, Internal, and External CSS?
2. Which type of CSS is recommended for large websites and why?
3. What is the purpose of the <link> tag in HTML?
4. Can we use all three CSS types together in one page? Why or why not?
5. What is CSS specificity and how does it affect style priority?

Unit II: HTML5 and CSS Styling

Experiment No.: 10

Title: Demonstrate various selector types (simple, combinators, pseudo-classes/elements, attribute selectors) in a single web page

---

1. Aim

To create a single HTML page that demonstrates the use of CSS selectors, including simple selectors, combinators, pseudo-classes, pseudo-elements, and attribute selectors.

---

2. Objectives

- To understand the different types of CSS selectors.
- To learn how to style HTML elements using selectors efficiently.
- To demonstrate practical examples of each selector type.

---

3. Software & Hardware Requirements

- **Software:** Text Editor (VS Code, Sublime Text, Notepad) and modern web browser (Chrome, Firefox, Edge).
- **Hardware:** Desktop or Laptop with basic specifications.

---

4. Theory

CSS selectors define the pattern to select HTML elements for styling.

Types of CSS Selectors

1. **Simple Selectors** – Select elements based on name, id, or class.
    - Example: p, .className, #idName
2. **Combinator Selectors** – Define a relationship between elements.
    - **Descendant selector (space)** – div p
    - **Child selector (>)** – ul > li
    - **Adjacent sibling selector (+)** – h1 + p
    - **General sibling selector (~)** – h1 ~ p
3. **Pseudo-classes** – Style elements in a specific state.
    - Example: a:hover, p:first-child, input:focus
4. **Pseudo-elements** – Style specific parts of an element.
    - Example: p::first-letter, p::after
5. **Attribute Selectors** – Select elements based on attributes and values.
    - Example: input[type="text"], a[target="_blank"]

---

5. Procedure

1. Create an HTML file named selectors_demo.html.
2. Include various HTML elements such as headings, paragraphs, links, lists, and form inputs.
3. Write CSS code to demonstrate each selector type.
4. Save the file and open it in a browser to verify styling.

---

6. Program Code

HTML File: selectors_demo.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Selector Demonstration</title>
```

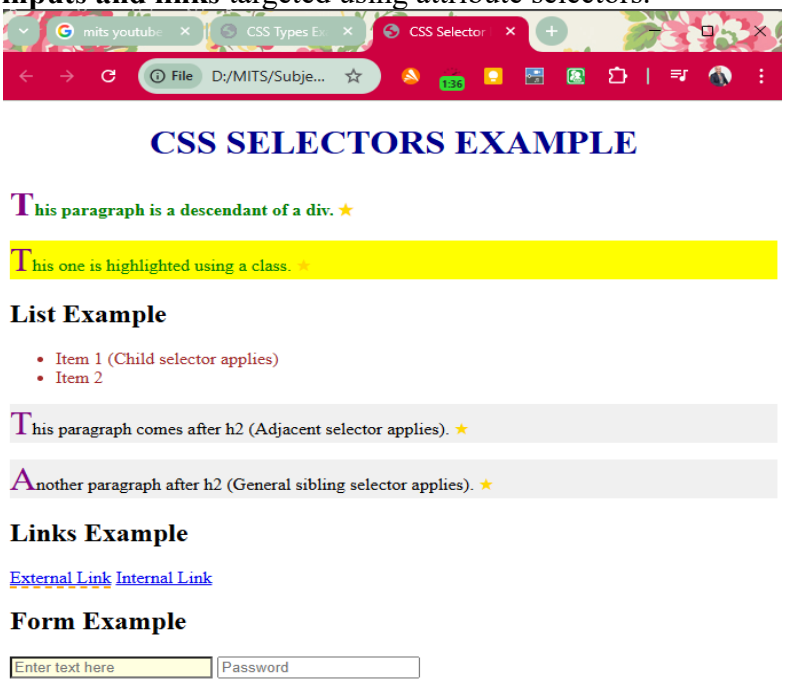
```
    <style>
        /* Simple Selectors */
        h1 {
            color: darkblue;
            text-align: center;
        }
        .highlight {
            background-color: yellow;
        }
        #main-title {
            text-transform: uppercase;
        }

        /* Combinator Selectors */
        div p { color: green; } /* Descendant */
        ul > li { color: brown; } /* Child */
        h2 + p { font-style: italic; } /* Adjacent sibling */
        h2 ~ p { background-color: #f0f0f0; } /* General sibling */

        /* Pseudo-classes */
        a:hover { color: red; }
        p:first-child { font-weight: bold; }
        input:focus { border: 2px solid blue; }

        /* Pseudo-elements */
        p::first-letter { font-size: 200%; color: purple; }
        p::after { content: " ★"; color: gold; }

        /* Attribute Selectors */
        input[type="text"] { background-color: lightyellow; }
        a[target="_blank"] { border-bottom: 2px dashed orange; }
    </style>
</head>
<body>

    <h1 id="main-title">CSS Selectors Example</h1>

    <div>
        <p>This paragraph is a descendant of a div.</p>
        <p class="highlight">This one is highlighted using a class.</p>
    </div>

    <h2>List Example</h2>
    <ul>
        <li>Item 1 (Child selector applies)</li>
        <li>Item 2</li>
    </ul>

    <p>This paragraph comes after h2 (Adjacent selector applies).</p>
    <p>Another paragraph after h2 (General sibling selector applies).</p>

    <h2>Links Example</h2>
```

```
    <a href="https://example.com" target="_blank">External Link</a>
    <a href="page.html">Internal Link</a>

    <h2>Form Example</h2>
    <form>
        <input type="text" placeholder="Enter text here">
        <input type="password" placeholder="Password">
    </form>

</body>
</html>
```

---

7. Output

- **Headings** styled using simple selectors (element, .class, #id).
- **Lists, paragraphs, and siblings** styled using combinator selectors.
- **Links, form inputs** respond to pseudo-classes like :hover and :focus.
- **First letters and symbols** added using pseudo-elements.
- **Specific inputs and links** targeted using attribute selectors.



---

8. Result

A web page was successfully created to demonstrate the usage of various CSS selectors—simple, combinators, pseudo-classes, pseudo-elements, and attribute selectors—in a single HTML file.

---

9. Viva Questions

1. Differentiate between pseudo-classes and pseudo-elements.
2. What is the difference between a child selector (>) and a descendant selector (space)?
3. How does the :hover pseudo-class work?
4. Give examples of when attribute selectors are useful.
5. Can combinator selectors be chained together? Give an example.