Birla Vishvakarma Mahavidyalaya (BVM) Engineering College

An Autonomous Institution

Computer Workshop – II (103SB)

(LTP = 0-0-4)

Unit 1: HTML&XML

❖ Concept of WWW (World Wide Web)

The World Wide Web (WWW) is a system of interlinked hypertext documents and multimedia content accessible via the Internet. Using web browsers, users can access web pages containing text, images, videos, and other content through hyperlinks.

Key Features of WWW:

- **Hypertext:** Links between web pages, allowing navigation across content.
- **Multimedia:** Supports text, images, videos, and audio.
- **Interactivity:** Enables user interaction via forms, links, and dynamic content.

❖ Internet and WWW

The **Internet** is the global network of interconnected computers enabling communication and data sharing. The **WWW** is a service that operates over the Internet, providing access to web resources via protocols like HTTP.

Aspect	Internet	WWW
Definition	Network infrastructure for data exchange	Service using the Internet to access web content
Protocols Used	TCP/IP, FTP, SMTP, etc.	HTTP, HTTPS
Scope	Includes WWW, email, and more	Subset of the Internet

❖ HTTP Protocol

The **Hypertext Transfer Protocol (HTTP)** is the foundation of communication on the WWW. It defines how web browsers (clients) request resources from servers and how servers respond.

Key Features:

- **Request-Response Model:** Clients send requests to servers, which respond with the requested resource.
- **Stateless Protocol:** Each request is independent, without memory of previous interactions.
- **HTTP Methods:** Common methods include:
  - **GET:** Retrieve data.
  - **POST:** Submit data.
  - **PUT:** Update data.
  - **DELETE:** Remove data.

❖ What is HTML?

- HTML stands for Hyper Text Markup Language
- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

➢ A Simple HTML Document/Basic Structure of HTML

```
<!DOCTYPE html>
<html>
<head>
    <title>Basic HTML Structure</title>
</head>
<body>
    <h1>Welcome to HTML</h1>
    <p>This is a simple web page.</p>
</body>
</html>
```

- The **<!DOCTYPE html>** declaration defines that this document is an HTML5 document
- The **<html>** element is the root element of an HTML page
- The **<head>** element contains meta information about the HTML page
- The **<title>** element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab)

- The **<body>** element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.
- The **<h1>** element defines a large heading
- The **<p>** element defines a paragraph

Exercise 1 :- Structuring Content with HTML

Problem Statement: Create a basic HTML structure with the following:

- A title in the browser tab.
- A heading saying "**Welcome to Web Development**."
- A paragraph introducing the page purpose.

❖ HTML Element

HTML elements are the building blocks of web pages. An element consists of a start tag, content, and an end tag. Some elements are self-closing and don't have an end tag.

Structure of an HTML Element

```
<tagname attribute="value">Content</tagname>
```

Example:-

```
<p>This is a paragraph.</p>
```

➢ Types of HTML Elements: -

1. **Block-Level Elements**
   - These elements take up the full width of the parent container.
   - Always start on a new line.

   Examples:
   - ** :** Generic container for grouping elements
   - **<p>:** Paragraph of text
   - **<h1>** to **<h6>:** Headings (from largest to smallest)
   - **<ul>** and **<ol>:** Lists
   - <**table**>**:** Tables

```
<div>
    <h1>Main Heading</h1>
    <p>This is a block element.</p>
</div>
```

2. **Inline Elements**
   - These elements occupy only as much width as necessary.
   - Do not start on a new line.

Examples:

- **:** Generic inline container for styling
- **<a>:** Hyperlinks
- **<b>, <i>, , :** Text formatting
- **<img>:** Images

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

3. **Empty (Self-Closing) Elements**
   - Do not have content or a closing tag.
   - Example**: <img>,
, <hr>, <input>.**

```
<img src="image.jpg" alt="Example Image">
<br>
```

➢ **HTML Element Categories**

1. **Structural Elements**

- Define the structure of a webpage.
- Examples:
  - **<html>:** Root element
  - **<head>:** Metadata
  - **<body>:** Main content area

```
<html>
    <head>
        <title>Page Title</title>
```

```
    </head>
    <body>
        <h1>Page Heading</h1>
    </body>
</html>
```

2. **Text Content Elements**

   - Define and format text.
   - Examples: <p>, <h1>, <b>, <i>.

3. **Image and Multimedia Elements**

   - Handle images, audio, and video.
   - Examples: <img>, <audio>, <video>.

4. **Form Elements**

   - Create interactive forms.
   - Examples: <form>, <input>, <textarea>, <button>.

```
<form>
    <input type="text" placeholder="Enter Name">
    <button type="submit">Submit</button>
</form>
```

5. **List Elements**

   - Organize items in ordered or unordered lists.
   - Examples: <ul>, <ol>, <li>.

6. **Table Elements**

   - Create tables for data.
   - Examples: <table>, <tr>, <td>, <th>.

```
<table border="1">
    <tr>
        <th>Name</th>
        <th>Age</th>
    </tr>
    <tr>
        <td>Alice</td>
        <td>25</td>
```

```
    </tr>
</table>
```

7. **Semantic Elements (HTML5)**

    - Convey meaning and structure to browsers and developers.
    - Examples: <header>, <footer>, <section>, <article>, <nav>.

```
<header>
    <h1>Website Header</h1>
</header>
<article>
    <h2>Article Title</h2>
    <p>This is an article.</p>
</article>
```

8. **Interactive Elements**

    - Create user interaction.
    - Examples: <button>, <details>, <dialog>.

❖ Attributes in HTML Elements

Attributes provide additional information about an element. They are written within the start tag.

Common Attributes:

- **id:** Unique identifier for an element
- **class:** Class name for grouping elements
- **style:** Inline CSS styles
- **src:** Source for images and multimedia
- **href:** URL for hyperlinks

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

❖ Nested Elements

HTML allows elements to be nested inside other elements.

```
<div>
    <h1>Main Heading</h1>
    <p>This is a paragraph with a <a href="#">link</a>.</p>
</div>
```

❖ Text Formatting and Fonts

HTML offers tags to style and format text.

Formatting Tags:

- **Bold:** <b> or
- **Italic:** <i> or
- **Underline:** <u>
- **Strikethrough:** <s>

```
<p><b>Bold</b>, <i>Italic</i>, and <u>Underlined</u> text.</p>
<p style="font-family: Arial; font-size: 20px;">Custom Font
Example</p>
```

❖ Commenting Code

Use comments to annotate code for better understanding.

```
<!-- This is a comment -->
<p>Visible content is here.</p>
```

❖ Working with Colors

Add color using CSS styles or attributes.

- **Named Colors:** red, blue
- **Hex Codes:** #FF0000 (Red)
- **RGB:** rgb(255, 0, 0)

```
<p style="color: red;">This is red text.</p>
<p style="background-color: #00FF00;">Green Background</p>
```

❖ Hyperlinks

Hyperlinks connect web pages and external resources.

```
<a href="https://www.google.com" target="_blank">Visit Google</a>
<a href="#section1">Jump to Section 1</a>
```

Exercise 2:- Formatting Text, Adding Hyperlinks, and Applying Colors

Problem Statement: Create a webpage with the following features:

- A paragraph demonstrating text formatting using bold, italics, and strikethrough.
- A hyperlink to a favorite website that opens in a new tab.
- A section titled "Contact Me" with a hyperlink to jump to it from the top of the page.
- Use colors to style the headings and text.

❖ Lists

Lists display content in an organized way.

Unordered List:

```
<ul>
    <li>HTML</li>
    <li>CSS</li>
    <li>JavaScript</li>
</ul>
```

Ordered List:

```
<ol>
    <li>Introduction</li>
    <li>Basics</li>
    <li>Advanced</li>
</ol>
```

Nested Lists:

```
<ul>
    <li>Frontend
        <ul>
            <li>HTML</li>
            <li>CSS</li>
        </ul>
    </li>
</ul>
```

Exercise 3:- Using Lists in HTML

Problem Statement: Create a webpage displaying your three favorite hobbies using:

- An unordered list for hobbies.
- A nested ordered list for sub-details of one hobby.

❖ Tables

Tables organize data into rows and columns.

```
<table border="1">
    <tr>
        <th>Subject</th>
        <th>Marks</th>
    </tr>
    <tr>
        <td>Math</td>
        <td>95</td>
    </tr>
</table>
```

HTML Table Tags:

Tag	Description
<table>	Defines a table
<th>	Defines a header cell in a table
<tr>	Defines a row in a table
<td>	Defines a cell in a table
<caption>	Defines a table caption
<colgroup>	Specifies a group of one or more columns in a table for formatting
<col>	Specifies column properties for each column within a <colgroup> element
<thead>	Groups the header content in a table
<tbody>	Groups the body content in a table
<tfoot>	Groups the footer content in a table

Exercise 4:- Creating a Table

Problem Statement:
Create a table showing a list of students with columns for Name and Age. Add a header row and border.

❖ Images in HTML

Images enhance the visual content of a webpage and are added using the <img> tag.

Attributes of <img> Tag:

- **src:** Specifies the image source.
- **alt:** Provides alternative text for accessibility.
- **width and height**: Define dimensions.
- **title:** Displays a tooltip when hovered over.

```
<img src="example.jpg" alt="Sample Image" width="400" height="300"
title="Image Example">
```

Exercise 5 :- Embedding an Image

Problem Statement: Add an image to your webpage:

- Use the alt attribute for accessibility.
- Set a width of 300px and height of 200px.

Practical 1: Creating a Personal Profile Web Page

Problem Statement:

Create a simple web page that serves as a personal profile. This page should introduce you, display some basic information about yourself, and list your hobbies and interests. Use various HTML tags for headings, paragraph alignment, line breaks, horizontal rules, and text formatting to enhance the layout.

Objectives:

1. To understand the structure of an HTML document.
2. To learn the use of different HTML tags for headings, paragraphs, formatting, and alignment.
3. To create a visually appealing and organized personal profile web page.

Theory:

HTML (HyperText Markup Language) is the standard language for creating web pages. It provides a structure to display text, images, and other elements in a browser. Below are the key elements used in this practical:

1. **HTML Structure**
    - The <!DOCTYPE html> declaration defines the document type as HTML5.
    - <html>: The root element of an HTML page.
    - <head>: Contains metadata and the <title> tag.
    - <body>: Contains the content displayed on the web page.

2. **Headings (<h1> to <h6>)**
    - Used for titles and section headers, ranging from largest (<h1>) to smallest (<h6>).

3. **Paragraphs and Alignment**
    - <p>: Defines paragraphs.
    - align attribute: Aligns paragraph text (e.g., align="center").

4. **Line Breaks and Horizontal Rules**
    -
: Inserts a line break.
    - <hr>: Creates a horizontal rule for section separation.

5. **Text Formatting Tags**
    - <b> and <i>: Bold and italicize text.
    - <sup> and <sub>: Superscript and subscript text.

- <ins> and <del>: Show added and removed text.
- : Emphasizes text for readability.

6. **Lists**
    - : Creates an unordered (bulleted) list.
    - : Defines individual list items.

7. **Attributes for Styling**
    - bgcolor: Sets background color (e.g., bgcolor="#e0f7fa").
    - text: Sets the text color (e.g., text="#37474f").

Requirements:

- **Hardware**: Need 20 Windows operating machines.
- **Software**:
    - A text editor or IDE (e.g., **VS Code**) to write code.
    - A web browser (e.g., Chrome, Firefox,Edge) to render the web page.

Procedure:

1. Use **VS Code** to write your HTML code.
2. **Start with the Basic HTML Structure**:

- Begin the file with <!DOCTYPE html> to declare the document as HTML5.
- Add the <html> tag to define the HTML document.
- Inside the <head> section, include a <title> tag to set the title of your webpage.

3. **Set Background and Text Colors**:

- In the <body> tag, use attributes like bgcolor and text to set the background and text colors of the page.

4. **Add a Title and Introduction**:

- Use the <h1> tag to display your name as the title of the page.
- Add a subtitle using <h3> to introduce yourself.
- Write a short introduction paragraph about yourself using the <p> tag. Align the paragraph to the center with the align attribute.

5. **Include a Horizontal Line**:

- Insert a horizontal rule using <hr> to visually separate sections.

6. **Create an About Section**:

- Use the <h4> tag to add a heading for the "About Me" section.
- Write a descriptive paragraph using the <p> tag.
- Apply formatting tags such as <b> for bold, <i> for italics, <ins> for underline, and <del> for strikethrough text.

7. **List Your Hobbies and Interests**:

- Use the <h5> tag to create a heading for the "Hobbies and Interests" section.
- Create an unordered list (<ul>) to list your hobbies using <li> tags for each item.
- Enhance the list items with additional tags like <sup> for superscript and <small> for smaller text.

8. **Add Additional Notes**:

- Use the <p> tag to add more information or interests.
- Use to emphasize text within the paragraph.

9. **Save Your File**:

- Save the file with a **.html** extension, such as "**personal_profile.html**".

10. **Test the Web Page**:

- Open the saved HTML file in a web browser to view your personal profile page.

11. **Modify as Needed**:

- Make any adjustments or enhancements to the layout, colors, or text based on your preference.

Implementation and Result:

1. Students are required to implement this practical on their own using the above procedure.
2. After successful execution of the HTML file, students must:
    - Manually write the source code in their record book.
    - Attach a digital copy of the output or screenshot to their record book.

3. Save the HTML files and results in the college server at the following location:
    - Server Path: **10.10.30.20**
    - Folder: **New Anonymous Share > 103SB CW-2 > Division Folder (e.g., D12 or D03) > Batch Folder**
    - Create a folder inside the batch folder with your **Enrollment ID** (e.g., 24CP01).
    - Inside your folder, create a subfolder for this practical named "**Practical_1**".
    - Save the HTML file and result screenshot in this folder.

Conclusion:

Students must write the conclusion in their record book themselves based on their experience and understanding of the practical.

Practical 2: Creating a Travel Itinerary Web Page

Problem Statement:

Create a web page to display a travel itinerary for a planned three-day trip. The page should organize the itinerary details in a table format, listing each day's planned activities, timings, and locations. Enhance the layout by adding images of the travel destinations and hyperlinks to additional information about each location. Use various HTML formatting tags to improve readability and style.

Objectives:

1. To create and format tables in HTML.
2. To use images and hyperlinks effectively for enhancing web page interactivity.
3. To organize information clearly using headings and text formatting.
4. To improve visual presentation using background colors and text attributes.

Theory:

HTML provides a variety of elements to create structured and visually appealing content. For this practical, the key elements are:

1. **Tables**:
    - <table>: Defines the table structure.
    - <tr>, <th>, <td>: Create table rows, headers, and cells respectively.
    - Attributes like border, cellpadding, and cellspacing enhance the table's appearance.
2. **Images**:
    - <img src="..." height="..." width="..." alt="...">: Displays images with specified dimensions and alternate text.
3. **Hyperlinks**:
    - <a href="URL" target="_blank">: Adds clickable links that open in a new tab.
4. **Lists**:
    - and <li>: Create unordered lists for additional content such as packing lists.
5. **Text Formatting**:
    - <b>, <i>: Highlight important text.
    - <sup>, <small>: Add superscripts and smaller text details.
6. **Body Tag Attributes**:
    - bgcolor: Sets the background color.
    - text, link, and vlink: Define text and link colors for better readability.

Requirements:

- **Hardware**: Need 20 Windows operating machines.
- **Software**:
    - A text editor or IDE (e.g., VS Code) to write code.
    - A web browser (e.g., Chrome, Firefox,Edge) to render the web page.
    - Image files of travel destinations (stored locally or online).

Procedure:

1. **Set Up the HTML File**:

    - Open a **VS code** create a new file.
    - Begin with the <!DOCTYPE html> declaration to define the document as HTML5.

2. **Define the Basic Structure**:

    - Add <html> and <head> tags. Inside <head>, include a <title> tag for the browser tab title.
    - Inside the <body> tag, set the page's background color, text color, and link colors using bgcolor, text, link, and vlink attributes.

3. **Add the Page Title and Introduction**:

    - Use <h2> for the main heading to introduce the travel itinerary.
    - Include a <p> tag to briefly explain the purpose of the page and provide instructions for clicking on location names to access more information.

4. **Create the Travel Itinerary Table**:

    - Use <h3> for the heading of the itinerary section.
    - Add a <table> tag with attributes for border, cellpadding, cellspacing, and alignment using align.
    - Create the header row with <tr> and <th> tags to label columns (Day, Time, Activity, Location).
    - Add rows for each day using <tr> and <td> tags. Alternate the row colors using the bgcolor attribute for better readability.
    - Use <a> tags within the Location column to provide hyperlinks to detailed information about each destination. Set target="_blank" to open the links in a new tab.

5. **Insert Photos of Destinations**:

- Use <h4> for the heading of the photos section.
- Include <img> tags for each destination, specifying the src for the image file path, alt text for accessibility, and height and width attributes for size.

6. **Add a Packing List**:

- Use <h5> for the heading of the packing list section.
- Add an unordered list (<ul>) with type="square" to display items to pack for the trip.
- Use <sup> and <small> tags to add extra details like measurements or optional items.

7. **Save and Test**:

- Save the file with a **.html** extension.
- Open it in a web browser to ensure the table, images, and hyperlinks are displayed correctly.

Implementation and Result:

1. Students are required to implement this practical on their own using the above procedure.
2. After successful execution of the HTML file, students must:
    - Manually write the source code in their record book.
    - Attach a digital copy of the output or screenshot to their record book.

3. Save the HTML files and results in the college server at the following location:

    - Server Path: **10.10.30.20**
    - Folder: **New Anonymous Share > 103SB CW-2 > Division Folder (e.g., D12 or D03) > Batch Folder**
    - Create a folder inside the batch folder with your **Enrollment ID** (e.g., 24CP01).
    - Inside your folder, create a subfolder for this practical named "**Practical_3**".
    - Save the HTML file and result screenshot in this folder.

Conclusion:

Students must write the conclusion in their record book themselves based on their experience and understanding of the practical.

Practical 3: Creating a Web Page with an Irregular Table Layout

Problem Statement:

Create a web page that presents a weekly class schedule using an irregular table format. The table should include class days, timings, subjects, and rooms, but some cells should span multiple rows or columns to reflect periods that cover multiple time blocks or days. Apply various HTML table tags and attributes, such as **border, colspan, rowspan, cellpadding, cellspacing, width,** and **height,** to create a well-structured, visually appealing table.

Objectives:

1. To understand the use of HTML table attributes for creating irregular table layouts.
2. To practice applying colspan and rowspan attributes to merge table cells.
3. To enhance the presentation and readability of table content with styling attributes.
4. To learn how to structure a web page with headings, background colors, and alignment.

Theory:

HTML tables are essential for organizing data in a grid format. An irregular table layout uses merged cells and varying column or row spans to present complex information effectively. Below are the key concepts used in this practical:

1. **HTML Table Structure**:
    - <table>: Defines the table container.
    - <tr>: Represents a table row.
    - <th>: Creates a header cell.
    - <td>: Creates a standard data cell.
2. **Attributes for Table Design**:
    - colspan: Merges multiple columns into a single cell.
    - rowspan: Merges multiple rows into a single cell.
    - border: Adds borders to the table.
    - cellpadding and cellspacing: Define spacing within and between cells.
    - width and height: Specify the dimensions of the table or cells.
3. **Styling Attributes**:
    - bgcolor: Sets background color for table cells or rows.
    - text: Specifies text color.
    - Alternating row colors improve readability and aesthetics.
4. **HTML Page Structure**:
    - <!DOCTYPE html>: Declares the HTML5 document type.

- <html>, <head>, and <body>: Provide the main structure of the page.
- <title>: Sets the title displayed in the browser tab.

Requirements:

- **Hardware**: Need 20 Windows operating machines.
- **Software**:
  - A text editor or IDE (e.g., VS Code) to write code.
  - A web browser (e.g., Chrome, Firefox,Edge) to render the web page.

Procedure:

1. **Set Up the HTML File**:

   - Open a **VS Code** create a new file.
   - Start with the <!DOCTYPE html> declaration for HTML5.
   - Add <html> and <head> tags, and include a <title> tag inside <head> for the browser tab title.

2. **Design the Page Layout**:

   - Use the <body> tag to define the page's background color and text color using bgcolor and text attributes.
   - Add a <h2> tag for the main title of the page and a <p> tag to provide an introduction or explanation of the schedule.

3. **Create the Table Structure**:

   - Add a <table> tag with attributes for border, cellpadding, cellspacing, and width to style the table.
   - Use <tr> for rows, <th> for headers, and <td> for data cells. Align the table using align="center" for better aesthetics.

4. **Add Table Headers**:

   - Create a header row with <th> tags for columns like Day and Time Slots (e.g., 9:00 - 10:00 AM, 10:00 - 11:00 AM).

5. **Design Rows with Cell Spanning**:

   - Use the rowspan attribute for subjects that span multiple rows (e.g., Math, Physics Lab).
   - Use the colspan attribute for subjects or activities that span multiple columns (e.g., Math Workshop, Library Session).
   - Alternate row background colors using bgcolor for visual clarity.

6. **Fill in Data**:

- Add specific subjects, timings, and breaks for each day using <td> tags.
- Ensure proper alignment and structure for irregular cells by combining rowspan and colspan as required.

7. **Finalize the Design**:

- Add a horizontal rule (<hr>) above and below the table to separate it from other content.
- Save the file with a .html extension.

8. **Test the Page**:

- Save the file with a **.html** extension and open it in a web browser.
- Open the file in a web browser to ensure the table layout matches the desired weekly class schedule.
- Verify the alignment, colors, and spanning of cells for accuracy.

Implementation and Result:

1. Students are required to implement this practical on their own using the above procedure.
2. After successful execution of the HTML file, students must:
    - Manually write the source code in their record book.
    - Attach a digital copy of the output or screenshot to their record book.

3. Save the HTML files and results in the college server at the following location:

    - Server Path: **10.10.30.20**
    - Folder: **New Anonymous Share > 103SB CW-2 > Division Folder (e.g., D12 or D03) > Batch Folder**
    - Create a folder inside the batch folder with your **Enrollment ID** (e.g., 24CP01).
    - Inside your folder, create a subfolder for this practical named "**Practical_4**".
    - Save the HTML file and result screenshot in this folder.

Conclusion:

Students must write the conclusion in their record book themselves based on their experience and understanding of the practical.

❖ Forms in HTML

Forms collect user input and send it to a server for processing.

Key Form Tags and Attributes:

- **<form>:** Main container for a form. Attributes:
    - action: URL where form data is sent.
    - method: HTTP method (GET or POST).
- **<input>:** Creates input fields. Types include text, password, email, radio, checkbox, submit, etc…
- **<textarea>:** Multiline text input
- **<select> and <option>:** Dropdown menus.
- **<button>:** Button for actions

➢ Common HTML Form Elements

1. **Text Input Fields:** Used to collect single-line text inputs.

```
<input type="text" name="username" placeholder="Enter your
name">
```

2. **Password Input:** Masks the input characters for security.

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

3. **Radio Buttons:** Allows the user to select one option from a group.

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

4. **Checkboxes:** Allows the user to select one or more options.

```
<input type="checkbox" name="hobby" value="reading"> Reading
<input type="checkbox" name="hobby" value="traveling">
Traveling
```

5. **Drop-Down Menu:** Allows the user to select one option from a dropdown list.

```
<select name="country">
    <option value="india">India</option>
    <option value="usa">USA</option>
</select>
```

6. **File Upload:** Allows users to upload files.

```
<input type="file" name="profile_picture">
```

7. **Text Area:** Collects multi-line text input.

```
<textarea name="comments" rows="5" cols="30"></textarea>
```

8. **Submit Button:** Submits the form data to the server.

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

9. **Reset Button:** Resets all fields to their default values.

```
<input type="reset" value="Reset">
```

➢ Exercise 6: Adding Forms

Problem Statement: Create a form that collects a user's name ,email address and gender. Add a submit and reset button.

Practical 4: Creating a Registration Form Web Page

Problem Statement:

Create a web page that serves as a registration form for a fictional event. The form should collect various user details, such as name, email, password, contact preferences, and file uploads (e.g., profile picture). Include an embedded frame (**iframe**) for event rules, and use tags such as **<font>, <frame>, <noframes>, <iframe>,** and form elements like **<input>, <textarea>, <select>, <option>, <fieldset>, and <legend>**. Also, demonstrate how to link a mail message.

Objectives:

1. To learn how to create and structure a registration form in HTML.
2. To understand the use of form elements like <input>, <textarea>, <select>, <fieldset>, and <legend>.
3. To embed external content using <iframe>.
4. To demonstrate interactive features like submit/reset buttons, dropdowns, and checkboxes.

Theory:

Forms in HTML are used to collect user input. They consist of various elements and attributes that define their structure and interactivity. Below are the essential concepts:

1. **Form Elements**:

    - <form>: Defines a form for user input.
    - <input>: Used for various input types (text, password, file, etc.).
    - <textarea>: Creates a multi-line input field.
    - <select> and <option>: Create dropdown menus.
    - <fieldset> and <legend>: Group and label related form elements.

2. **Styling and Organization**:

    - Attributes like bgcolor and text enhance the page's visual appeal.
    - <label> ensures accessibility by associating labels with input fields.

3. **Embedding and Linking**:

    - <iframe>: Embeds external content (e.g., event rules).
    - <a href="mailto:info@example.com">: Creates an email link.

Requirements:

- **Hardware**: Need 20 Windows operating machines.
- **Software**:
    - A text editor or IDE (e.g., VS Code) to write code.
    - A web browser (e.g., Chrome, Firefox,Edge) to render the web page.

Procedure:

1. **Set Up the HTML File**:
    - Open **VS Code** and create a new file.
    - Start with the <!DOCTYPE html> declaration to define the document as HTML5.
    - Add the <html> and <head> tags, and include a <title> tag inside <head> to set the browser tab title.
2. **Design the Page Layout**:
    - Use the <body> tag to set the page's bgcolor (background color), text (text color), and link (link color).
    - Add a <h1> tag for the main page title and a <p> tag to introduce the registration form with a mail link (mailto) for inquiries.
3. **Create the Registration Form**:
    - Use the <form> tag to define the form structure, and set the action attribute to specify where the form data should be submitted.
    - Choose the method="POST" attribute for secure data submission.
4. **Add Form Fields**:
    - Use <fieldset> and <legend> to group related form fields for better organization.
    - Include the following input types:
        - **Text Input**: <input type="text"> for name and email fields.
        - **Password**: <input type="password"> for the password field.
        - **File Upload**: <input type="file"> for profile picture upload.
        - **Radio Buttons**: <input type="radio"> for gender selection.
        - **Checkboxes**: <input type="checkbox"> for contact preferences.
        - **Dropdown Menu**: <select> and <option> for session preferences.
        - **Text Area**: <textarea> for additional comments or information.

5. **Add Submission and Reset Buttons**:

- Use <input type="submit"> to add a button for form submission.
- Use <input type="reset"> to reset the form fields to their default values.

6. **Embed Event Rules Using Iframe**:

- Use the <iframe> tag to embed a separate HTML file (rules.html) containing the event rules.
- Set the src attribute of the iframe to the file path of the event rules and define its width and height.
- Provide an alternate text link (<a>) to the rules file for users unable to view the iframe.

7. **Handle Unsupported Frames**:

- Use the <noframes> tag to display a fallback message for browsers that do not support frames.

8. **Test the Web Page**:

- Save the file with a **.html** extension and open it in a web browser.
- Check the functionality of form fields, buttons, iframe, and links.
- Ensure the page layout is visually appealing and all elements are aligned properly.

Important Notes

The registration form provided in this exercise is a static front-end form. This means that without a backend to process the form data, students will not be able to submit the form. The action="/submit_form" is simply a placeholder, and for the form to work in a real application, it would need to be linked to a server-side script like PHP, Node.js, or Python.

Additionally, the rules.html file referenced in the <iframe> tag is not included in this exercise. Before using the iframe, students should first create a rules.html file containing the event rules and save it in the same directory as the main HTML file. Once the rules.html file is created, the iframe will properly display the content.

Implementation and Result:

1. Students are required to implement this practical on their own using the above procedure.
2. After successful execution of the HTML file, students must:
    - Manually write the source code in their record book.
    - Attach a digital copy of the output or screenshot to their record book.

3. Save the HTML files and results in the college server at the following location:

    - Server Path: **10.10.30.20**
    - Folder: **New Anonymous Share > 103SB CW-2 > Division Folder (e.g., D12 or D03) > Batch Folder**
    - Create a folder inside the batch folder with your **Enrollment ID** (e.g., 24CP01).
    - Inside your folder, create a subfolder for this practical named "**Practical_5**".
    - Save the HTML file and result screenshot in this folder.

Conclusion:

Students must write the conclusion in their record book themselves based on their experience and understanding of the practical.

❖ XHTML (Extensible HyperText Markup Language)

XHTML is a stricter, XML-based version of HTML, ensuring clean and well-structured documents.

The Most Important Differences from HTML

- <!DOCTYPE> is **mandatory**
- The xmlns attribute in <html> is **mandatory**
- <html>, <head>, <title>, and <body> are **mandatory**
- Elements must always be **properly nested**, **closed** and in **lowercase**
- Attribute values must always be **quoted**
- Attribute minimization is **forbidden**

Example:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>XHTML Example</title>
</head>
<body>
    <h1>Hello XHTML!</h1>
    <img src="example.jpg" alt="Example Image" />
    <p>This is XHTML content.</p>
</body>
</html>
```

❖ Meta Tags

Meta tags provide metadata about a webpage, such as its description, keywords, and author. These are included within the <head> section.

Common Meta Tags:

- **charset:** Defines character encoding
- **description:** Short description of the webpage
- **keywords:** Relevant keywords for SEO
- **author:** Specifies the author
- **viewport:** Responsive design settings

```
<head>
    <meta charset="UTF-8">
    <meta name="description" content="Learn HTML basics and
examples">
    <meta name="keywords" content="HTML, Web Development, Tutorial">
    <meta name="author" content="John Doe">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Meta Tags Example</title>
</head>
```

Example:

❖ HTML5 Additions

HTML5 introduced new semantic elements, multimedia tags, and APIs to enhance functionality.

➢ **Semantic Elements:**

- **<header>:** Header section of a page
- **<footer>:** Footer section of a page
- **<article>:** Independent content
- **<section>:** Groups related content
- **<nav>:** Navigation links

```
<header>
    <h1>Website Title</h1>
</header>
<nav>
    <a href="#home">Home</a>
    <a href="#about">About</a>
</nav>
<section>
    <h2>About Section</h2>
    <p>Details about the site.</p>
</section>
<footer>
    <p>&copy; 2024 Web Development</p>
</footer>
```

➢ **Multimedia Tags:**

- **<audio>:** Embeds audio files.
- **<video>:** Embeds video files.

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

<video width="600" controls>
    <source src="video.mp4" type="video/mp4">
    Your browser does not support the video tag.
</video>
```

➢ **Canvas and Graphics:**

```
<canvas id="myCanvas" width="400" height="300" style="border:1px
solid #000;"></canvas>
<script>
    var canvas = document.getElementById("myCanvas");
    var ctx = canvas.getContext("2d");
    ctx.fillStyle = "#FF0000";
    ctx.fillRect(50, 50, 100, 100);
</script>
```

- **<canvas>:** Drawing graphics via JavaScript.

➢ **Responsive Design with Meta Tags:**

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

Practical 5: Creating a Static Web Page for a Music Player Interface

Problem Statement:

Create a web page that resembles a simple music player interface featuring multiple songs. The page should include a header with the music player's title, a main section that displays a playlist with individual audio controls for each song, and a sidebar with additional information about the music player. Additionally, use the **<canvas>** element to create a visual sound wave effect, and add media controls using the **<audio>** tag.

Objectives:

1. To design a static web page layout using semantic HTML5 tags.
2. To embed audio files with individual controls using the <audio> tag.
3. To use the <canvas> element for creating visual effects.
4. To apply responsive design principles using meta tags.

Theory:

HTML5 provides a rich set of elements to design interactive web pages. Below are the essential concepts:

1. **HTML5 Semantic Tags**:
    - <header>: Defines the page's header content.
    - <article>: Represents the playlist section.
    - <aside>: Contains sidebar information.
    - <footer>: Defines the footer content.
2. **Audio Embedding**:
    - <audio controls>: Embeds an audio player with controls.
    - <source src="..." type="...">: Specifies the audio file format.
3. **Canvas for Visual Effects**:
    - <canvas>: A versatile element for rendering graphics and animations using JavaScript.
4. **Responsive Design**:
    - <meta charset="UTF-8">: Ensures proper encoding for text.
    - <meta name="viewport" content="width=device-width, initial-scale=1.0">: Ensures the page adapts to different screen sizes.

Requirements:

- **Hardware:** Need 20 Windows operating machines.
- **Software:**
    - A text editor or IDE (e.g.,VS Code) to write code.
    - A web browser (e.g., Chrome, Firefox,Edge) to render the web page.
    - JavaScript-enabled browser to test canvas interactivity.
    - Audio files in MP3 format.

Procedure:

1. **Set Up HTML Structure**:

- Start the HTML file with <!DOCTYPE html> to define it as an HTML5 document.
- Include <html>, <head>, and <body> sections.
- In the <head>, set the character encoding to UTF-8 and include a title for the web page. Add a meta viewport tag for responsiveness.

2. **Design Header**:

- Use a <header> tag for the title and a short description of the music player.
- Add styling to make the header visually distinct, such as a background color and centered text.

3. **Main Section Layout**:

- Use the <main> tag to structure the content into two sections:
    - **Sidebar (<aside>)**: Provide information about the music player.
    - **Main Content (<article>)**: Display the playlist with individual audio controls and a canvas for sound visualization.

4. **Add the Playlist**:

- Use <audio> tags for each song in the playlist.
- Inside <audio>, use <source> tags to define the audio file paths and types.
- Include a fallback message for browsers that don't support the <audio> element.

5. **Integrate the Sidebar**:

- Use an <aside> tag to describe the music player's functionality.
- Include styled text that explains how to use the player.

6. **Add a Sound Visualization Canvas**:

- Create a <canvas> element to display a visual sound wave effect.
- Use JavaScript to draw randomized bars on the canvas, mimicking a sound wave visualization.

7. **Footer Section**:

- Use a <footer> tag to add copyright information or additional notes.

8. **Styling with CSS**:

- Use a <style> block within the <head> section to define the appearance of the header, main section, sidebar, playlist, and footer.
- Set display properties to organize content, background-color for aesthetics, and border-radius for rounded edges.

9. **JavaScript for Visualization**:

- Add a <script> block at the end of the body to implement the sound visualization logic.
- Use CanvasRenderingContext2D methods to draw rectangles of varying heights on the canvas.

10. **Test the Page**:

- Save the file with a **.html** extension and open it in a browser.
- Check that the audio controls work, the canvas displays the visual effect, and the layout is as intended.

Implementation and Result:

1. Students are required to implement this practical on their own using the above procedure.
2. After successful execution of the HTML file, students must:
    - Manually write the source code in their record book.
    - Attach a digital copy of the output or screenshot to their record book.

3. Save the HTML files and results in the college server at the following location:

    - Server Path: **10.10.30.20**

- Folder: **New Anonymous Share > 103SB CW-2 > Division Folder (e.g., D12 or D03) > Batch Folder**
- Create a folder inside the batch folder with your **Enrollment ID** (e.g., 24CP01).
- Inside your folder, create a subfolder for this practical named "**Practical_6**".
- Save the HTML file and result screenshot in this folder.

Conclusion:

Students must write the conclusion in their record book themselves based on their experience and understanding of the practical.