Practical No. 1

Aim: Write a code to display the basic tags and various text formatting methods available in HTML.(i.e.<h1>,<b>,<u> etc...)

Theory:
HTML (HyperText Markup Language) is the standard language used to create and design web pages. It uses various tags to structure and format content on a webpage. Each HTML tag has a specific purpose, and when combined, they provide a comprehensive way to present information in a clear and organized manner.

1. Headings (`<h1>` to `<h6>`):

- HTML provides six levels of headings, where `<h1>` defines the most important heading, and `<h6>` defines the least important. Headings are used to create a hierarchy and organize content, making it easier for users to read and navigate a webpage.

2. Paragraphs (`<p>`):

- The `<p>` tag defines a block of text as a paragraph. It is used to separate sections of text, creating a clear and organized structure. Paragraphs automatically have some space before and after them.

3. Text Formatting Tags:

- **Bold (`<b>` and `<strong>`):** Both tags are used to make text bold. The `<b>` tag is used for stylistic purposes, while `<strong>` implies that the text has strong importance or emphasis.
- **Italic (`<i>` and `<em>`):** These tags are used to italicize text. The `<i>` tag is purely for styling, whereas `<em>` emphasizes the importance of the text.
- **Underline (`<u>`):** The `<u>` tag is used to underline text.
- **Strikethrough (`<s>`):** The `<s>` tag displays text with a line through it, indicating that the text is no longer relevant.
- **Superscript (`<sup>`) and Subscript (`<sub>`):** These tags are used for text that should be displayed slightly above (`<sup>`) or below (`<sub>`) the normal line of text, often used for scientific formulas or footnotes.
- **Monospaced Text (`<code>`):** The `<code>` tag is used to display text in a monospaced font, typically for representing code snippets.

4. Blockquote (`<blockquote>`):

- The `<blockquote>` tag is used to define a section that is quoted from another source. It is typically indented to distinguish it from the surrounding content.

5. Lists (`<ul>` and `<ol>`):

- **Unordered List (`<ul>`):** This tag is used to create a list of items that do not need to be in a specific order. Each item within the list is marked with the `<li>` tag.
- **Ordered List (`<ol>`):** Similar to an unordered list, but the items are numbered, indicating a sequence or ranking.

6. Links (`<a>`):

- The `<a>` tag is used to create hyperlinks. These can link to other web pages, different sections of the same page, or even trigger downloads. The `href` attribute within the `<a>` tag specifies the URL of the linked resource.

7. Images (`<img>`):

- The `<img>` tag is used to embed images in a webpage. The `src` attribute specifies the path to the image file, and the `alt` attribute provides alternative text if the image cannot be displayed.

8. Horizontal Rule (`<hr>`):

- The `<hr>` tag inserts a horizontal line across the page, often used to separate sections of content.

9. Line Break (`<br>`):

- The `<br>` tag is used to insert a line break, forcing the text to continue on a new line.

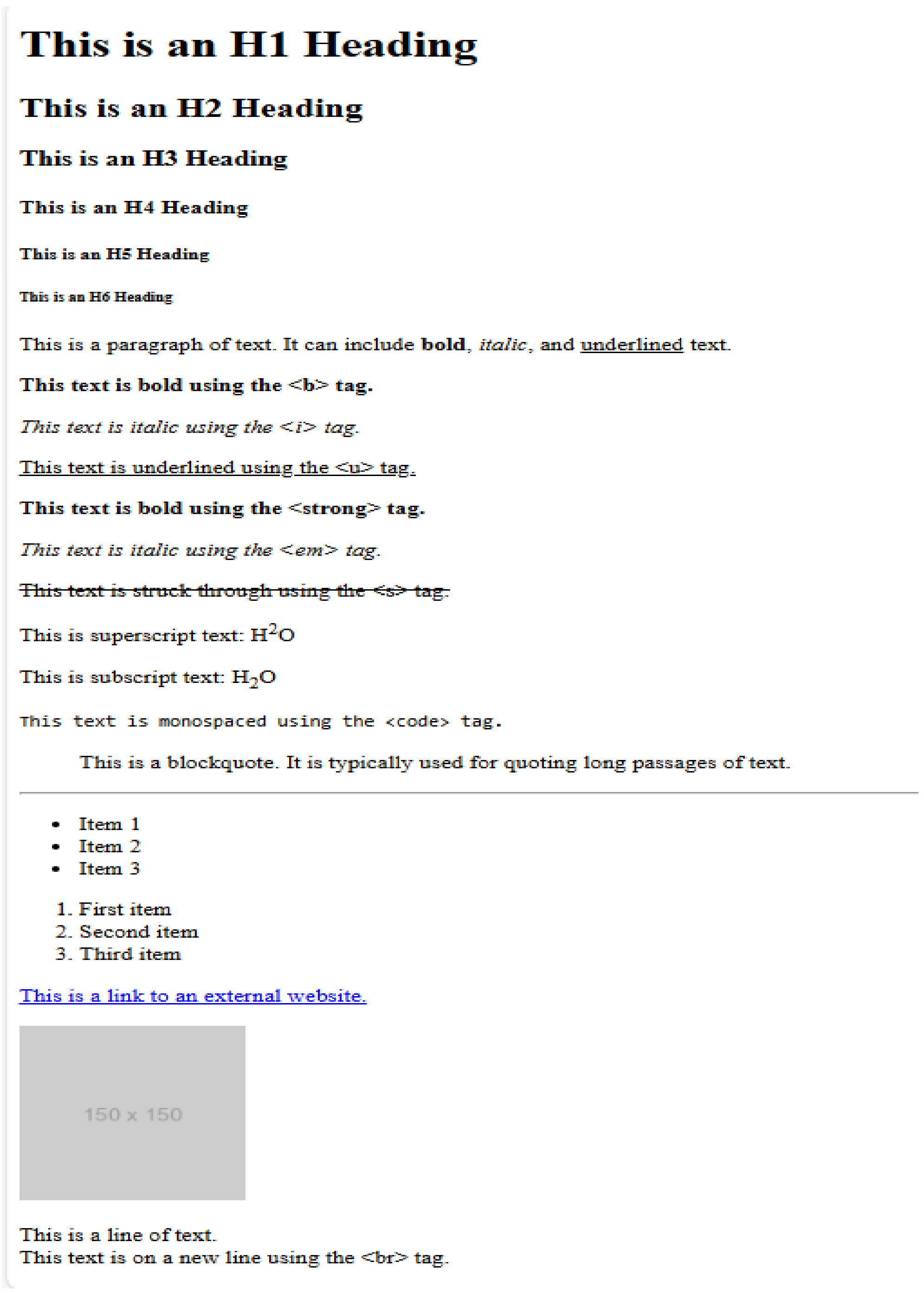
10. Comments (`<!-- -->`):

- HTML comments are used to include notes or annotations within the HTML code. These comments are not displayed in the browser and are useful for developers when reviewing or debugging code.

Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Basic HTML Tags and Formatting</title>
</head>
<body>
   <!-- Headings -->
   <h1>This is an H1 Heading</h1>
   <h2>This is an H2 Heading</h2>
   <h3>This is an H3 Heading</h3>
   <h4>This is an H4 Heading</h4>
   <h5>This is an H5 Heading</h5>
   <h6>This is an H6 Heading</h6>
   <!-- Paragraph -->
   <p>This is a paragraph of text. It can include <strong>bold</strong>, <em>italic</em>,
and <u>underlined</u> text.</p>

   <!-- Bold text -->
   <p><b>This text is bold using the &lt;b&gt; tag.</b></p>
   <!-- Italic text -->
   <p><i>This text is italic using the &lt;i&gt; tag.</i></p>
   <!-- Underlined text -->
   <p><u>This text is underlined using the &lt;u&gt; tag.</u></p>
   <!-- Strong text (bold) -->
   <p><strong>This text is bold using the &lt;strong&gt; tag.</strong></p>
   <!-- Emphasized text (italic) -->
   <p><em>This text is italic using the &lt;em&gt; tag.</em></p>
   <!-- Strikethrough text -->
   <p><s>This text is struck through using the &lt;s&gt; tag.</s></p>
```

```
    <!-- Superscript text -->
    <p>This is superscript text: H<sup>2</sup>O</p>
    <!-- Subscript text -->
    <p>This is subscript text: H<sub>2</sub>O</p>
    <!-- Monospaced (code) text -->
    <p><code>This text is monospaced using the &lt;code&gt; tag.</code></p>
    <!-- Blockquote -->
    <blockquote>
        This is a blockquote. It is typically used for quoting long passages of text.
    </blockquote>
    <!-- Horizontal rule -->
    <hr>
    <!-- Unordered list -->
    <ul>
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
    </ul>
    <!-- Ordered list -->
    <ol>
        <li>First item</li>
        <li>Second item</li>
        <li>Third item</li>
    </ol>
    <!-- Link -->
    <p><a href="https://www.example.com">This is a link to an external website.</a></p>
    <!-- Image -->
    <p><img src="https://via.placeholder.com/150" alt="Placeholder Image"></p>
    <!-- Line break -->
    <p>This is a line of text.<br>This text is on a new line using the &lt;br&gt; tag.</p>
    <!-- HTML comment -->
    <!-- This is an HTML comment. It will not be displayed in the browser. -->
</body>
</html>
```
Output:



This is an H1 Heading

This is an H2 Heading

This is an H3 Heading

This is an H4 Heading

This is an H5 Heading

This is an H6 Heading

This is a paragraph of text. It can include **bold**, *italic*, and underlined text.

This text is bold using the <b> tag.

This text is italic using the <i> tag.

This text is underlined using the <u> tag.

This text is bold using the <strong> tag.

This text is italic using the <em> tag.

~~This text is struck through using the <s> tag.~~

This is superscript text: H^2O

This is subscript text: H_2O

`This text is monospaced using the <code> tag.`

> This is a blockquote. It is typically used for quoting long passages of text.

- Item 1
- Item 2
- Item 3

1. First item
2. Second item
3. Third item

This is a link to an external website.

This is a line of text.
This text is on a new line using the <br> tag.

Conclusion: In this practical, we explored the fundamental HTML tags and text formatting methods that form the basis of web page structure and design. By using various heading levels, paragraph tags, and text formatting options like bold, italic, underline, and lists, we can create well-organized, readable, and visually engaging web content.

Practical No. 2

Aim: Create a HTML file which displays 3 images at LEFT, RIGHT and CENTER respectively in the browser and Create a HTML file which contains hyperlinks.

Theory:

Image Alignment and Hyperlinks in HTML

HTML (HyperText Markup Language) is the standard language used to create and design web pages. It provides a wide range of tags and attributes that allow developers to structure content, control the layout, and create interactive elements. Two important aspects of web design include image alignment and the creation of hyperlinks, both of which play crucial roles in enhancing the user experience and functionality of a website.

1. Image Alignment in HTML:

Image alignment involves positioning images on a webpage to achieve a desired layout. This is essential for creating visually appealing and well-structured pages.

Using CSS for Alignment:

Floating Images (float): The float property in CSS is commonly used to align images to the left or right of a container. When an image is floated, the surrounding text will wrap around it, creating a fluid and dynamic layout. For example, float: left; aligns an image to the left, while float: right; aligns it to the right.

Centering Images: To center an image, the margin-left: auto; and margin-right: auto; properties are used in conjunction with display: block;. This ensures that the image is centered horizontally within its container. Centering images is particularly useful for drawing attention to specific content or creating a balanced design.

2. Hyperlinks in HTML:

Hyperlinks are a fundamental component of the web, allowing users to navigate between different pages or sections within a webpage, link to external resources, initiate downloads, and even trigger email clients.

Types of Hyperlinks:

External Links: These are used to direct users to a different website or resource outside of the current webpage. The <a> tag with the href attribute specifies the URL, and the target="_blank" attribute can be added to open the link in a new tab.

Internal Links: Internal links navigate to different sections within the same webpage. This is done using the href attribute with a # symbol followed by the ID of the target element. Internal links are useful for creating a table of contents or quick navigation within a long page.

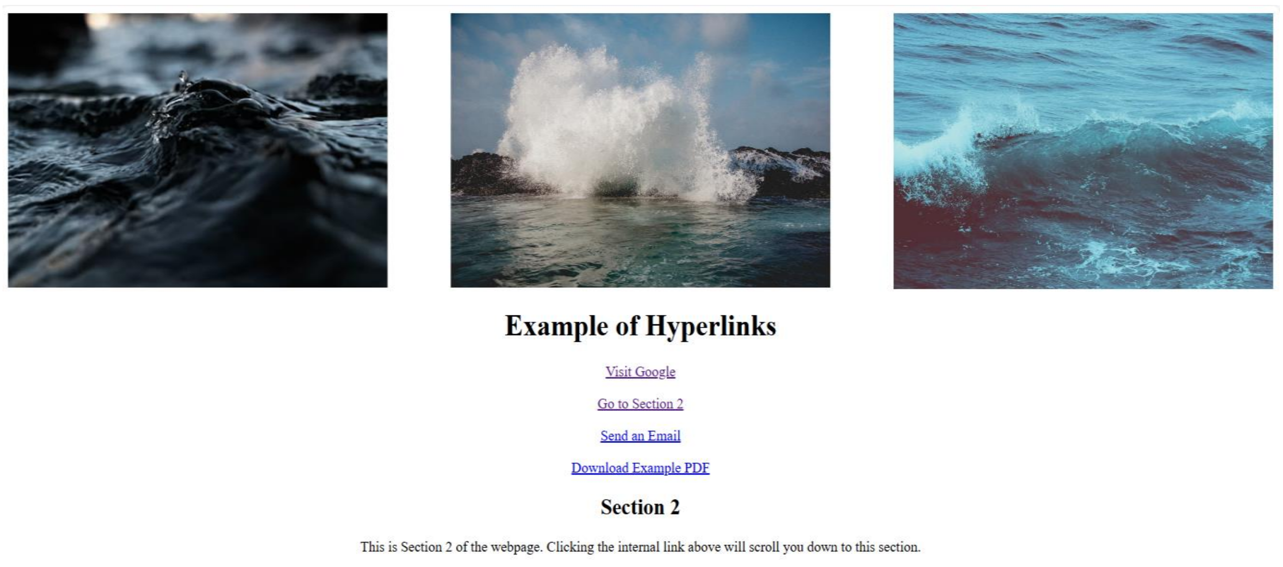
Email Links: The mailto: protocol in the href attribute creates a link that opens the user's default email client with a new message to the specified address. This is commonly used for contact links on websites.

Download Links: Adding the download attribute to an <a> tag prompts the browser to download the linked file rather than opening it. This is often used for providing resources like PDFs, images, or other downloadable content.

Program:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Image Alignment</title>
    <style>
        .left {
            float: left;
            width: 30%;
        }
        .right {
            float: right;
            width: 30%;
        }
        .center {
            display: block;
            margin-left: auto;
            margin-right: auto;
            width: 30%;
        }
        body {
            text-align: center;
        }
    </style>
</head>
<body>
    <img src="image 1.jpeg" alt="Left Image" class="left">
    <img src="Image 2.jpeg" alt="Right Image" class="right">
    <img src="image 3 .jpeg" alt="Center Image" class="center">
<h1>Example of Hyperlinks</h1>
    <!-- External Link -->
    <p><a href="https://www.google.com" target="_blank">Visit Google</a></p>
    <!-- Internal Link -->
    <p><a href="#section2">Go to Section 2</a></p>
    <!-- Email Link -->
    <p><a href="mailto:example@example.com">Send an Email</a></p>
    <!-- Download Link -->
    <p><a href="example.pdf" download>Download Example PDF</a></p>
    <!-- Section 2 (for internal link) -->
    <h2 id="section2">Section 2</h2>
    <p>This is Section 2 of the webpage. Clicking the internal link above will scroll you down to this section.</p>
</body>
</html>
```

Output:



Example of Hyperlinks

Visit Google

Go to Section 2

Send an Email

Download Example PDF

Section 2

This is Section 2 of the webpage. Clicking the internal link above will scroll you down to this section.

Conclusion:

Understanding and applying image alignment techniques and hyperlink creation is essential for developing effective web pages. Aligning images correctly ensures a visually appealing layout, while hyperlinks provide the necessary navigation and interactivity for a seamless user experience.

Practical No. 3

Aim: Create table with ROWSPAN and COLSPAN attribute of TABLE in HTML (Prepare timetable of your class) Include CELLSPACING & CELL PADDING

Theory: Creating Tables with ROWSPAN, COLSPAN, CELLSPACING, and CELLPADDING in HTML

Tables in HTML are a fundamental way to organize and display data in a structured format, using rows and columns. HTML provides various attributes to control the layout and appearance of tables, including rowspan, colspan, cellspacing, and cellpadding. These attributes enhance the visual representation of tabular data, making it easier to read and interpret.

1. Basic Table Structure:

An HTML table is defined using the <table> tag. Inside the table, rows are created with the <tr> tag (Table Row), and each row contains cells, defined by <td> (Table Data) for standard cells and <th> (Table Header) for header cells.

Example:

```
<table>
  <tr>
    <th>Header 1</th>
    <th>Header 2</th>
  </tr>
  <tr>
    <td>Data 1</td>
    <td>Data 2</td>
  </tr>
</table>
```

2. ROWSPAN Attribute:

The rowspan attribute allows a table cell to span multiple rows. This is useful when you want to merge cells vertically.

Example:

```
<td rowspan="2">Merged Cell</td>
```

In the above example, the cell will occupy two rows, effectively merging the space of two cells into one.

3. COLSPAN Attribute:

The colspan attribute allows a table cell to span multiple columns. This is used to merge cells horizontally.

Example:

```
<td colspan="3">Merged Cell</td>
```

Here, the cell will span across three columns, merging them into a single wide cell.

4. CELLSPACING Attribute:

The cellspacing attribute defines the space between the cells in a table. This attribute is set within the <table> tag.

Example:

<table cellspacing="10">

The value 10 specifies that there will be 10 pixels of space between each cell in the table.

5. CELLPADDING Attribute:

The cellpadding attribute sets the amount of space between the content of a cell and its borders. It is also set within the <table> tag.

Example:

<table cellpadding="10">

This means that the content inside each cell will have 10 pixels of padding around it, creating a buffer between the text and the cell border.

Application: Class Timetable

In this practical, these attributes are applied to create a class timetable:

Row and Column Merging: rowspan and colspan are used to merge cells where subjects span across multiple time slots or days, making the timetable clear and concise.

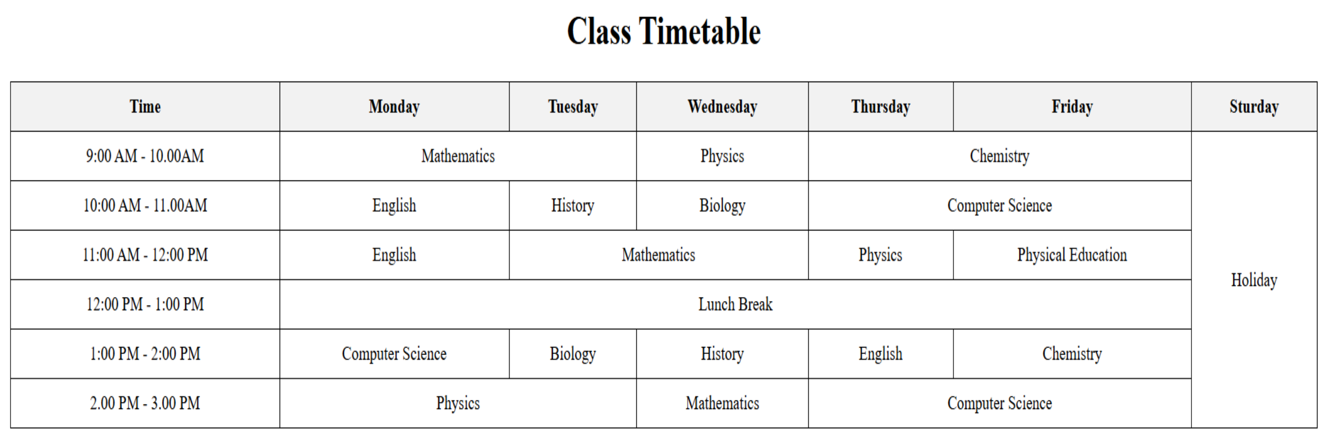
Spacing and Padding: cellspacing is used to create space between cells, and cellpadding ensures that the text within each cell has adequate space, improving readability.

Program:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Class Timetable</title>

  <style>

    table {

      width: 100%;

      border-collapse: collapse;

      margin: 20px 0;

    }

    table, th, td {

      border: 1px solid black;

    }
```

```
        th, td {
            padding: 10px;
            text-align: center;
        }
        th {
            background-color: #f2f2f2;
        }
    </style>
</head>
<body>

    <h1 style="text-align: center;">Class Timetable</h1>

    <table cellspacing="5" cellpadding="10">
        <tr>
            <th>Time</th>
            <th>Monday</th>
            <th>Tuesday</th>
            <th>Wednesday</th>
            <th>Thursday</th>
            <th>Friday</th>
            <th>Sturday</th>
        </tr>
        <tr>
            <td>9:00 AM - 10.00AM</td>
            <td colspan="2">Mathematics</td>
            <td>Physics</td>
            <td colspan="2">Chemistry</td>
            <td rowspan="6">Holiday</td>
        </tr>
        <tr>
            <td>10:00 AM - 11.00AM</td>
            <td>English</td>
            <td>History</td>
            <td>Biology</td>
            <td colspan="2">Computer Science</td>
        </tr>
        <tr>
            <td>11:00 AM - 12:00 PM</td>
            <td>English</td>
            <td colspan="2">Mathematics</td>
            <td>Physics</td>
            <td>Physical Education</td>
        </tr>
        <tr>
            <td>12:00 PM - 1:00 PM</td>
            <td colspan="5">Lunch Break</td>
        </tr>
        <tr>
            <td>1:00 PM - 2:00 PM</td>
            <td>Computer Science</td>
            <td>Biology</td>
            <td>History</td>
            <td>English</td>
            <td>Chemistry</td>
        </tr>
        <tr>
            <td>2.00 PM - 3.00 PM</td>
            <td colspan="2">Physics</td>
            <td>Mathematics</td>
            <td colspan="2">Computer Science</td>
        </tr>
    </table>

</body>
</html>
```

Output:



Class Timetable

Time	Monday	Tuesday	Wednesday	Thursday	Friday	Sturday
9:00 AM - 10.00AM	Mathematics		Physics	Chemistry		Holiday
10:00 AM - 11.00AM	English	History	Biology	Computer Science		
11:00 AM - 12:00 PM	English	Mathematics		Physics	Physical Education	
12:00 PM - 1:00 PM	Lunch Break					
1:00 PM - 2:00 PM	Computer Science	Biology	History	English	Chemistry	
2.00 PM - 3.00 PM	Physics		Mathematics	Computer Science		

Conclusion:

Understanding and applying rowspan, colspan, cellspacing, and cellpadding in HTML tables is crucial for creating well-structured and visually appealing tables. These attributes enable developers to present data in a more organized and readable format, which is especially useful in applications like creating timetables, schedules, and any other data-intensive layouts.

Practical No. 5

Aim: Write a HTML program to develop a static Login Page and static Registration Page.

Theory:

Developing Static Login and Registration Pages in HTML

Creating a static login and registration page using HTML is a foundational task in web development. These pages form the backbone of user authentication systems, allowing users to access and register for services on a website. Although these examples are static (i.e., they don't interact with a backend server), they demonstrate key concepts in structuring forms and applying basic CSS for layout and styling.

1. HTML Forms:

Forms in HTML are used to collect user input. A form typically includes various types of input fields like text boxes, password fields, email fields, checkboxes, radio buttons, and submit buttons.

Form Structure:

<form>: The container for form elements. The action attribute (if used) specifies the URL where the form data is sent for processing.

<label>: Describes the purpose of an input field. It is associated with a specific input field using the for attribute.

<input>: The most commonly used form element, with various types (e.g., text, password, email, submit).

<button>: Typically used for form submission, it triggers the action specified in the form.

2. Static Login Page:

Purpose: A login page allows users to enter their credentials (usually a username and password) to access a protected area of a website.

Key Elements:

Username Field (<input type="text">): Allows users to enter their username.

Password Field (<input type="password">): Masks the input characters for security.

Submit Button (<button type="submit">): Sends the form data for authentication (in a real scenario, this would be handled by a backend server).

3. Static Registration Page:

Purpose: A registration page enables new users to create an account by entering required information, such as a username, email, and password.

Key Elements:

Username Field: Similar to the login page.

Email Field (<input type="email">): Ensures the input follows the correct email format.

Password and Confirm Password Fields (<input type="password">): Collects and confirms the password, ensuring both match.

Submit Button: Initiates the account creation process.

4. CSS Styling:

Layout and Positioning: CSS is used to center the forms on the page, providing a user-friendly interface.

Container Styling: The login and registration forms are wrapped in a container <div>, which is styled with padding, border-radius, and box-shadow to create a modern, polished look.

Input Fields: The input fields are styled for consistency, with adequate padding and border-radius for a smooth user experience.

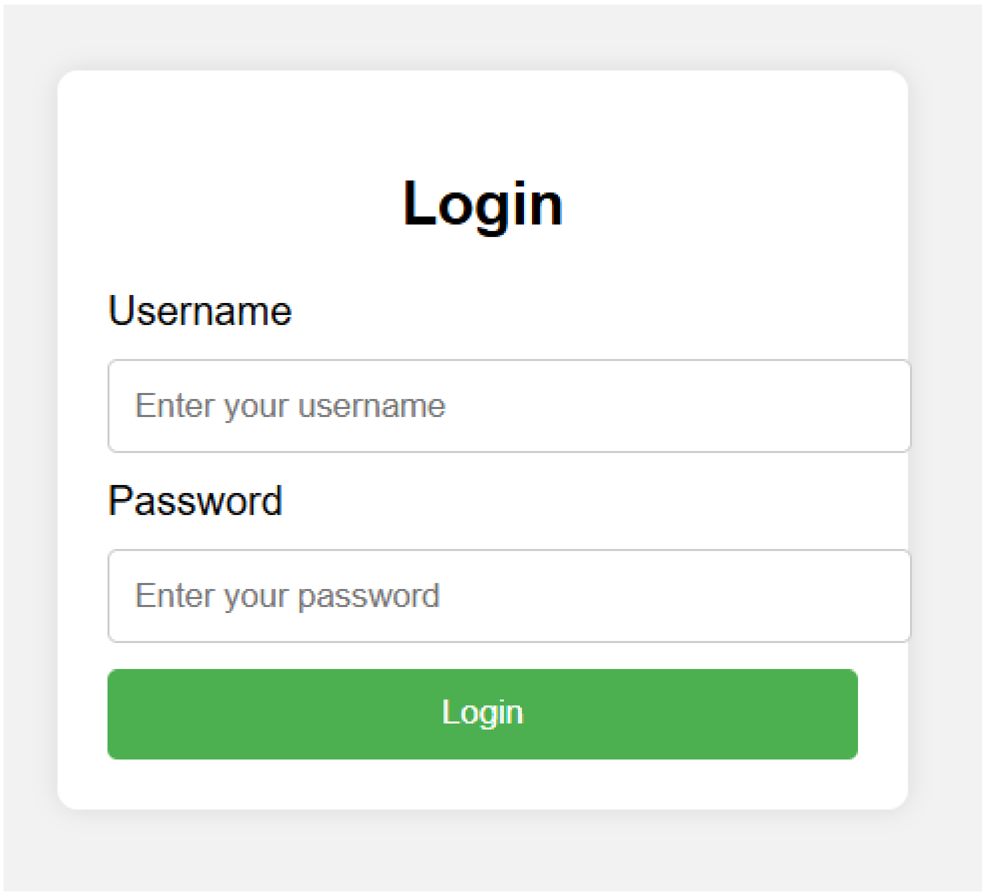
Button Styling: The submit buttons are styled to match the overall design, with hover effects to enhance interactivity.

Program: 1. For Static Login Page

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login Page</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f2f2f2;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
    }
    .login-container {
      background-color: white;
      padding: 20px;
      border-radius: 8px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
      width: 300px;
    }
    .login-container h2 {
      text-align: center;
      margin-bottom: 20px;
    }
```

```
        .login-container input[type="text"],
        .login-container input[type="password"] {
            width: 100%;
            padding: 10px;
            margin: 10px 0;
            border: 1px solid #ccc;
            border-radius: 4px;
        }
        .login-container button {
            width: 100%;
            padding: 10px;
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 4px;

            cursor: pointer;

        }
        .login-container button:hover {
            background-color: #45a049;
        }
    </style>

</head>

<body>

    <div class="login-container">

        <h2>Login</h2>

        <form action="#">

            <label for="username">Username</label>

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

            <label for="password">Password</label>

            <input type="password" id="password" name="password" placeholder="Enter your
password" required>

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

        </form>

    </div>

</body>

</html>
```

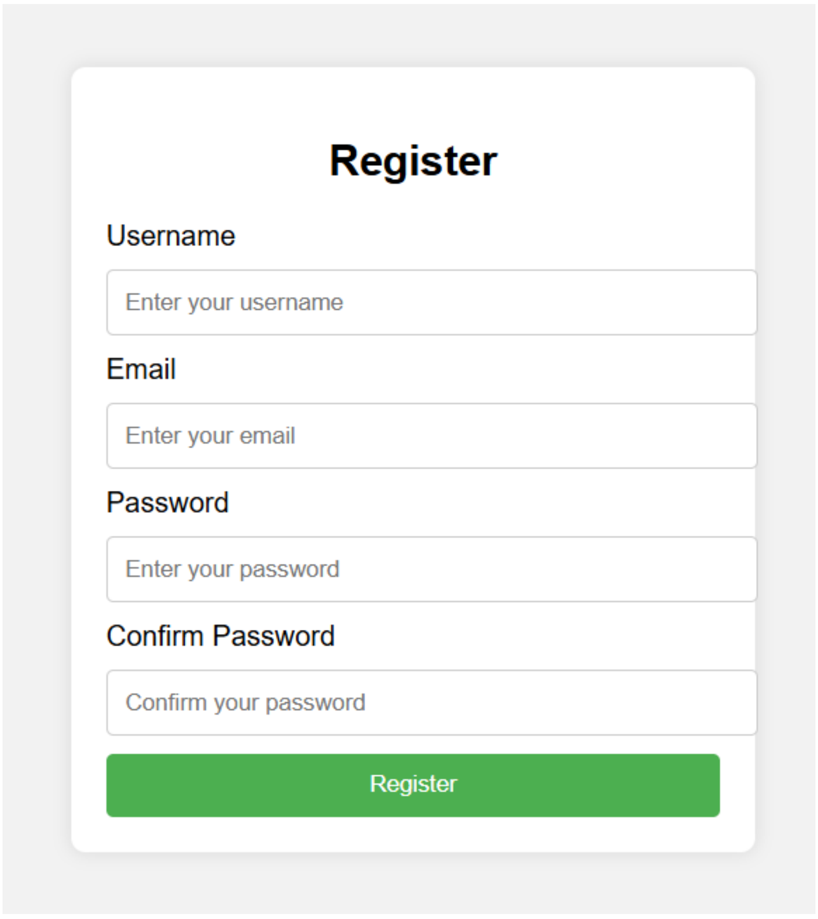
Output:



2. For Static Registration Page

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Registration Page</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f2f2f2;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
    }
    .registration-container {
      background-color: white;
      padding: 20px;
      border-radius: 8px;

      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

      width: 350px;

    }

    .registration-container h2 {
            text-align: center;

            margin-bottom: 20px;

        }

        .registration-container input[type="text"],
        .registration-container input[type="email"],
        .registration-container input[type="password"] {
            width: 100%;
            padding: 10px;
            margin: 10px 0;
            border: 1px solid #ccc;
            border-radius: 4px;
        }
        .registration-container button {
            width: 100%;
            padding: 10px;
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
        }
        .registration-container button:hover {
            background-color: #45a049;
        }
    </style>

</head>

<body>

    <div class="registration-container">

        <h2>Register</h2>

        <form action="#">

            <label for="username">Username</label>

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

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

            <input type="email" id="email" name="email" placeholder="Enter your email"
required>

            <label for="password">Password</label>

            <input type="password" id="password" name="password" placeholder="Enter your
password" required>

            <label for="confirm-password">Confirm Password</label>

            <input type="password" id="confirm-password" name="confirm-password"
placeholder="Confirm your password" required>

            <button type="submit">Register</button>
        </form>
    </div>
</body>
</html>
```

Output:



Conclusion:

Developing static login and registration pages in HTML is an essential skill for any web developer. These pages demonstrate how to collect and manage user input through forms. By applying CSS, developers can enhance the appearance and usability of these pages, providing a better user experience.

Practical No. 4

Aim: Demonstrate the types of CSS and following attributes using CSS

Color and background, Font, Text Border, Margin and list

Theory:

CSS (Cascading Style Sheets) is a stylesheet language used to describe the presentation (appearance and layout) of a document written in HTML or XML. CSS is a cornerstone technology of the web, alongside HTML and JavaScript, and is used to control how web pages are displayed in a browser.

Types of CSS

- **Inline CSS**: Applied directly within the HTML element using the style attribute.
- **Internal CSS**: Placed within a <style> tag in the <head> section of an HTML document.
- **External CSS**: Written in a separate .css file and linked to the HTML document.

CSS Attributes

1. **Color and Background**:
    - color: Sets the color of the text.
    - background-color: Sets the background color of an element.
2. **Font**:
    - font-family: Defines the font type (e.g., Arial, Verdana).
    - font-size: Specifies the size of the font.
3. **Text Border**:
    - border: Adds a border around the element. The format is border: [width] [style] [color];.
4. **Margin and List**:
    - margin: Defines the space outside the border of an element.
    - list-style-type: Specifies the bullet style for a list (e.g., square, circle).

Program:

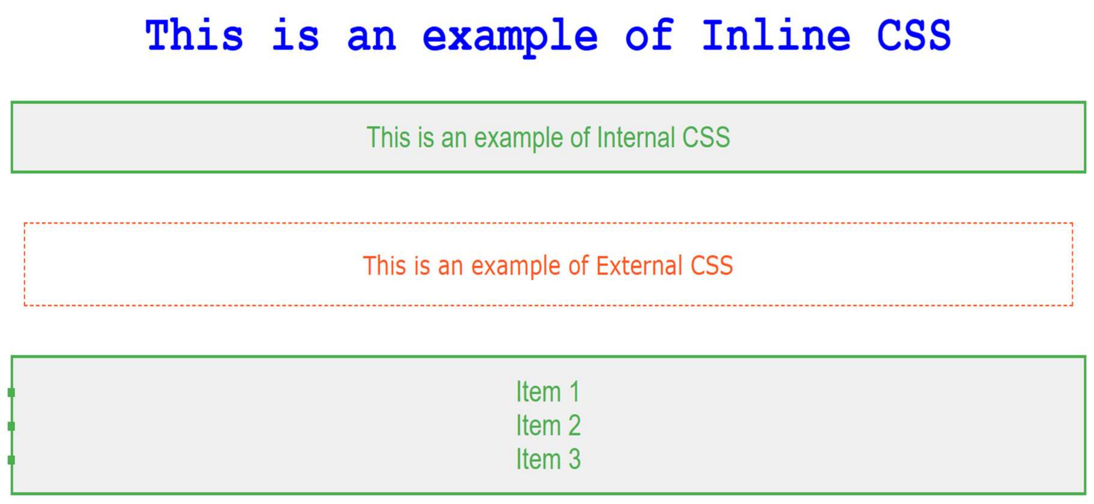
HTML File code

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>CSS Example</title>

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

  <!-- Internal CSS -->

  <style>

    /* Internal CSS styles */

    .internal-style {

      color: #4CAF50; /* Font color */

      background-color: #f0f0f0; /* Background color */
      font-family: Arial, sans-serif; /* Font */
      font-size: 18px; /* Font size */
      text-align: center; /* Text alignment */
      border: 2px solid #4CAF50; /* Text border */
      margin: 20px; /* Margin */
      padding: 10px; /* Padding */
      list-style-type: square; /* List style */
    }
  </style>
</head>
<body>
  <!-- Inline CSS -->
  <h1 style="color: blue; font-family: 'Courier New', Courier, monospace; text-align:
center;">
    This is an example of Inline CSS
  </h1>
  <!-- Internal CSS -->
  <div class="internal-style">
    This is an example of Internal CSS
  </div>
  <!-- Mimicking External CSS -->
  <div class="external-style">
    This is an example of External CSS
  </div>
  <!-- List Example -->
  <ul class="internal-style">
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
  </ul>
</body>
</html>
```

External CSS Code:

```
.external-style {
    color: #ff5722; /* Font color */
    background-color: #ffffff; /* Background color */
    font-family: 'Verdana', sans-serif; /* Font */
    font-size: 16px; /* Font size */
    text-align: center; /* Text alignment */
    border: 1px dashed #ff5722; /* Text border */
    margin: 30px; /* Margin */
    padding: 15px; /* Padding */
    list-style-type: circle; /* List style */
  }
```

Output:



```
This is an example of Inline CSS

This is an example of Internal CSS

This is an example of External CSS

Item 1
Item 2
Item 3
```

Conclusion: we explored the different types of CSS—inline, internal, and external—and how they can be used to style HTML documents effectively. We demonstrated the application of various CSS attributes, including color and background, font properties, text borders, margins, and list styles.

Practical No. 6

Aim: Create simple application using javascript that will demonstrate the following concepts:

● Declare And assign variable

● Operators and expression in JavaScript

● Looping in JavaScript

● Declare an Array

● User defined and Built in functions in Javascript

● Dialog boxes

Theory:

JavaScript is essential for front-end development and plays a vital role in enhancing the user experience on websites. Below is the theoretical overview of the key concepts covered in this practical:

1. Variables in JavaScript

Variables are used to store data that can be used and manipulated throughout a script. In JavaScript, variables are declared using the var, let, or const keywords:

let: Used to declare a variable whose value can change.

const: Used to declare a constant whose value cannot change after being set.

var: An older way to declare variables (generally replaced by let and const).

2. Operators and Expressions

Operators are symbols that perform operations on variables and values. Expressions combine variables, constants, and operators to compute new values.

Arithmetic Operators: +, -, *, /, % for mathematical calculations.

Assignment Operators: =, +=, -=, etc., for assigning values.

Comparison Operators: ==, ===, !=, >, <, etc., for comparing values.

Logical Operators: &&, ||, ! for logical operations.

3. Looping in JavaScript

Loops are used to execute a block of code repeatedly as long as a specified condition is true. JavaScript supports several types of loops:

for Loop: Executes a block of code a certain number of times.

while Loop: Executes a block of code as long as a specified condition is true.

do...while Loop: Similar to the while loop, but guarantees that the code block will execute at least once.

4. Arrays in JavaScript

An array is a data structure that can hold multiple values in a single variable. Arrays in JavaScript are dynamic and can store different data types such as numbers, strings, objects, etc.

5. Functions in JavaScript

Functions are reusable blocks of code designed to perform specific tasks. JavaScript supports:

User-Defined Functions: Created by the user to execute custom tasks.

Built-in Functions: Predefined functions provided by JavaScript (e.g., alert(), prompt(), console.log()).

Functions help in code organization, reusability, and modularity.

6. Dialog Boxes

JavaScript provides built-in functions to interact with users via dialog boxes:

alert(): Displays a message to the user.

prompt(): Asks the user for input and returns it.

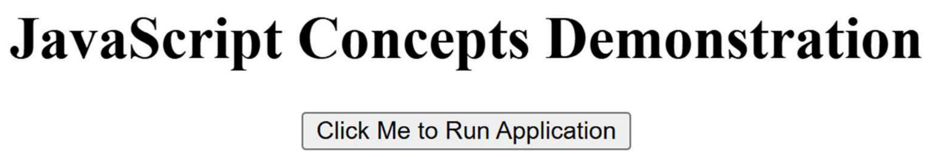
confirm(): Displays a message with OK and Cancel options and returns a boolean value (true for OK, false for Cancel).

Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>JavaScript Concepts Demo</title>
  <style>
    body{
      text-align: center;
    }
  </style>
</head>
<body>
  <h1>JavaScript Concepts Demonstration</h1>
  <button onclick="runApplication()">Click Me to Run Application</button>
  <script>
    // Function to run the application
    function runApplication() {
      // Declare and assign variables
      let userName = prompt("Enter your name:"); // Dialog box to get user input
      let age = parseInt(prompt("Enter your age:")); // Convert input to an integer
      // Operators and expressions
      let nextYearAge = age + 1; // Arithmetic operator
      let message = "Hello " + userName + "!"; // String concatenation
      // Output initial message using a dialog box
```

```
            alert(message + " You are currently " + age + " years old, and you will be " +
nextYearAge + " next year.");

            // Declare an array

            let colors = ["Red", "Green", "Blue", "Yellow", "Purple"];

            // Built-in function: Using 'length' property to get the array length

            alert("We have an array of colors: " + colors.join(", ") + ". The total number of colors
is: " + colors.length);

            // Looping through the array

            let colorMessage = "Here are the colors using a loop:\n";

            for (let i = 0; i < colors.length; i++) {

                colorMessage += (i + 1) + ". " + colors[i] + "\n"; // Creating a numbered list of
colors

            }

            // Displaying the list of colors using an alert dialog box

            alert(colorMessage);

            // User-defined function to calculate the square of a number

            function squareNumber(number) {

                return number * number;

            }

            // Using the user-defined function

            let numberToSquare = parseInt(prompt("Enter a number to find its square:"));

            let squaredResult = squareNumber(numberToSquare);

            // Displaying the result

            alert("The square of " + numberToSquare + " is: " + squaredResult);

        }

    </script>

</body>

</html>
```

Output:


JavaScript Concepts Demonstration

Conclusion:

Understanding how to declare variables, use operators, work with loops, define arrays, create functions, and interact with users through dialog boxes lays a solid foundation for building interactive and functional web applications.

Practical No. 7

Aim: Develop a javascript where user can select and change background and foreground color (hint: use color palette and button for changing)

Theory:

Implementing Dynamic Background and Foreground Color Selection in HTML

One of the powerful features of web development is the ability to create interactive web pages that respond to user actions. In this practical, we explore how to dynamically change the background and foreground (text) colors of a webpage using HTML, CSS, and JavaScript. This kind of interaction is a fundamental aspect of user experience design, allowing users to customize the appearance of a webpage in real-time.

1. HTML Basics:

HTML (HyperText Markup Language) is the standard language used to create and structure content on the web. It provides the structure of the webpage, while CSS (Cascading Style Sheets) is used for styling, and JavaScript is used for scripting or adding interactivity.

In this example, HTML is used to create a basic form with two dropdown (<select>) elements. Each dropdown contains several color options (<option>), allowing the user to choose a background color and a foreground (text) color.

2. CSS for Styling:

CSS (Cascading Style Sheets) is used to control the layout and appearance of the HTML elements on the page. In this case, CSS is applied to:

Center the content on the page using Flexbox, a modern CSS layout mode.

Style the dropdown menus to make them more user-friendly, with padding and font size adjustments.

3. JavaScript for Interactivity:

JavaScript is a versatile scripting language that enables dynamic content on websites. It can manipulate the HTML DOM (Document Object Model) to change elements on the page without requiring a page reload.

Functions for Color Changing:

changeBackgroundColor() Function: This function is triggered when the user selects a background color from the dropdown menu. It retrieves the selected value from the dropdown (<select> element) and applies it to the page's background using the bgColor property of document.body.

changeForegroundColor() Function: Similarly, this function is triggered when the user selects a foreground (text) color. It changes the text color on the page by setting the color style of document.body using inline CSS.

4. Event Handling:

In JavaScript, event handling refers to the ability to capture and respond to user actions, such as clicks, mouse movements, or key presses. In this practical:

The onchange event listener is used on the dropdown menus. This event triggers the associated JavaScript function whenever the user selects a different option from the dropdown, allowing the webpage to respond immediately by changing the colors.

5. DOM Manipulation:

The DOM (Document Object Model) represents the structure of a webpage as a tree of objects. JavaScript can access and modify these objects to dynamically change content, styles, and attributes.

By manipulating the DOM, JavaScript in this practical allows for real-time updates to the webpage's appearance based on user selections.

Practical Application:

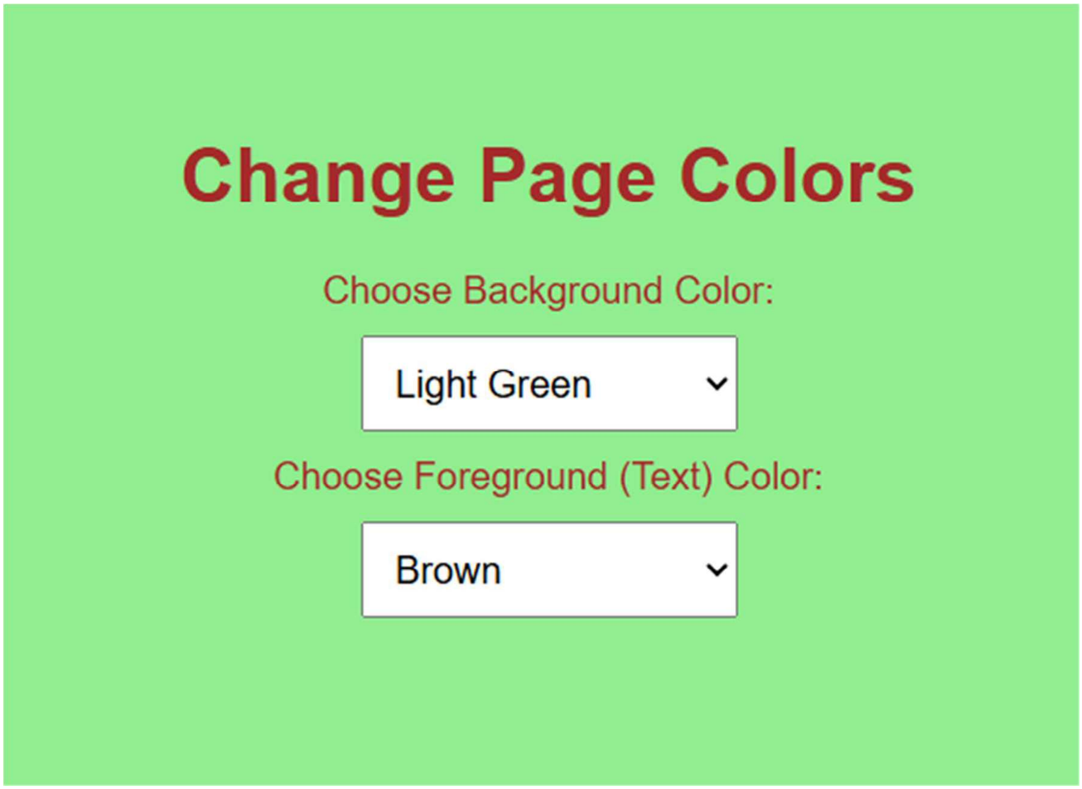
This type of interactive feature is common in web applications where customization is important. For instance, a user might want to personalize the appearance of their dashboard in a web application, or a developer might provide a theme switcher that allows users to toggle between light and dark modes.

Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Change Background and Foreground Color</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      flex-direction: column;
      text-align: center;
    }
    select {
      padding: 10px;
      margin: 10px;
      font-size: 16px;
    }
  </style>
</head>
<body>
  <h1>Change Page Colors</h1>
  <label for="bgColorSelect">Choose Background Color:</label>
```

```
    <select id="bgColorSelect" onchange="changeBackgroundColor()">
        <option value="">--Select a color--</option>
        <option value="white">White</option>
        <option value="lightgray">Light Gray</option>
        <option value="lightblue">Light Blue</option>
        <option value="lightgreen">Light Green</option>
        <option value="lightpink">Light Pink</option>
        <option value="yellow">Yellow</option>
    </select>
    <label for="fgColorSelect">Choose Foreground (Text) Color:</label>
    <select id="fgColorSelect" onchange="changeForegroundColor()">
        <option value="">--Select a color--</option>
        <option value="black">Black</option>
        <option value="red">Red</option>
        <option value="blue">Blue</option>
        <option value="green">Green</option>
        <option value="purple">Purple</option>
        <option value="brown">Brown</option>
    </select>
    <script>
        function changeBackgroundColor() {
            var bgColor = document.getElementById('bgColorSelect').value;
            document.body.bgColor = bgColor;
        }
        function changeForegroundColor() {
            var fgColor = document.getElementById('fgColorSelect').value;
            document.body.style.color = fgColor;
        }
    </script>
</body>
</html>
```

Output:



Conclusion:

The ability to dynamically change the background and foreground colors of a webpage enhances user interaction and provides a more engaging experience.

Practical No. 8

Aim: Design a javascript to change images every second

Theory:

Dynamic Image Changing and Status Message Scrolling with JavaScript

JavaScript is a powerful language for creating interactive and dynamic web experiences. This example demonstrates how to use JavaScript to dynamically change images at a very fast rate and to display a message in the status bar of the browser. These techniques are useful for understanding how JavaScript interacts with the DOM and how periodic actions can be managed.

1. Dynamic Image Changing:

Image Elements in HTML:

The <img> tag is used to embed images in a webpage. The src attribute specifies the path to the image file.

Example: <img id="changingImage" src="image1.jpg" alt="Image" />

JavaScript for Changing Images:

Array of Images: A JavaScript array holds the URLs of the images that will be displayed.

const images = ['image1.jpg', 'image2.jpg', 'image3.jpg', 'image4.jpg'];

Change Image Function (changeImage): This function updates the src attribute of the image element to display the next image in the array. It loops back to the first image after the last one.

function changeImage() {

```
    const imgElement = document.getElementById('changingImage');
    currentImageIndex = (currentImageIndex + 1) % images.length;
    imgElement.src = images[currentImageIndex];
}
```

Set Interval Function (setInterval): The setInterval method repeatedly executes the changeImage function every millisecond (or any specified interval).

```
function startImageChange() {
    setInterval(changeImage, 1);
}
```

2. Status Message Scrolling:

Status Bar Messages:

The status bar, typically located at the bottom of the browser window, can display messages using JavaScript. This feature is deprecated in most modern browsers due to security concerns.

Setting Status Message: The window.status property was used to display a message in the status bar.

```
function scrollStatusMessage(message) {
    window.status = message;
}
```

Modern Alternatives:

Due to security and user experience reasons, modern browsers have removed support for status bar messages. Alternatives include using in-page notifications or alerts to communicate information to users.

Application:

Dynamic Content: The ability to dynamically change images and other content based on user actions or timed events enhances the interactivity of web pages. This technique can be used for image sliders, dynamic galleries, and other interactive media.

Event Handling and Timing: Understanding how to use setInterval and other timing functions is essential for creating animations and periodic updates on a webpage.

Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Image Changer and Status Message</title>
    <style>
        #imageContainer {
            text-align: center;
            margin-top: 20px;
        }
        img {
            max-width: 100%;
            height: auto;
        }
    </style>
</head>
<body>
    <h1>Image Changer and Status Message</h1>
    <div id="imageContainer">
        <img id="changingImage" src="image1.jpg" alt="Image" />
    </div>
    <script>
        // Array of image URLs
        const images = [
            'image1.jpg',
            'image2.jpg',
            'image3.jpg',
            'image4.jpg'
        ];
        let currentImageIndex = 0;
        // Function to change the image
        function changeImage() {
            const imgElement = document.getElementById('changingImage');
            currentImageIndex = (currentImageIndex + 1) % images.length;
            imgElement.src = images[currentImageIndex];
        }
        // Change the image every 1 millisecond
        function startImageChange() {
            setInterval(changeImage, 1000);  // Change the image every 1000 millisecond
        }

        // Scroll a message in the status bar
        function scrollStatusMessage(message) {
            // Scroll the message in the status bar (may not work in modern browsers)
            window.status = message;
        }
        // Start the image changing process
        startImageChange();
        // Scroll a message
        scrollStatusMessage('This is a scrolling status message.');
    </script>
</body>
</html>
```

Output:

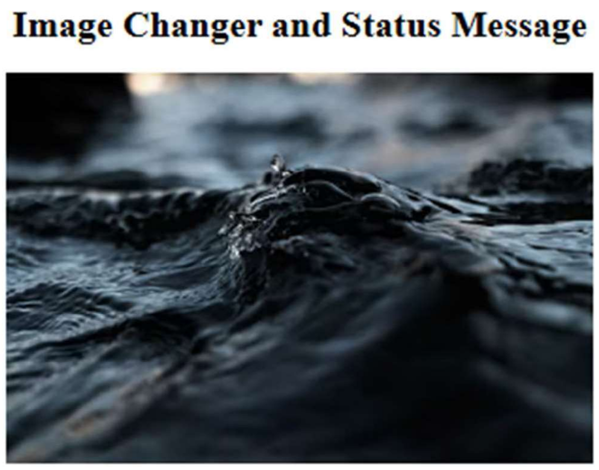


Image Changer and Status Message

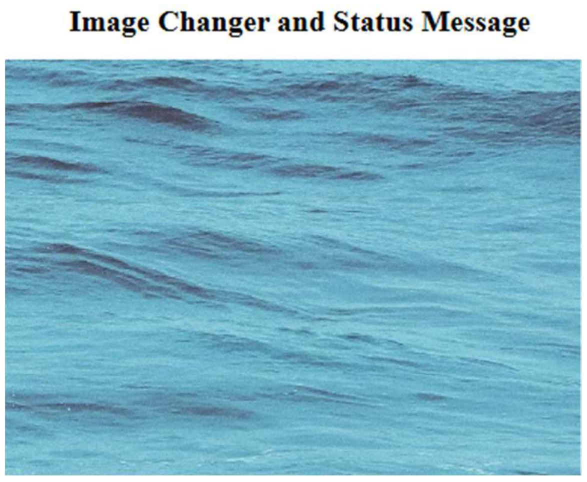


Image Changer and Status Message

Conclusion:

This example illustrates fundamental concepts in JavaScript, including DOM manipulation, event handling, and the use of timing functions like setInterval. While the status bar message feature is largely obsolete, the example provides historical context and demonstrates how JavaScript can be used to interact with different parts of the browser environment

Practical 9

Aim:

To demonstrate the usage of **XML** and **JSON** files in a web application for storing, retrieving, and displaying data using **JavaScript**.

Theory:

XML (eXtensible Markup Language) and **JSON (JavaScript Object Notation)** are popular formats for data interchange on the web. Both formats are used to store and transmit data between a server and a web application.

1. **XML**:
   - A markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.
   - Uses a tree structure to represent data with custom tags.
   - Heavily used in older systems and some specific areas like configuration files and legacy data formats.
2. **JSON**:
   - A lightweight data-interchange format that is easy for humans to read and write, and easy for machines to parse and generate.
   - Uses key-value pairs and arrays to represent data.
   - Preferred in modern web development for being simpler, faster, and more compact than XML.

XML vs JSON:

- **Readability**: JSON is more readable and less verbose compared to XML.
- **Efficiency**: JSON is more efficient in terms of parsing and data transfer.
- **Use Cases**: XML is still used in some legacy systems, while JSON is widely used in REST APIs and web services.

In Web Applications, both XML and JSON files are often used to fetch data from the server (AJAX) without reloading the webpage. This data is processed and displayed dynamically on the webpage using JavaScript.

Code:

Here we will create a simple web application that demonstrates fetching data from both **XML** and **JSON** files using **JavaScript** and displaying it dynamically.

1. `data.xml` (XML Data File):

```
<users>
  <user>
    <name>John Doe</name>
    <email>john.doe@example.com</email>
  </user>
  <user>
```

```
        <name>Jane Smith</name>
        <email>jane.smith@example.com</email>
    </user>
</users>
```

2. `data.json` (JSON Data File):

```
{
    "users": [
        {
            "name": "Alice Johnson",
            "email": "alice.johnson@example.com"
        },
        {
            "name": "Bob Brown",
            "email": "bob.brown@example.com"
        }
    ]
}
```

3. `index.html` (HTML File to Display Data):

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>XML and JSON Demo</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <h1>Data from XML and JSON</h1>
        <button id="loadXml">Load XML Data</button>
        <button id="loadJson">Load JSON Data</button>

        <h2>Data:</h2>
        <ul id="data-list"></ul>
    </div>

    <script src="script.js"></script>
</body>
</html>
```

4. `styles.css` (Optional CSS for Styling):

```
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 20px;
}

.container {
    max-width: 600px;
    margin: auto;
    background-color: #fff;
    padding: 20px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

button {
    padding: 10px 15px;
    margin: 5px;
    background-color: #007BFF;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

button:hover {
    background-color: #0056b3;
}

ul {
    list-style: none;
    padding: 0;
}

li {
    padding: 10px;
    border-bottom: 1px solid #ddd;
}
```

5. `script.js` (JavaScript to Fetch and Display Data):

```
document.getElementById("loadXml").addEventListener("click", loadXmlData);
document.getElementById("loadJson").addEventListener("click", loadJsonData);

// Function to load data from XML file
function loadXmlData() {
  const xhr = new XMLHttpRequest();
  xhr.open("GET", "data.xml", true);
  xhr.onload = function () {
    if (xhr.status === 200) {
      const xmlData = xhr.responseXML;
      const users = xmlData.getElementsByTagName("user");
      let output = "";
      for (let i = 0; i < users.length; i++) {
```

```
                const name = users[i].getElementsByTagName("name")[0].textContent;
                const email = users[i].getElementsByTagName("email")[0].textContent;
                output += `<li><strong>Name:</strong> ${name} <br><strong>Email:</strong> ${email}</li>`;
            }
            document.getElementById("data-list").innerHTML = output;
        }
    };
    xhr.send();
}

// Function to load data from JSON file
function loadJsonData() {
    const xhr = new XMLHttpRequest();
    xhr.open("GET", "data.json", true);
    xhr.onload = function () {
        if (xhr.status === 200) {
            const jsonData = JSON.parse(xhr.responseText);
            let output = "";
            jsonData.users.forEach(function (user) {
                output += `<li><strong>Name:</strong> ${user.name} <br><strong>Email:</strong>
${user.email}</li>`;
            });
            document.getElementById("data-list").innerHTML = output;
        }
    };
    xhr.send();
}
```

Output:

1. **Initial Screen**: When the page loads, the user will see two buttons: "Load XML Data" and "Load JSON Data."

Load XML Data: Clicking on the "Load XML Data" button will fetch data from the `data.xml` file and display the list of users (names and emails) on the webpage.
Example Output:
makefile
Copy code

```
Name: John Doe
Email: john.doe@example.com

Name: Jane Smith
Email: jane.smith@example.com
```

2.

Load JSON Data: Clicking on the "Load JSON Data" button will fetch data from the `data.json` file and display the list of users (names and emails) on the webpage.

Example Output:
makefile
Copy code
```
Name: Alice Johnson
Email: alice.johnson@example.com

Name: Bob Brown
Email: bob.brown@example.com
```

3.

Conclusion:

In this practical, we successfully demonstrated the use of **XML** and **JSON** files in a web application. We fetched data from both XML and JSON files using **AJAX** requests and displayed it dynamically on a webpage. This practical highlights the importance of XML and JSON as data formats in web applications and the ease of working with JSON compared to XML due to its simpler structure and better integration with JavaScript. JSON is generally preferred in modern web applications, especially for APIs and web services, due to its lightweight nature and speed.

Aim: **Write a PHP script to crate a data base student DB an to craate a table student in the data base student DB**

Theory:

MySQL Database: MySQL is a relational database management system (RDBMS) that stores data in tables. Each table consists of rows and columns where each column has a specific data type.

PHP: PHP is a server-side scripting language designed for web development. PHP can interact with databases, such as MySQL, using functions or extensions like mysqli or PDO (PHP Data Objects). The mysqli extension allows you to connect, query, and manipulate data stored in MySQL databases.

Steps in the PHP script:

1. **Connect to MySQL**: The script will connect to the MySQL server using a MySQL user account.
2. **Create Database**: The script will create a database called studentDB.
3. **Create Table**: Inside the studentDB database, the script will create a table called student with columns to store the student's ID, name, and age.

Program :

PHP Script

```
php

<?php

// Define database connection parameters

$servername = "localhost";  // Server name

$username = "root";        // MySQL username

$password = "";           // MySQL password


// Create connection

$conn = new mysqli($servername, $username, $password);


// Check connection

if ($conn->connect_error) {

   die("Connection failed: " . $conn->connect_error);

}

echo "Connected successfully to MySQL server.<br>";


// Create database
```

```
$sql = "CREATE DATABASE IF NOT EXISTS studentDB";
if ($conn->query($sql) === TRUE) {
   echo "Database studentDB created successfully.<br>";
} else {
   echo "Error creating database: " . $conn->error;
}

// Select the newly created database
$conn->select_db("studentDB");

// SQL query to create table
$table = "CREATE TABLE IF NOT EXISTS student (
   id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(50) NOT NULL,
   age INT(3) NOT NULL
)";

// Execute the query to create the table
if ($conn->query($table) === TRUE) {
   echo "Table 'student' created successfully in database studentDB.<br>";
} else {
   echo "Error creating table: " . $conn->error;
}

// Close the connection
$conn->close();
?>
```

Output:

When you run this PHP script, you should get the following output (assuming there are no errors):

```
Connected successfully to MySQL server.

Database student DB created successfully.
```

```
Table 'student' created successfully in database studentDB.
```

Conclusion:

This PHP script demonstrates how to create a MySQL database and table using PHP. The database studentDB is created, and a table student is created inside it with columns to store the student's ID, name, and age. This approach can be used to automate database creation and structure setup for web applications.