1. Lists, Links and Images

a) Write a_HTML program, to explain the working of lists.

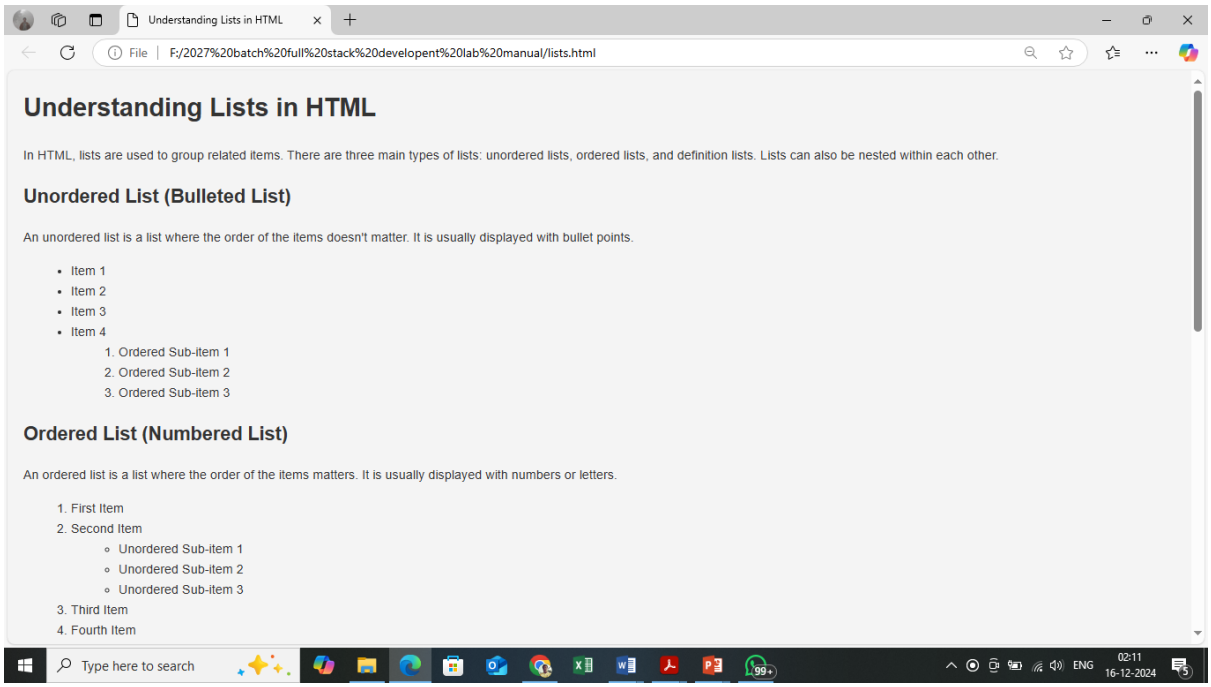
Note: It should have an ordered list, unordered list, nested definition lists. lists and ordered list in an unordered list and definition lists.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Understanding Lists in HTML</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            color: #333;
            line-height: 1.6;
            margin: 20px;
        }
        h1, h2, h3 {
            color: #333;
        }
        ul, ol, dl {
            margin-left: 20px;
        }
        dt {
            font-weight: bold;
        }
        dd {
            margin-left: 20px;
        }
    </style>
```

```
</head>
<body>
    <h1>Understanding Lists in HTML</h1>
    <p>In HTML, lists are used to group related items. There are three main types of lists: unordered lists, ordered lists, and definition lists. Lists can also be nested within each other.</p>
    <h2>Unordered List (Bulleted List)</h2>
    <p>An unordered list is a list where the order of the items doesn't matter. It is usually displayed with bullet points.</p>
    <ul>
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
        <li>Item 4
            <ol>
                <li>Ordered Sub-item 1</li>
                <li>Ordered Sub-item 2</li>
                <li>Ordered Sub-item 3</li>
            </ol>
        </li>
    </ul>
    <h2>Ordered List (Numbered List)</h2>
    <p>An ordered list is a list where the order of the items matters. It is usually displayed with numbers or letters.</p>
    <ol>
        <li>First Item</li>
        <li>Second Item
            <ul>
                <li>Unordered Sub-item 1</li>
                <li>Unordered Sub-item 2</li>
                <li>Unordered Sub-item 3</li>
            </ul>
```

```
        </li>
        <li>Third Item</li>
        <li>Fourth Item</li>
    </ol>

    <h2>Definition List</h2>
    <p>A definition list is a list of terms and their corresponding definitions.</p>
    <dl>
        <dt>HTML</dt>
        <dd>HyperText Markup Language, the standard language for creating web pages.</dd>
        <dt>CSS</dt>
        <dd>Cascading Style Sheets, used to style HTML elements.</dd>
        <dt>JavaScript</dt>
        <dd>A programming language used to create dynamic and interactive effects within web browsers.</dd>
        <dt>Nested Definition List</dt>
        <dd>
            <dl>
                <dt>Term 1</dt>
                <dd>Definition for term 1</dd>
                <dt>Term 2</dt>
                <dd>Definition for term 2</dd>
            </dl>
        </dd>
    </dl>
    <h2>Combining Lists</h2>
    <p>Lists can be combined and nested to create complex structures. Here is an example:</p>
    <ul>
        <li>Item with nested lists
```

```
        <ul>
            <li>Sub-item in an unordered list
                <ol>
                    <li>Ordered sub-item in an unordered list</li>
                    <li>Another ordered sub-item</li>
                </ol>
            </li>
        </ul>
    </li>
    <li>Another item with nested definition list
        <dl>
            <dt>Term in a nested list</dt>
            <dd>Definition for the term in a nested list</dd>
        </dl>
    </li>
</ul>
<h2>Conclusion</h2>
<p>HTML provides various ways to present data in a list format. Whether you use unordered lists for bullet points, ordered lists for numbered items, or definition lists for terms and definitions, lists help organize information clearly for users. Additionally, lists can be nested within each other to create more complex and organized structures.</p>
</body>
</html>
```


Understanding Lists in HTML
File | F:/2027%20batch%20full%20stack%20developent%20lab%20manual/lists.html
Understanding Lists in HTML
In HTML, lists are used to group related items. There are three main types of lists: unordered lists, ordered lists, and definition lists. Lists can also be nested within each other.
Unordered List (Bulleted List)
An unordered list is a list where the order of the items doesn't matter. It is usually displayed with bullet points.
Item 1
Item 2
Item 3
Item 4
1. Ordered Sub-item 1
2. Ordered Sub-item 2
3. Ordered Sub-item 3
Ordered List (Numbered List)
An ordered list is a list where the order of the items matters. It is usually displayed with numbers or letters.
1. First Item
2. Second Item
Unordered Sub-item 1
Unordered Sub-item 2
Unordered Sub-item 3
3. Third Item
4. Fourth Item
Type here to search
ENG
02:11
16-12-2024

1.b Write a HTML program, to explain the working of hyperlinks using <a> tag and href, target Attributes.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Understanding Hyperlinks in HTML</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            color: #333;
            line-height: 1.6;
            margin: 20px;
        }
        h1, h2 {
            color: #333;
        }
        p {
            margin-bottom: 20px;
        }
        a {
            color: #1a0dab;
            text-decoration: none;
        }
        a:hover {
            text-decoration: underline;
        }
    </style>
</head>
```

<body>

<h1>Understanding Hyperlinks in HTML</h1>

In HTML, hyperlinks are created using the <code><a></code> (anchor) tag. The <code>href</code> attribute specifies the URL of the page the link goes to, and the <code>target</code> attribute specifies where to open the linked document.

<h2>Basic Hyperlink</h2>

A basic hyperlink is created by using the <code><a></code> tag with the <code>href</code> attribute.

Visit Example.com

In the above example, clicking the link will take you to <code>https://www.example.com</code>.

<h2>Open Link in a New Tab</h2>

To open the link in a new tab, use the <code>target="_blank"</code> attribute.

Visit Example.com in a New Tab

In the above example, clicking the link will open <code>https://www.example.com</code> in a new tab.

<h2>Open Link in the Same Frame</h2>

To open the link in the same frame (default behavior), use the <code>target="_self"</code> attribute.

Visit Example.com in the Same Frame

In the above example, clicking the link will open <code>https://www.example.com</code> in the same tab or frame.

<h2>Open Link in a Parent Frame</h2>

To open the link in the parent frame, use the <code>target="_parent"</code> attribute. This is useful when dealing with frames.

Visit Example.com in the Parent Frame

<h2>Open Link in the Full Body of the Window</h2>

To open the link in the full body of the window, use the <code>target="_top"</code> attribute. This is also useful when dealing with frames.

Visit Example.com in the Full Body of the Window

<h2>Relative Links</h2>

Links can also be relative, pointing to other pages within the same website.

Contact Us

In the above example, clicking the link will take you to the <code>contact.html</code> page within the same website.

<h2>Email Links</h2>

You can also create a link to send an email using the <code>mailto:</code> scheme.

Send an Email

In the above example, clicking the link will open the default email client to send an email to <code>example@example.com</code>.

<h2>Conclusion</h2>

Hyperlinks are a fundamental part of HTML, allowing users to navigate between different pages and resources. By using the <code>href</code> attribute, you can specify the destination of the link, and by using the <code>target</code> attribute, you can control where the link opens.

</body>

</html>

1. C) Create a HTML document that has your image and your friends image with a specific height and width. Also when clicked on the images it should navigate to their respective profiles.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Profile Links</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            color: #333;
            text-align: center;
            margin: 20px;
        }
        h1 {
            color: #333;
        }
        .profile {
            display: inline-block;
            margin: 20px;
        }
        .profile img {
            width: 150px;
            height: 150px;
            border-radius: 50%;
            cursor: pointer;
            transition: transform 0.3s;
        }
```

```
        .profile img:hover {
            transform: scale(1.1);
        }    </style>
</head>
<body>
    <h1>Profile Links</h1>
    <div class="profile">
        <a href="https://www.example.com/my-profile" target="_blank">
            <img src="my-image.jpg" alt="My Profile Image">
        </a>
        <p>My Profile</p>
    </div>
    <div class="profile">
        <a href="https://www.example.com/friend-profile" target="_blank">
            <img src="friend-image.jpg" alt="Friend's Profile Image">
        </a>
        <p>Friend's Profile</p>
    </div>
</body>
</html>
```

1. D) Write a HTML program, 1n such a way that rather than placing large images on a page, the preferred technique is to use thumbnails by setting the height and width parameters to something like to 100*100 pixels. Each thumbnails image is also a link a full sized version of the image. Create a image gallery using this technique.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial
scale=1.0">
   <title>Image Gallery</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         background-color: #f4f4f4;
         color: #333;
         text-align: center;
         margin: 20px;
      }
      h1 {
         color: #333;
      }
      .gallery {
         display: flex;
         flex-wrap: wrap;
         justify-content: center;
      }
      .gallery-item {
         margin: 10px;
      }
      .gallery-item img {
         width: 100px;
         height: 100px;
         border-radius: 5px;
         cursor: pointer;
         transition: transform 0.3s;
      }
      .gallery-item img:hover {
         transform: scale(1.1);
      }
   </style>
</head>
<body>
```

```
    <h1>Image Gallery</h1>
    <div class="gallery">
        <div class="gallery-item">
            <a href="images/image1.jpg" target="_blank">
                <img src="images/thumbnails/image1.jpg" alt="Image 1">
            </a>
        </div>
        <div class="gallery-item">
            <a href="images/image2.jpg" target="_blank">
                <img src="images/thumbnails/image2.jpg" alt="Image 2">
            </a>
        </div>
        <div class="gallery-item">
            <a href="images/image3.jpg" target="_blank">
                <img src="images/thumbnails/image3.jpg" alt="Image 3">
            </a>
        </div>
        <div class="gallery-item">
            <a href="images/image4.jpg" target="_blank">
                <img src="images/thumbnails/image4.jpg" alt="Image 4">
            </a>
        </div>
        <!-- Add more images as needed -->
    </div>
</body>
</html>
```

2. HTML Tables. forms and frames

a. Write a HTML program, to explain the working of tables. (use tags: <table>, <tr>,<th>, <td> and attributes: border, rowspan, colspan)

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Understanding HTML Tables</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         background-color: #f4f4f4;
         color: #333;
         margin: 20px;
      }
      h1 {
         color: #333;
      }
      table {
         width: 100%;
         border-collapse: collapse;
         margin-bottom: 20px;
      }
      table, th, td {
         border: 1px solid #333;
      }
      th, td {
         padding: 10px;
         text-align: left;
      }
      th {
         background-color: #f2f2f2;
      }
   </style>
</head>
<body>
   <h1>Understanding HTML Tables</h1>

   <p>In HTML, tables are created using the <code>&lt;table&gt;</code> tag. The table is structured with rows (<code>&lt;tr&gt;</code>) and cells, which can be either header cells (<code>&lt;th&gt;</code>) or data cells (<code>&lt;td&gt;</code>). Attributes like <code>border</code>,
```

<code>rowspan</code>, and <code>colspan</code> can be used to enhance the table's appearance and structure.</p>

```
    <h2>Basic Table</h2>
    <table border="1">
        <tr>
            <th>Header 1</th>
            <th>Header 2</th>
            <th>Header 3</th>
        </tr>
        <tr>
            <td>Row 1, Cell 1</td>
            <td>Row 1, Cell 2</td>
            <td>Row 1, Cell 3</td>
        </tr>
        <tr>
            <td>Row 2, Cell 1</td>
            <td>Row 2, Cell 2</td>
            <td>Row 2, Cell 3</td>
        </tr>
    </table>

    <h2>Table with Rowspan</h2>
    <p>The <code>rowspan</code> attribute allows a cell to span multiple rows.</p>
    <table border="1">
        <tr>
            <th>Header 1</th>
            <th>Header 2</th>
            <th>Header 3</th>
        </tr>
        <tr>
            <td rowspan="2">Rowspan 2 Rows</td>
            <td>Row 1, Cell 2</td>
            <td>Row 1, Cell 3</td>
        </tr>
        <tr>
            <td>Row 2, Cell 2</td>
            <td>Row 2, Cell 3</td>
        </tr>
    </table>

    <h2>Table with Colspan</h2>
```

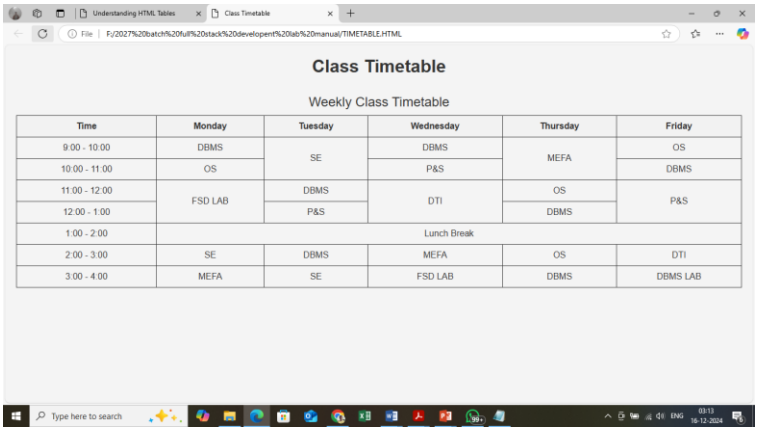
```
  <p>The <code>colspan</code> attribute allows a cell to span multiple columns.</p>
  <table border="1">
    <tr>
      <th>Header 1</th>
      <th>Header 2</th>
      <th>Header 3</th>
    </tr>
    <tr>
      <td>Row 1, Cell 1</td>
      <td colspan="2">Colspan 2 Columns</td>
    </tr>
    <tr>
      <td>Row 2, Cell 1</td>
      <td>Row 2, Cell 2</td>
      <td>Row 2, Cell 3</td>
    </tr>
  </table>

  <h2>Complex Table Example</h2>
  <p>This table combines both <code>rowspan</code> and <code>colspan</code> attributes.</p>
  <table border="1">
    <tr>
      <th>Header 1</th>
      <th>Header 2</th>
      <th>Header 3</th>
      <th>Header 4</th>
    </tr>
    <tr>
      <td rowspan="2">Rowspan 2 Rows</td>
      <td>Row 1, Cell 2</td>
      <td colspan="2">Colspan 2 Columns</td>
    </tr>
    <tr>
      <td>Row 2, Cell 2</td>
      <td>Row 2, Cell 3</td>
      <td>Row 2, Cell 4</td>
    </tr>
    <tr>
      <td>Row 3, Cell 1</td>
      <td colspan="3">Colspan 3 Columns</td>
    </tr>
  </table>  </body>  </html>
```

b. Write a HTML program, to explain the working of tables.by preparing a timetable. (Note: use <caption> tag to set the caption to the table & also use cell spacing, cell padding, border, rowspan, colspan etc.)

```
<!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>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
      color: #333;
      margin: 20px;
      text-align: center;
    }
    h1 {
      color: #333;
    }
    table {
      width: 100%;
      border-collapse: collapse;
      margin-bottom: 20px;
    }
    table, th, td {
      border: 1px solid #333;
    }
    th, td {
      padding: 10px;
      text-align: center;
    }
    th {
      background-color: #f2f2f2;
    }
    caption {
      caption-side: top;
      font-size: 1.5em;
      margin: 10px;
    }
  </style>
```

```
</head>
<body>
  <h1>Class Timetable</h1>

  <table cellspacing="0" cellpadding="10">
    <caption>Weekly Class Timetable</caption>
    <tr>
      <th>Time</th>
      <th>Monday</th>
      <th>Tuesday</th>
      <th>Wednesday</th>
      <th>Thursday</th>
      <th>Friday</th>
    </tr>
    <tr>
      <td>9:00 - 10:00</td>
      <td>DBMS</td>
      <td rowspan="2">SE</td>
      <td>DBMS</td>
      <td rowspan="2">MEFA</td>
      <td>OS</td>
    </tr>
    <tr>
      <td>10:00 - 11:00</td>
      <td>OS</td>
      <td>P&S</td>
      <td>DBMS</td>
    </tr>
    <tr>
      <td>11:00 - 12:00</td>
      <td rowspan="2">FSD LAB</td>
      <td>DBMS</td>
      <td rowspan="2">DTI</td>
      <td>OS</td>
      <td rowspan="2">P&S</td>
    </tr>
    <tr>
      <td>12:00 - 1:00</td>
      <td>P&S</td>
      <td>DBMS</td>
    </tr>
    <tr>
      <td>1:00 - 2:00</td>
      <td colspan="5">Lunch Break</td>
```

```
    </tr>
    <tr>
      <td>2:00 - 3:00</td>
      <td>SE</td>
      <td>DBMS</td>
      <td>MEFA</td>
      <td>OS</td>
      <td>DTI</td>
    </tr>
    <tr>
      <td>3:00 - 4:00</td>
      <td>MEFA</td>
      <td>SE</td>
      <td>FSD LAB</td>
      <td>DBMS</td>
      <td>DBMS LAB</td>
    </tr>
  </table>

</body>
</html>
```



Class Timetable

Weekly Class Timetable

Time	Monday	Tuesday	Wednesday	Thursday	Friday
9:00 - 10:00	DBMS	SE	DBMS	MEFA	OS
10:00 - 11:00	OS		P&S		DBMS
11:00 - 12:00	FSD LAB	DBMS	DTI	OS	P&S
12:00 - 1:00		P&S		DBMS	
1:00 - 2:00	Lunch Break				
2:00 - 3:00	SE	DBMS	MEFA	OS	DTI
3:00 - 4:00	MEFA	SE	FSD LAB	DBMS	DBMS LAB

1. **C)** Write a HTML program, to explain the working of forms, by designing registration form.
   Note: include text field, password field, number field, date of birth field, checkboxes, radio buttons, list boxes using <select> ,<option> tags, <text area> and two buttons submit and reset. Use tables to provide better views.

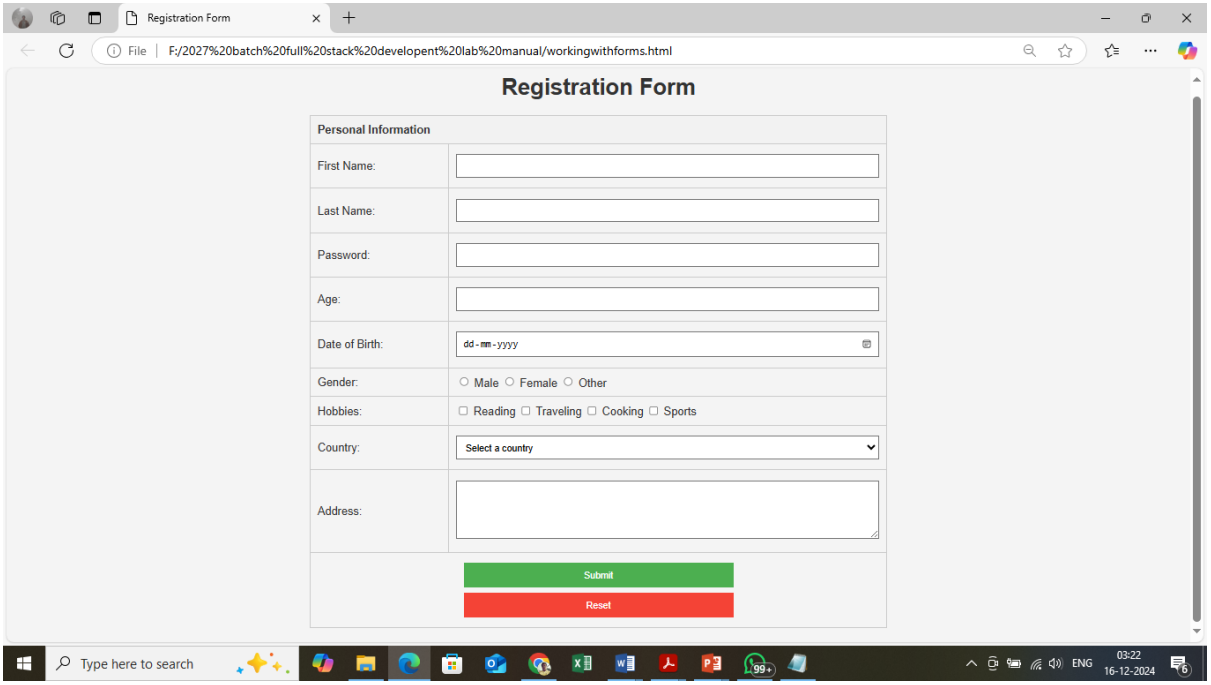
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Registration Form</title>
  <style>
```

```
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            color: #333;
            margin: 20px;
        }
        h1 {
            text-align: center;
            color: #333;
        }
        table {
            margin: 0 auto;
            border-collapse: collapse;
            width: 50%;
        }
        table, th, td {
            border: 1px solid #ccc;
        }
        th, td {
            padding: 10px;
            text-align: left;
        }
        th {
            background-color: #f2f2f2;
        }
        input[type="text"], input[type="password"],
input[type="number"], input[type="date"], select, textarea {
            width: 100%;
            padding: 8px;
            margin: 4px 0;
            box-sizing: border-box;
        }
        input[type="submit"], input[type="reset"] {
            width: 48%;
            padding: 10px;
            margin: 4px 1%;
            background-color: #4CAF50;
            color: white;
            border: none;
            cursor: pointer;
        }
        input[type="reset"] {
            background-color: #f44336;
        }
```

```
input[type="submit"]:hover, input[type="reset"]:hover {
            opacity: 0.9;
        }
    </style>
</head>
<body>
    <h1>Registration Form</h1>
    <form action="#" method="post">
        <table>
            <tr>
                <th colspan="2">Personal Information</th>
            </tr>
            <tr>
                <td>First Name:</td>
                <td><input type="text" name="firstname"
required></td>
            </tr>
            <tr>
                <td>Last Name:</td>
                <td><input type="text" name="lastname"
required></td>
            </tr>
            <tr>
                <td>Password:</td>
                <td><input type="password" name="password"
required></td>
            </tr>
            <tr>
                <td>Age:</td>
                <td><input type="number" name="age"
required></td>
            </tr>
            <tr>
                <td>Date of Birth:</td>
                <td><input type="date" name="dob" required></td>
            </tr>
            <tr>
                <td>Gender:</td>
                <td>
                    <input type="radio" name="gender" value="male"
required> Male
                    <input type="radio" name="gender" value="female"
required> Female
```

```
                    <input type="radio" name="gender" value="other"
            required> Other
                </td>
              </tr>
              <tr>
                <td>Hobbies:</td>
                <td>
                  <input type="checkbox" name="hobby1"
            value="Reading"> Reading
                  <input type="checkbox" name="hobby2"
            value="Traveling"> Traveling
                  <input type="checkbox" name="hobby3"
            value="Cooking"> Cooking
                  <input type="checkbox" name="hobby4"
            value="Sports"> Sports
                </td>
              </tr>
              <tr>
                <td>Country:</td>
                <td>
                  <select name="country" required>

<option value="">Select a country</option

<option value="USA">USA</option>

<option value="Canada">Canada</option>

<option value="UK">UK</option>

<option value="Australia">Australia</option

<option value="India">India</option>

</select>

             </td>
              </tr>
              <tr>
              <td>Address:</td>

 <td><textarea name="address" rows="4" required></textarea></td>

</tr>

              <tr>
              <td colspan="2" style="text-align: center;">
            <input type="submit" value="Submit">
            <input type="reset" value="Reset">
                </td>
```

```
            </tr>
        </table>
    </form>
</body>
</html>
```



2. **D)** Write a HTML program, to Explain the working of frames, such that page is to be divided into 3 parts on either direction. (Note: first frame image, second frame paragraph, third frame hyperlink. And also make sure of using no frame attribute such that frames to be fixed).

```
<!DOCTYPE html>
<html>
<head>
  <title>Frames Example</title>
</head>
<frameset rows="33%, 33%, 34%" frameborder="0"
framespacing="0">
  <frame src="image.html" name="image_frame"
frameborder="0" scrolling="no">
  <frame src="paragraph.html" name="paragraph_frame"
frameborder="0" scrolling="no">
  <frame src="hyperlink.html" name="hyperlink_frame"
frameborder="0" scrolling="no">
</frameset>
</html>
```

image.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Image Frame</title>
</head>
<body>
  <img src="image.jpg" width="100%" height="100%">
</body>
</html>
```

paragraph.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Paragraph Frame</title>
</head>
<body>
  <p>This is a paragraph of text in the second frame.</p>
</body>
</html>
```

hyperlink.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Hyperlink Frame</title>
</head>
<body>
  <p><a href="(link unavailable)">Visit (link
unavailable)</a></p>
</body></html>
```

3. HTML 5 and Cascading Style Sheets, Types of CSS

a) Write a HTML Program that makes use of <article>, <aside>, <figure>, <figcaption>, <footer>, <header>, <main>, <nav>, <section>, , , tags.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>HTML5 Semantic Tags Example</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         margin: 0;
         padding: 0;
      }
      header, footer {
         background-color: #333;
         color: white;
         padding: 10px;
         text-align: center;
      }
      nav {
         background-color: #f4f4f4;
         padding: 10px;
         text-align: center;
      }
      main {
         margin: 20px;
      }
      section {
         background-color: #e2e2e2;
         padding: 15px;
```

```
            margin-bottom: 20px;
        }
        article {
            background-color: #f9f9f9;
            padding: 15px;
            margin: 10px 0;
        }
        aside {
            background-color: #e9e9e9;
            padding: 10px;
            margin-top: 10px;
        }
        figure {
            display: inline-block;
            margin: 10px;
        }
        figcaption {
            font-size: 0.9em;
            text-align: center;
        }
        .content {
            display: flex;
            justify-content: space-between;
        }
    </style>
</head>
<body>
<header>
    <h1>Welcome to My Website</h1>
    <p>Your one-stop destination for information</p>
</header>
<nav>
    <ul>
        <li><a href="#home">Home</a></li>
        <li><a href="#about">About</a></li>
        <li><a href="#services">Services</a></li>
        <li><a href="#contact">Contact</a></li>
    </ul>
</nav>
<main>
    <section>
        <h2>Introduction</h2>
        <p>This is the main section of the page where the content
will be displayed.</p>
```

```
    </section>
    <div class="content">
        <article>
            <h2>Article 1: Exploring HTML5</h2>
<p>HTML5 is the latest version of HTML and includes many new features and improvements...</p>
        </article>
        <article>
      <h2>Article 2: Understanding CSS3</h2>
<p>CSS3 is the latest evolution of the Cascading Style Sheets language...</p>
        </article>
    </div>
    <aside>
        <h3>Related Resources</h3>
        <p>Check out these resources for more information on web development:</p>
        <ul>
            <li><a href="https://developer.mozilla.org/en-US/docs/Web/HTML">MDN Web Docs</a></li>
            <li><a href="https://www.w3schools.com/">W3Schools</a></li>
            <li><a href="https://www.css-tricks.com/">CSS-Tricks</a></li>
        </ul>
    </aside>
```

```
    <figure>
     <img src="https://via.placeholder.com/300" alt="HTML5
    Logo">
    <figcaption>HTML5 Logo</figcaption>
    </figure>
    </main>
    <footer>
    <p>&copy; 2024 My Website. All Rights Reserved.</p>
    </footer>
    </body>
    </html>
```

3.b) Write a HTML program to embed audio and video into HTML web page.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Embedding Audio and Video</title>
</head>
<body>
  <h1>Embedding Audio and Video into HTML</h1>
  <!-- Audio Section -->
  <section>
    <h2>Audio Example</h2>
    <p>Click the play button to listen to the audio file.</p>
    <audio controls>
<source src="https://www.soundhelix.com/examples/mp3/SoundHelix-Song-1.mp3" type="audio/mp3">Your browser does not support the audio element.
 </audio>
  </section>
  <!-- Video Section -->
  <section>
```

<h2>Video Example</h2>

** Click the play button to watch the video. <video width="640" height="360" controls>**

<source src="https://www.w3schools.com/html/movie.mp4" type="video/mp4">

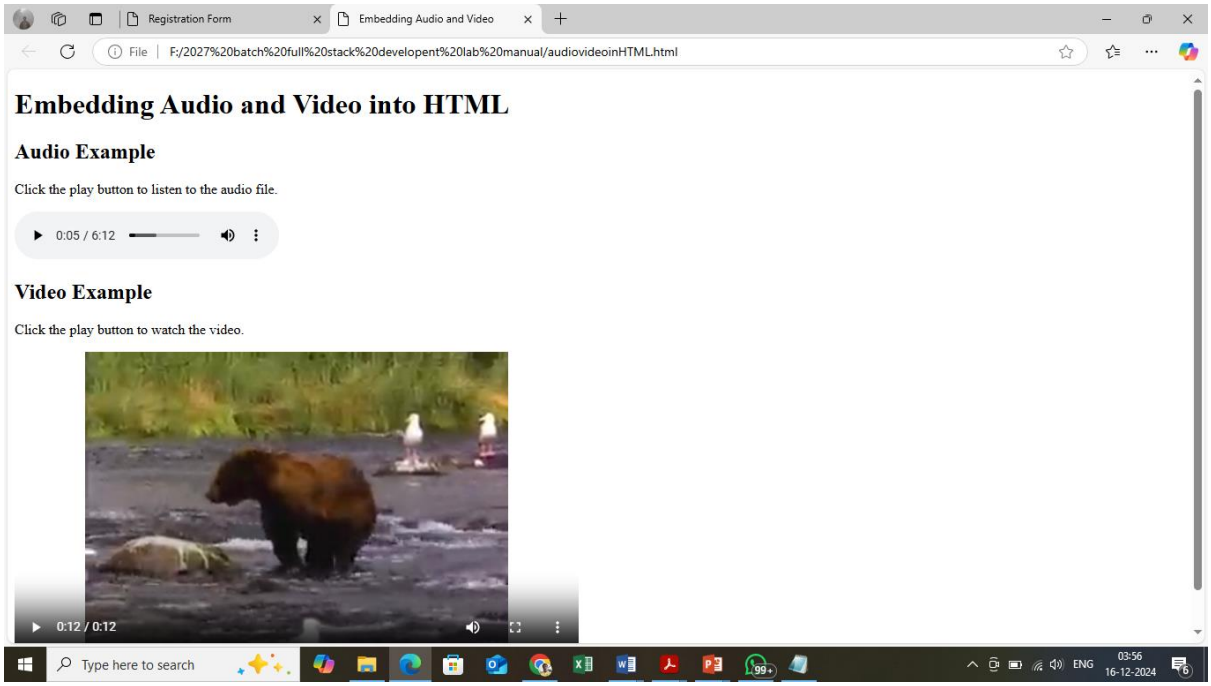
<source src="https://www.w3schools.com/html/movie.ogg" type="video/ogg"> Your browser does not support the video element.

</video>

</section>

</body>

</html>



3.C) Write a program to apply different types(or levels of styles or style specification formats) – inline, internal, external styles to HTML elements(identify selector, property and value).

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>CSS Styling Examples</title>
   <!-- Link to external CSS file -->
   <link rel="stylesheet" href="styles.css">
   <!-- Internal CSS -->
   <style>
      .internal-style {
         color: blue;
         font-size: 20px;
      }
   </style>
</head>
<body>
   <h1 style="color: red; font-size: 24px;">Inline Style</h1>
   <p class="internal-style">This paragraph uses internal styles.</p>
   <div class="external-style">This div uses external styles.</div>
</body>
</html>
```

External css style.css

```
.external-style {
   color: green;
   font-size: 18px;
```

}

4. Selector forms

4.a) Write a program to apply different types of selectors forms.

i) Simple selector (element, id, class, group, universal)

ii) combinatory selector(descendent, child, adjacent sibling, general sibling)

iii) Pseudo class selector

iv) Pseudo element selector

v) Attribute selector.

4.a) Write a program to apply different types of selectors forms.

i) Simple selector (element, id, class, group, universal)

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>CSS Selectors Examples</title>
   <!-- Internal CSS to demonstrate various selectors -->
   <style>
      /* Element Selector */
      p {
         color: blue;
         font-size: 18px;
      }
      /* ID Selector */
      #unique-element {
         background-color: lightgray;
         padding: 10px;
```

```
        }
        /* Class Selector */
        .highlight {
            color: red;
            font-weight: bold;
        }
        /* Group Selector */
        h1, h2, h3 {
            color: green;
        }
        /* Universal Selector */
        * {
            margin: 10px;
            padding: 5px;
        }
    </style>
</head>
<body>
    <!-- Element Selector Example -->
    <p>This paragraph is styled using an element selector.</p>
    <!-- ID Selector Example -->
    <div id="unique-element">This div is styled using an ID selector.</div>
    <!-- Class Selector Example -->
    <span class="highlight">This text is styled using a class selector.</span>
    <!-- Group Selector Example -->
    <h1>This is a heading 1 styled using a group selector.</h1>
    <h2>This is a heading 2 styled using a group selector.</h2>
    <h3>This is a heading 3 styled using a group selector.</h3>
    <!-- Universal Selector Example -->
```

** All elements have margin and padding set by the universal selector. **

** All elements have margin and padding set by the universal selector. **

</body>

</html>

How to Run

1. Create a file named index.html and type the aboveHTML code into it.
2. Open the index.html file in a web browser to see the different styles applied to the elements.

4.ii)combinatory selector(descendent, child, adjacent sibling, general sibling)

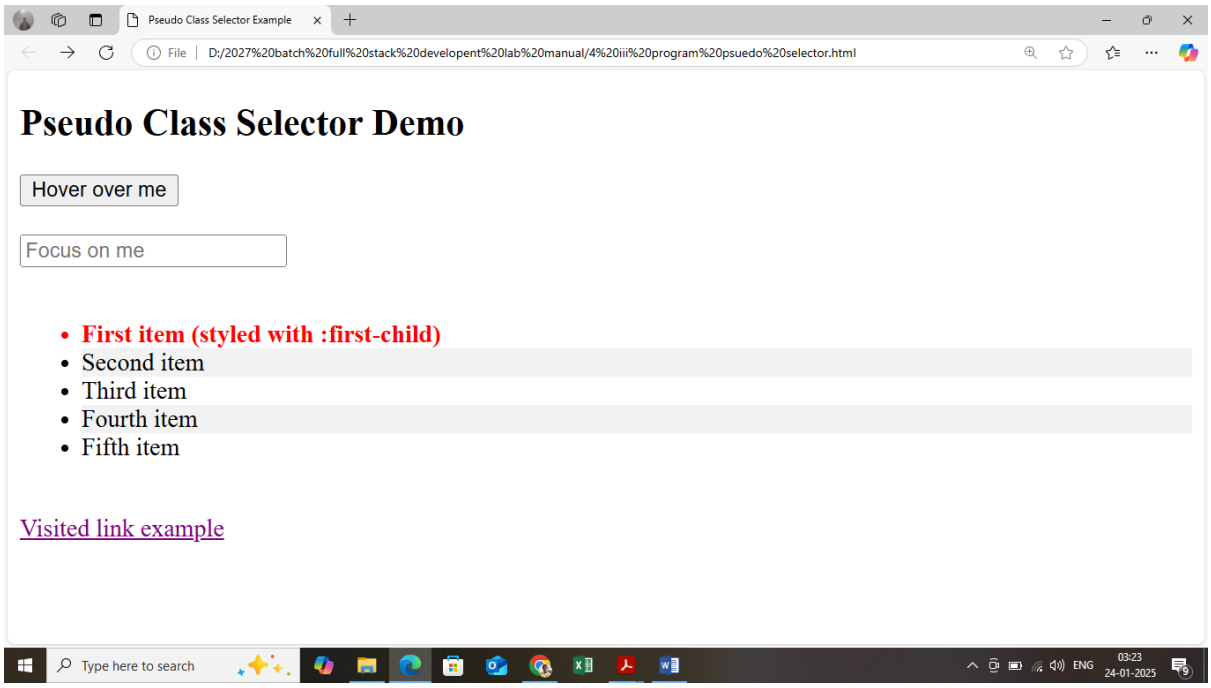
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Combinatory Selectors Examples</title>
    <!-- Internal CSS to demonstrate various combinatory selectors -->
    <style>
        /* Descendant Selector */
        .container p {
            color: blue;
        }
        /* Child Selector */
        .container > p {
            font-weight: bold;
        }
        /* Adjacent Sibling Selector */
        h2 + p {
            color: green;
```

```
        }
        /* General Sibling Selector */
        h2 ~ p {
            font-size: 14px;
        }
    </style>
</head>
<body>
    <div class="container">
        <p>This paragraph is a descendant of .container and is styled using the descendant selector.</p>
        <div>
            <p>This paragraph is a descendant of .container but not a direct child, so only the color is applied.</p>
        </div>
        <p>This paragraph is a direct child of .container and is styled using both the descendant and child selectors.</p>
    </div>
    <h2>This heading is an adjacent sibling example.</h2>
    <p>This paragraph is an adjacent sibling to the h2 and is styled using the adjacent sibling selector.</p>
    <p>This paragraph is also a sibling of the h2 but is not immediately adjacent, so only the general sibling selector applies.</p>
</body>
</html>
```

4. iii) Pseudo class selector

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Pseudo Class Selector Example</title>
<style>
/* Change background color when hovering over a button */
        button:hover {
           background-color: lightblue;
           color: white;
        }

/* Change border color when an input field is focused */
        input:focus {
           border: 2px solid green;
        }
        /* Style the first child of a list */
        ul li:first-child {
           font-weight: bold;
           color: red;
        }
        /* Style every even child in the list */
        ul li:nth-child(even) {
           background-color: #f2f2f2;
        }
        /* Change color when a link is visited */
        a:visited {
           color: purple;
```

```
            }
        </style>
    </head>
    <body>
        <h2>Pseudo Class Selector Demo</h2>
        <button>Hover over me</button>
        <br><br>
<input type="text" placeholder="Focus on me">
        <br><br>
         <ul>
            <li>First item (styled with :first-child)</li>
            <li>Second item</li>
            <li>Third item</li>
            <li>Fourth item</li>
            <li>Fifth item</li>
        </ul>
        <br>
<a href="https://google.com">Visited link example</a>
    </body>
    </html>
```

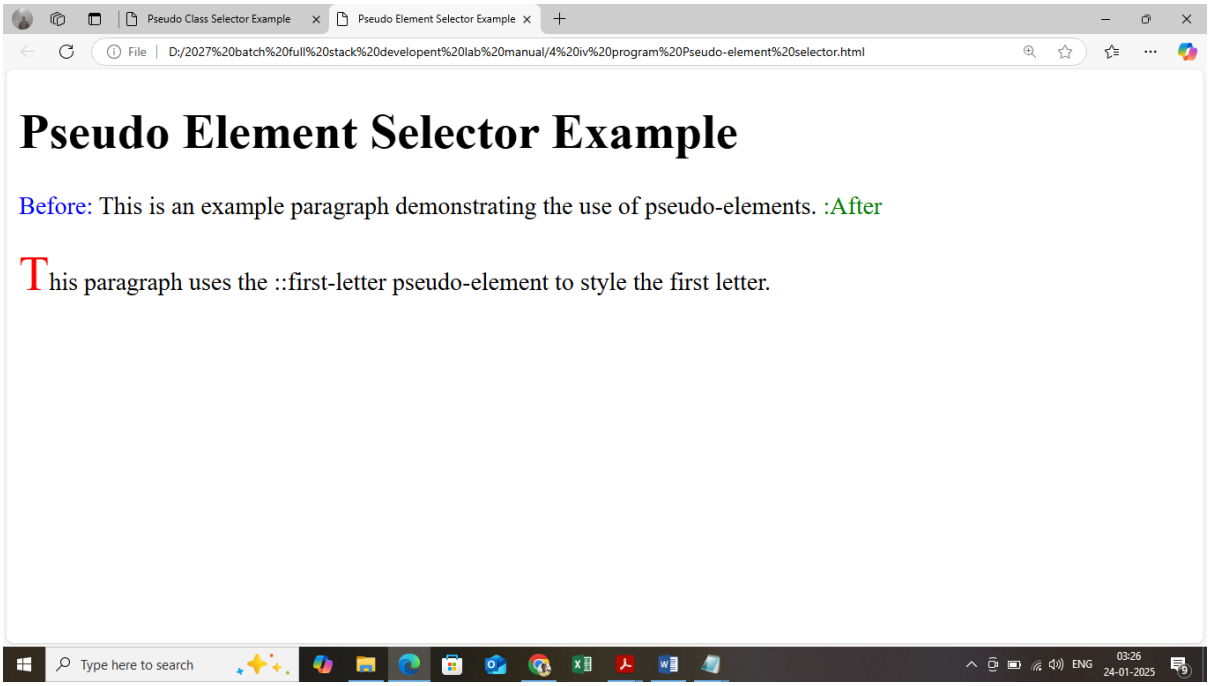


4.iv) Pseudo element selector

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pseudo Element Selector Example</title>
    <style>
        .example::before {
            content: "Before: ";
            color: blue;
        }
        .example::after {
            content: " :After";
            color: green;
        }
        .highlight::first-letter {
            font-size: 2em;
            color: red;
```

```
    }
  </style>
</head>
<body>
  <h1>Pseudo Element Selector Example</h1>
  <p class="example">This is an example paragraph demonstrating the use of pseudo-elements.</p>
  <p class="highlight">This paragraph uses the ::first-letter pseudo-element to style the first letter.</p>
</body>
</html>
```

- The ::before pseudo-element is used to insert content before the content of the selected element. In this case, it adds "Before: " before the paragraph with the class example.
- The ::after pseudo-element is used to insert content after the content of the selected element. In this case, it adds " :After" after the paragraph with the class example.
- The ::first-letter pseudo-element is used to style the first letter of the selected element. Here, it enlarges and changes the color of the first letter of the paragraph with the class highlight.



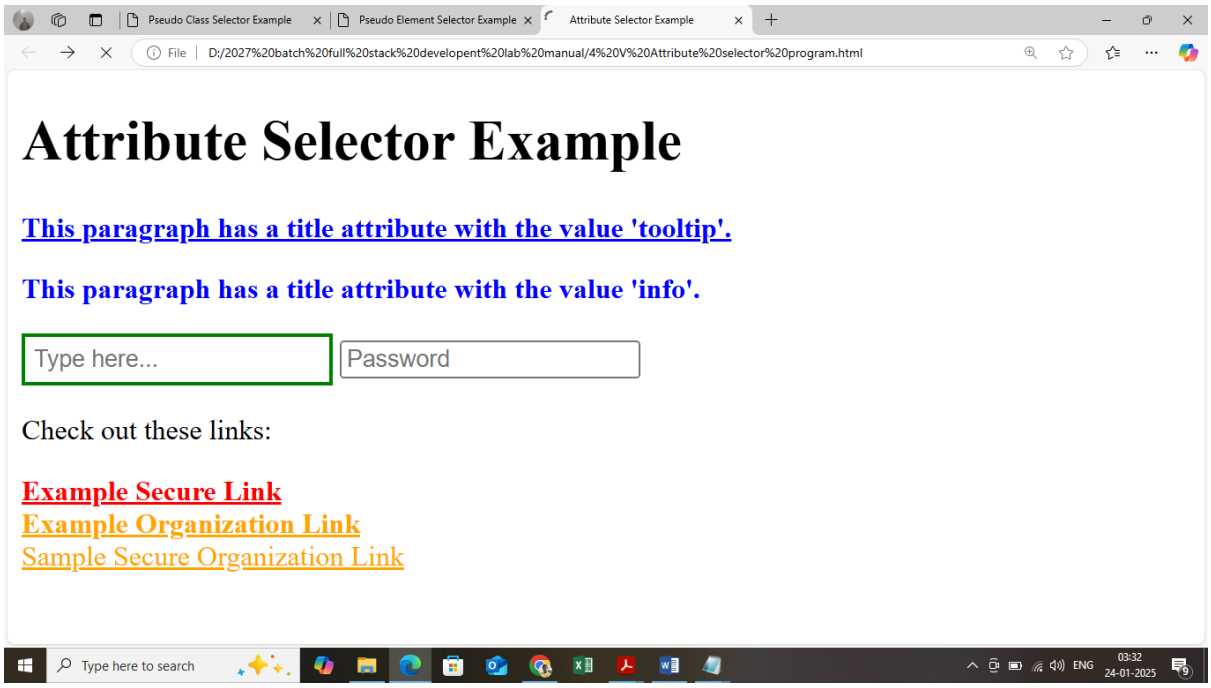
4.v) Attribute selector.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Attribute Selector Example</title>
   <style>
      /* Selects all elements with a title attribute */
      [title] {
         color: blue;
         font-weight: bold;
      }
      /* Selects all elements with a title attribute equal to 'tooltip' */
      [title="tooltip"] {
         text-decoration: underline;
      }
      /* Selects all input elements with a type attribute equal to 'text' */
      input[type="text"] {
         border: 2px solid green;
         padding: 5px;
      }
      /* Selects all links (a elements) where the href attribute starts with 'https' */
      a[href^="https"] {
         color: red;
      }
      /* Selects all links (a elements) where the href attribute ends with '.org' */
      a[href$=".org"] {
```

```
            color: orange;
        }
        /* Selects all links (a elements) where the href attribute contains 'example' */
        a[href*="example"] {
            font-weight: bold;
        }
    </style>
</head>
<body>
    <h1>Attribute Selector Example</h1>
    <p title="tooltip">This paragraph has a title attribute with the value 'tooltip'.</p>
    <p title="info">This paragraph has a title attribute with the value 'info'.</p>
<input type="text" placeholder="Type here...">
<input type="password" placeholder="Password">
    <p>Check out these links:</p>
    <a href="https://www.example.com">Example Secure Link</a><br>
    <a href="http://www.example.org">Example Organization Link</a><br>
    <a href="https://www.sample.org">Sample Secure Organization Link</a>
</body>
</html>
```

- The [title] selector styles all elements that have a title attribute, making their text color blue and bold.
- The [title="tooltip"] selector styles elements with a title attribute exactly equal to tooltip, adding an underline.
- The input[type="text"] selector styles input elements with a type attribute equal to text, adding a green border and padding.
- The a[href^="https"] selector styles links (a elements) where the href attribute starts with https, changing their text color to red.

 The a[href$=".org"] selector styles links where the href attribute ends with .org, changing their text color to orange.
 The a[href*="example"] selector styles links where the href attribute contains the word example, making their text bold.



5.CSS with Color, Background, Font, Text, and CSS Box model.

5.a) Write a program to demonstrate the various ways you can reference a color in CSS.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Color Reference</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        .color-box {
            width: 150px;
```

```
            height: 150px;
            margin: 10px;
            display: inline-block;
            color: white;
            text-align: center;
            line-height: 150px;
            font-weight: bold;
            border-radius: 8px;
        }
        .color-name {
            background-color: red;
        }
        .hex {
            background-color: #00FF00;
        }
        .rgb {
            background-color: rgb(0, 0, 255);
        }
        .rgba {
            background-color: rgba(255, 255, 0, 0.5);
            color: black;
        }
        .hsl {
            background-color: hsl(240, 100%, 50%);
        }
        .hsla {
            background-color: hsla(120, 100%, 50%, 0.3);
        }
    </style>
</head>
```

<body>

<h1>CSS Color Reference Demonstration</h1>

** red **

** #00FF00 **

** rgb(0, 0, 255) **

** rgba(255, 255, 0, 0.5) **

** hsl(240, 100%, 50%) **

** hsla(120, 100%, 50%, 0.3) **

</body>

</html>

Explanation:

- **Color Name (red)**: This is a predefined color name that browsers recognize.
- **Hex (#00FF00)**: This is a hexadecimal color code.
- **RGB (rgb(0, 0, 255))**: This uses the RGB color model where each parameter (red, green, blue) defines the intensity of the color as an integer between 0 and 255.
- **RGBA (rgba(255, 255, 0, 0.5))**: This is similar to RGB but includes an alpha parameter for transparency, ranging from 0 (fully transparent) to 1 (fully opaque).
- **HSL (hsl(240, 100%, 50%))**: This uses the HSL color model which stands for hue, saturation, and lightness.
- **HSLA (hsla(120, 100%, 50%, 0.3))**: This is similar to HSL but includes an alpha parameter for transparency.

When you open this HTML file in a browser, you will see a demonstration of different ways to reference colors in CSS. Each colored box represents a different method of specifying colors.

5.b) Write a CSS rule that places a background image halfway down the page, titling it horizontally. The image should remain in place when the user scrolls up or down.

```
body {
   /* Set the background image */
   background-image: url('path/to/your/image.jpg');
/* Place the image halfway down the page and tiling it horizontally */
   background-position: center 50%;
/* Ensure the image remains in place when scrolling */
   background-attachment: fixed;
 /* Tile the image horizontally */
   background-repeat: repeat-x;
 /* Optionally, set a background color as a fallback */
   background-color: #f0f0f0;
}
```

Explanation:

- **background-image: url('path/to/your/image.jpg');**: Specifies the background image to be used.
- **background-position: center 50%;**: Positions the background image in the center horizontally and halfway down the page vertically.
- **background-attachment: fixed;**: Fixes the background image in place so that it doesn't move when the user scrolls.
- **background-repeat: repeat-x;**: Tiles the background image horizontally.
- **background-color: #f0f0f0;**: Sets a fallback background color in case the image is not available.

You can add this CSS rule to your existing CSS file or within a <style> block in your HTML document. Just make sure to replace 'path/to/your/image.jpg' with the actual path to your background image.

5.c) Write a program using the following ites related to CSS font and text

i) font-size

ii) font-weight

iii) font-style

iv) text-decoration

v) text-transformation

vi) text – alignment

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Font and Text Properties</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        .example {
            margin: 20px 0;
        }
        .font-size {
            font-size: 24px; /* Adjust the font size */
        }
        .font-weight {
            font-weight: bold; /* Set the font weight */
        }
        .font-style {
            font-style: italic; /* Set the font style */
        }
```

```
    .text-decoration {
        text-decoration: underline; /* Set the text decoration */
    }
    .text-transform {
        text-transform: uppercase; /* Transform the text to uppercase */
    }
    .text-align {
        text-align: center; /* Align the text to the center */
    }
  </style>
</head>
<body>
  <h1>CSS Font and Text Properties Demonstration</h1>
  <div class="example font-size">This text has a font size of 24px.</div>
  <div class="example font-weight">This text is bold.</div>
  <div class="example font-style">This text is italic.</div>
  <div class="example text-decoration">This text is underlined.</div>
  <div class="example text-transform">This text is uppercase.</div>
  <div class="example text-align">This text is centered.</div>
</body>
</html>
```

Explanation:

- **font-size**: Sets the size of the font. In the example, it's set to 24px.
- **font-weight**: Sets the weight (or boldness) of the font. In the example, it's set to bold.
- **font-style**: Sets the style of the font, such as italic or normal. In the example, it's set to italic.
- **text-decoration**: Adds decorations to the text, such as underline, overline, or line-through. In the example, it's set to underline.
- **text-transform**: Controls the capitalization of the text. It can be set to uppercase, lowercase, or capitalize. In the example, it's set to uppercase.
- **text-align**: Sets the horizontal alignment of the text. It can be left, right, center, or justify. In the example, it's set to center.

This HTML and CSS code demonstrates how to apply these properties to different text elements on a web page. You can open this HTML file in a web browser to see the effects of each CSS property.

5.d) write a program to explain the importance of CSS Box model using

i. content ii.border iii. Margin iv. Padding

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>CSS Box Model Demonstration</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         margin: 20px;
      }
      .box {
         width: 200px; /* Width of the content */
         margin: 20px; /* Margin around the element */
         padding: 15px; /* Padding inside the element */
         border: 5px solid #333; /* Border around the element */
         background-color: #f0f0f0; /* Background color for visibility */
      }
      .box .content {
         background-color: #fff; /* Background color for the content area */
         padding: 10px; /* Padding inside the content area */
      }
   </style>
</head>
<body>
```

```
    <h1>CSS Box Model Demonstration</h1>
    <div class="box">
        <div class="content">
            <p>This is the content area.</p>
            <p>The CSS Box Model includes:</p>
            <ul>
                <li><strong>Content</strong>: The actual content of the element (text, images, etc.)</li>
                <li><strong>Padding</strong>: Space between the content and the border</li>
                <li><strong>Border</strong>: The border surrounding the padding (and content)</li>
                <li><strong>Margin</strong>: Space outside the border, separating the element from others</li>
            </ul>
        </div>
    </div>
</body>
</html>
```

Explanation:

1. **Content**: The content is the innermost part of the box, which holds the actual text, images, or other media. In the example, the content is contained within the element.
2. **Padding**: Padding is the space between the content and the border. It creates inner space within the element. In the example, padding: 15px; creates space between the content and the border of the .box element, and padding: 10px; creates additional space inside the content area.
3. **Border**: The border surrounds the padding and content. It can have different styles, widths, and colors. In the example, border: 5px solid #333; creates a solid black border around the .box element.
4. **Margin**: The margin is the outermost part of the box, creating space between the element and other elements. In the example, margin: 20px; creates space outside the border of the .box element, separating it from other elements on the page.

How to view:

To see the CSS Box Model in action, open this HTML file in a web browser. You will see a box with clearly defined content, padding, border, and margin, illustrating how each part of the box model affects the layout and spacing of the element.

6.Applying JavaScript – Internal and external I/O type conversion.

6.a) write a program to embed internal and external javascript in a web page.

script.js

```
// script.js
function showAlertExternal() {
    alert('This is an external JavaScript alert!');
}
```

HTML FILE

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Embedding Internal and External JavaScript</title>
    <script src="script.js"></script> <!-- Link to external JavaScript file -->
    <script>
        // Internal JavaScript
        function showAlertInternal() {
            alert('This is an internal JavaScript alert!');
        }
    </script>
</head>
<body>
    <h1>Embedding Internal and External JavaScript</h1>
    <button onclick="showAlertInternal()">Internal JavaScript Alert</button>
```

<button onclick="showAlertExternal()">External JavaScript Alert</button>

</body>

</html>

Explanation:

1. **External JavaScript**:
    - The external JavaScript file script.js contains a function showAlertExternal that displays an alert when called.
    - The HTML file links to this external JavaScript file using the <script src="script.js"></script> tag in the <head> section.
2. **Internal JavaScript**:
    - The internal JavaScript is included directly within the HTML file inside a <script> tag in the <head> section.
    - The internal JavaScript contains a function showAlertInternal that displays an alert when called.
3. **HTML Content**:
    - The HTML body contains two buttons. Each button has an onclick attribute that calls one of the JavaScript functions.
    - The first button calls the showAlertInternal function, demonstrating the use of internal JavaScript.
    - The second button calls the showAlertExternal function, demonstrating the use of external JavaScript.

How to Run:

1. Save the external JavaScript code in a file named script.js.
2. Save the HTML code in a file named index.html.
3. Open index.html in a web browser.
4. Click the buttons to see alerts triggered by the internal and external JavaScript functions.

6.b) write a program to explain the different ways for displaying output.

```
<html>
  <body>
    <h1>Output</h1>
    <p id="output"></p>
    <script src="script.js"></script>
  </body>
</html>
```

Script.js

```
// Method 1: Using alert() function
alert("Hello, World!");
// Method 2: Using document.write() function
document.write("Hello, World! <br>");
// Method 3: Using console.log() function
console.log("Hello, World!");
// Method 4: Using innerHTML property
let output = document.getElementById("output");
output.innerHTML = "Hello, World!";
// Method 5: Using prompt() function
let name = prompt("What is your name?");
document.write("Hello, " + name + "!");
// Method 6: Using confirm() function
let response = confirm("Are you sure?");
if (response) {
  document.write("You clicked OK!");
} else {
  document.write("You clicked Cancel!");
}
```

1. alert() function: Displays a pop-up alert box with the specified message.

2. document.write() function: Writes the specified message to the HTML document.

3. console.log() function: Writes the specified message to the browser's console.

4. innerHTML property: Sets the HTML content of an element with the specified ID.

5. prompt() function: Displays a pop-up prompt box that asks the user for input.

6. confirm() function: Displays a pop-up confirmation box that asks the user to confirm or cancel.

6.c) write a program to explain the different ways for taking input.

```
<html>
  <body>
    <form id="myForm">
      <input type="text" id="myInput" />
      <button type="submit">Submit</button>
    </form>
    <script src="script1.js"></script>
  </body>
</html>
```

Script1.js:

```
// Method 1: Using prompt() function
let name = prompt("What is your name?");
document.writeln("Hello, " + name + "!");
// Method 2: Using confirm() function
let response = confirm("Are you sure?");
if (response) {
  document.writeln("You clicked OK!");
} else {
  document.writeln("You clicked Cancel!");
}
// Method 3: Using HTML form and JavaScript
let form = document.getElementById("myForm");
let inputField = document.getElementById("myInput");
form.addEventListener("submit", function(event) {
  event.preventDefault();
  let inputValue = inputField.value;
  document.writeln("You entered: " + inputValue);
});
```

```
// Method 4: Using JavaScript readline() function (Node.js)
// Note: This method only works in Node.js environment
const readline = require("readline");
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});
rl.question("What is your name? ", function(answer) {
  console.log("Hello, " + answer + "!");
  rl.close();
});
```

1. prompt() function: Displays a pop-up prompt box that asks the user for input.

2. confirm() function: Displays a pop-up confirmation box that asks the user to confirm or cancel.

3. HTML form and JavaScript: Uses an HTML form to collect user input, and JavaScript to process the input.

4. readline() function (Node.js): Uses the readline module in Node.js to read user input from the console.

6.d) create a webpage which uses prompt dialogue box to ask a voter for his name and age . display the information in table format along with either the voter can vote or not.

```
<!DOCTYPE html>
<html>
<head>
  <title>Voter Information</title>
  <style>
    table {
      border-collapse: collapse;
    }
    th, td {
      border: 1px solid black;
```

```
    padding: 10px;
   }
  </style>
</head>
<body>
  <h2>Voter Information</h2>
  <button onclick="getVoterInfo()">Get Voter Info</button>
  <div id="voterInfo"></div>

  <script>
   function getVoterInfo() {
    let name = prompt("Please enter your name:");
    let age = parseInt(prompt("Please enter your age:"));
    let votingEligibility = (age >= 18) ? "Eligible" : "Not Eligible";
    let voterInfoHtml = `
      <table>
        <tr>
          <th>Name</th>
          <th>Age</th>
          <th>Voting Eligibility</th>
        </tr>
        <tr>
          <td>${name}</td>
          <td>${age}</td>
          <td>${votingEligibility}</td>
        </tr>
      </table>
    `;   document.getElementById("voterInfo").innerHTML =
voterInfoHtml;
   }
  </script> </body> </html>
```

7. JavaScript Pre-defined and User-defined objects

7 a) Write a program using document object properties and methods

Here is a complete JavaScript program and corresponding HTML file that uses various properties and methods of the document object:

HTML File (index.html)

```
<!DOCTYPE html>
<html>
<head>
    <title>Document Object Properties and Methods</title>
</head>
<body>
    <h1 id="heading">Document Object Properties and Methods</h1>
    <p id="paragraph">This is a paragraph of text.</p>
    <button onclick="changeText()">Change Text</button>
    <button onclick="changeColor()">Change Color</button>
    <button onclick="createElement()">Create Element</button>
    <button onclick="removeElement()">Remove Element</button>
    <script src="7a.js"></script>
</body>
</html>
```

JavaScript File (7a.js)

```
// Get the title of the document
console.log("Document Title: " + document.title);

// Get the URL of the document
console.log("Document URL: " + document.URL);
// Get the domain name of the document
console.log("Document Domain: " + document.domain);
// Get the last modified date of the document
console.log("Document Last Modified: " + document.lastModified);
```

```
// Function to change the text of the paragraph
function changeText() {
    let paragraph = document.getElementById("paragraph");
    paragraph.textContent = "This is the new text.";
}
// Function to change the color of the heading
function changeColor() {
    let heading = document.getElementById("heading");
    heading.style.color = "red";
}
// Function to create a new element
function createElement() {
    let newElement = document.createElement("p");
    newElement.textContent = "This is a new paragraph.";
    document.body.appendChild(newElement);
}
// Function to remove an element
function removeElement() {
    let paragraph = document.getElementById("paragraph");
    paragraph.remove();
}
```

This program uses the following properties and methods of the document object:

- title: Gets the title of the document.

- URL: Gets the URL of the document.

- domain: Gets the domain name of the document.

- lastModified: Gets the last modified date of the document.

- getElementById(): Gets an element by its ID.

- createElement(): Creates a new element.

- appendChild(): Adds a new child element to the document body.

- remove(): Removes an element from the document.

7 b) write a program using window object properties and methods

HTML File (7b.html)

```
<!DOCTYPE html>
<html>
<head>
    <title>Window Object Properties and Methods</title>
</head>
<body>
    <h1 id="heading">Window Object Properties and Methods</h1>
    <button onclick="openWindow()">Open Window</button>
    <button onclick="closeWindow()">Close Window</button>
    <button onclick="moveWindow()">Move Window</button>
    <button onclick="resizeWindow()">Resize Window</button>
    <button onclick="alertMessage()">Alert Message</button>
    <button onclick="confirmMessage()">Confirm Message</button>
    <button onclick="promptMessage()">Prompt Message</button>
    <script src="7b.js"></script>
</body>  </html>
```

JavaScript File (script.js)

```
// Get the width and height of the window
console.log("Window Width: " + window.innerWidth);
console.log("Window Height: " + window.innerHeight);
// Function to open a new window
function openWindow() {
    let newWindow = window.open("", "newWindow",
"width=400,height=200");
    newWindow.document.write("<h1>Hello, World!</h1>");
}
// Function to close the current window
function closeWindow() {
    window.close();
```

```
}
// Function to move the current window
function moveWindow() {
        window.moveBy(100, 100);
}
// Function to resize the current window
function resizeWindow() {
        window.resizeBy(100, 100);
}
// Function to display an alert message
function alertMessage() {
        window.alert("Hello, World!");
}
// Function to display a confirm message
function confirmMessage() {
        let response = window.confirm("Are you sure?");
        if (response) {
                console.log("You clicked OK!");
        } else {
                console.log("You clicked Cancel!");
        }
}
// Function to display a prompt message
function promptMessage() {
        let name = window.prompt("What is your name?");
        if (name) {
                console.log("Hello, " + name + "!");
        } else {
                console.log("You did not enter your name.");
        } }
```

This program uses the following properties and methods of the window object:

- innerWidth and innerHeight: Get the width and height of the window.

- open(): Opens a new window.

- close(): Closes the current window.

- moveBy(): Moves the current window by a specified amount.

- resizeBy(): Resizes the current window by a specified amount.

- alert(): Displays an alert message.

- confirm(): Displays a confirm message.

- prompt(): Displays a prompt message.

7 c) write a program using array object properties and methods

Here is a JavaScript program and corresponding HTML file that uses various properties and methods of the Array object:

HTML File (7c.html)

```
<!DOCTYPE html>
<html>
<head>
    <title>Array Object Properties and Methods</title>
</head>
<body>
    <h1 id="heading">Array Object Properties and Methods</h1>
    <button onclick="createArray()">Create Array</button>
    <button onclick="pushElement()">Push Element</button>
    <button onclick="popElement()">Pop Element</button>
    <button onclick="shiftElement()">Shift Element</button>
    <button onclick="unshiftElement()">Unshift Element</button>
    <button onclick="spliceElement()">Splice Element</button>
    <button onclick="sliceArray()">Slice Array</button>
    <button onclick="concatArray()">Concat Array</button>
    <button onclick="joinArray()">Join Array</button>
    <button onclick="reverseArray()">Reverse Array</button>
    <button onclick="sortArray()">Sort Array</button>
    <p id="output"></p>
    <script src="7c.js"></script>
</body>
</html>
```

JavaScript File (7c.js)

```
let myArray = [];
let output = document.getElementById("output");
// Function to create an array
function createArray() {
    myArray = [1, 2, 3, 4, 5];
    output.textContent = "Array created: " + myArray;
}
// Function to push an element to the array
function pushElement() {
    myArray.push(6);
    output.textContent = "Element pushed: " + myArray;
}
// Function to pop an element from the array
function popElement() {
    myArray.pop();
    output.textContent = "Element popped: " + myArray;
}
// Function to shift an element from the array
function shiftElement() {
    myArray.shift();
    output.textContent = "Element shifted: " + myArray;
}
// Function to unshift an element to the array
function unshiftElement() {
    myArray.unshift(0);
    output.textContent = "Element unshifted: " + myArray;
}
// Function to splice an element from the array
function spliceElement() {
```

```
    myArray.splice(2, 1);
    output.textContent = "Element spliced: " + myArray;
}
// Function to slice the array
function sliceArray() {
    let slicedArray = myArray.slice(1, 3);
    output.textContent = "Array sliced: " + slicedArray;
}
// Function to concat the array
function concatArray() {
    let concatArray = myArray.concat([6, 7, 8]);
    output.textContent = "Array concatenated: " + concatArray;
}
// Function to join the array
function joinArray() {
    let joinedArray = myArray.join("-");
    output.textContent = "Array joined: " + joinedArray;
}
// Function to reverse the array
function reverseArray() {
    myArray.reverse();
    output.textContent = "Array reversed: " + myArray;
}
// Function to sort the array
function sortArray() {
    myArray.sort();
    output.textContent = "Array sorted: " + myArray;
}
```

This program uses the following properties and methods of the Array object:

- push(): Adds one or more elements to the end of the array.

- pop(): Removes the last element from the array.

- shift(): Removes the first element from the array.

- unshift(): Adds one or more elements to the beginning of the array.

- splice(): Adds or removes elements from the array.

- slice(): Returns a shallow copy of a portion of the array.

- concat(): Returns a new array that contains the elements of the original array and the elements of the arrays or values provided.

- join(): Returns a string that contains the elements of the array, separated by a specified separator.

- reverse(): Reverses the order of the elements in the array.

- sort(): Sorts the elements in the array in place and returns the sorted array.

7 d) write a program using math object properties and methods

HTML File (7d.html)

```
<!DOCTYPE html>
<html>
<head>
    <title>Math Object Properties and Methods</title>
</head>
<body>
    <h1 id="heading">Math Object Properties and Methods</h1>
    <button onclick="absValue()">Absolute Value</button>
    <button onclick="ceilValue()">Ceiling Value</button>
    <button onclick="floorValue()">Floor Value</button>
    <button onclick="maxValue()">Maximum Value</button>
    <button onclick="minValue()">Minimum Value</button>
    <button onclick="powValue()">Power Value</button>
    <button onclick="randomValue()">Random Value</button>
    <button onclick="roundValue()">Round Value</button>
    <button onclick="sqrtValue()">Square Root Value</button>
    <p id="output"></p>
    <script src="7d.js"></script>
</body>
</html>
```

JavaScript File (7d.js)

```
let output = document.getElementById("output");
// Function to calculate absolute value
function absValue() {
    let num = 10;
    output.textContent = "Absolute value of " + num + " is: " +
Math.abs(num);
}
```

```
// Function to calculate ceiling value
function ceilValue() {
    let num = 10.5;
    output.textContent = "Ceiling value of " + num + " is: " + Math.ceil(num);
}
// Function to calculate floor value
function floorValue() {
    let num = 10.5;
    output.textContent = "Floor value of " + num + " is: " + Math.floor(num);
}
// Function to calculate maximum value
function maxValue() {
    let num1 = 10;
    let num2 = 20;
    output.textContent = "Maximum value of " + num1 + " and " + num2 + " is: " + Math.max(num1, num2);
}
// Function to calculate minimum value
function minValue() {
    let num1 = 10;
    let num2 = 20;
    output.textContent = "Minimum value of " + num1 + " and " + num2 + " is: " + Math.min(num1, num2);
}
// Function to calculate power value
function powValue() {
    let base = 2;
    let exponent = 3;
    output.textContent = "Power value of " + base + " raised to " + exponent + " is: " + Math.pow(base, exponent);
```

```
}
// Function to generate random value
function randomValue() {
    output.textContent = "Random value is: " + Math.random();
}
// Function to calculate round value
function roundValue() {
    let num = 10.5;
    output.textContent = "Round value of " + num + " is: " +
Math.round(num);
}
// Function to calculate square root value
function sqrtValue() {
    let num = 16;
    output.textContent = "Square root value of " + num + " is: " +
Math.sqrt(num);
}
```

This program uses the following properties and methods of the Math object:

- abs(): Returns the absolute value of a number.

- ceil(): Returns the smallest integer greater than or equal to a number.

- floor(): Returns the largest integer less than or equal to a number.

- max(): Returns the maximum value of a set of numbers.

- min(): Returns the minimum value of a set of numbers.

- pow(): Returns the value of a number raised to a power.

- random(): Returns a random number between 0 and 1.

- round(): Returns the value of a number rounded to the nearest integer.

- sqrt(): Returns the square root of a number.

7 e) write a program using string object properties and methods

HTML File (7e.html)

```
<!DOCTYPE html>
<html>
<head>
    <title>String Object Properties and Methods</title>
</head>
<body>
    <h1 id="heading">String Object Properties and Methods</h1>
    <button onclick="charCodeAtMethod()">charCodeAt Method</button>
    <button onclick="charAtMethod()">charAt Method</button>
    <button onclick="concatMethod()">concat Method</button>
    <button onclick="indexOfMethod()">indexOf Method</button>
    <button onclick="lastIndexOfMethod()">lastIndexOf Method</button>
    <button onclick="matchMethod()">match Method</button>
    <button onclick="replaceMethod()">replace Method</button>
    <button onclick="searchMethod()">search Method</button>
    <button onclick="sliceMethod()">slice Method</button>
    <button onclick="splitMethod()">split Method</button>
    <button onclick="substringMethod()">substring Method</button>
    <button onclick="toLowerCaseMethod()">toLowerCase Method</button>
    <button onclick="toUpperCaseMethod()">toUpperCase Method</button>
    <p id="output"></p>
    <script src="7e.js"></script>
</body>
</html>
```

JavaScript File (7e.js)

```
let output = document.getElementById("output");
// Function to demonstrate charCodeAt method
function charCodeAtMethod() {
        let str = "Hello, World!";
        let charCode = str.charCodeAt(0);
        output.textContent = "The character code of the first character is: "
+ charCode;
}
// Function to demonstrate charAt method
function charAtMethod() {
        let str = "Hello, World!";
        let char = str.charAt(0);
        output.textContent = "The first character is: " + char;
}
// Function to demonstrate concat method
function concatMethod() {
        let str1 = "Hello, ";
        let str2 = "World!";
        let result = str1.concat(str2);
        output.textContent = "The concatenated string is: " + result;
}
// Function to demonstrate indexOf method
function indexOfMethod() {
        let str = "Hello, World!";
        let index = str.indexOf("World");
        output.textContent = "The index of 'World' is: " + index;
}
// Function to demonstrate lastIndexOf method
function lastIndexOfMethod() {
        let str = "Hello, World!";
```

```
    let index = str.lastIndexOf("World");
    output.textContent = "The last index of 'World' is: " + index;
}
// Function to demonstrate match method
function matchMethod() {
    let str = "Hello, World!";
    let regex = /World/;
    let result = str.match(regex);
    output.textContent = "The match is: " + result;
}
// Function to demonstrate replace method
function replaceMethod() {
    let str = "Hello, World!";
    let regex = /World/;
    let replacement = "Universe";
    let result = str.replace(regex, replacement);
    output.textContent = "The replaced string is: " + result;
}
// Function to demonstrate search method
function searchMethod() {
    let str = "Hello, World!";
    let regex = /World/;
    let result = str.search(regex);
    output.textContent = "The index of the match is: " + result;
}
// Function to demonstrate slice method
function sliceMethod() {
    let str = "Hello, World!";
    let start = 7;
    let end = 12;
```

```
    let result = str.slice(start, end);
    output.textContent = "The sliced string is: " + result;
}
// Function to demonstrate split method
function splitMethod() {
    let str = "apple,banana,cherry";
    let separator = ",";
    let result = str.split(separator);
    output.textContent = "The split array is: " + result;
}
// Function to demonstrate substring method
function substringMethod() {
    let str = "Hello, World!";
    let start = 7;
    let end = 12;
    let result = str.substring(start, end);
    output.textContent = "The substring is: " + result;
}
// Function to demonstrate toLowerCase method
function toLowerCaseMethod() {
    let str = "HELLO, WORLD!";
    let result = str.toLowerCase();
    output.textContent = "The string in lowercase is: " + result;
}
// Function to demonstrate toUpperCase method
function toUpperCaseMethod() {
    let str = "hello, world!";
    let result = str.toUpperCase();
    output.textContent = "The string in uppercase is: " + result;
}
```

This program uses the following properties and methods of the String object:

- charCodeAt(): Returns the Unicode value of the character at the specified index.

- charAt(): Returns the character at the specified index.

- concat(): Concatenates two or more strings.

- indexOf(): Returns the index of the first occurrence of the specified value.

- lastIndexOf(): Returns the index of the last occurrence of the specified value.

- match(): Returns an array containing the matches of the specified regular expression.

- replace(): Replaces the

7 f) write a program using regex object properties and methods

HTML File (7f.html)

```
<!DOCTYPE html>
<html>
<head>
    <title>RegExp Object Properties and Methods</title>
</head>
<body>
    <h1 id="heading">RegExp Object Properties and Methods</h1>
    <button onclick="testMethod()">Test Method</button>
    <button onclick="execMethod()">Exec Method</button>
    <button onclick="getSource()">Get Source</button>
    <button onclick="getFlags()">Get Flags</button>
    <button onclick="getLastIndex()">Get Last Index</button>
    <button onclick="setLastIndex()">Set Last Index</button>
    <p id="output"></p>
    <script src="7f.js"></script>
</body>
</html>
```

JavaScript File (7f.js)

```
let output = document.getElementById("output");
let regex = /hello/i;
// Function to test the regex pattern
function testMethod() {
    let result = regex.test("Hello, World!");
    output.textContent = "Test result: " + result;
}
// Function to execute the regex pattern
function execMethod() {
    let result = regex.exec("Hello, World!");
```

```
    output.textContent = "Exec result: " + result;
}
// Function to get the source property of the regex object
function getSource() {
    output.textContent = "Source property: " + regex.source;
}
// Function to get the flags property of the regex object
function getFlags() {
    output.textContent = "Flags property: " + regex.flags;
}
// Function to get the lastIndex property of the regex object
function getLastIndex() {
    output.textContent = "LastIndex property: " + regex.lastIndex;
}
// Function to set the lastIndex property of the regex object
function setLastIndex() {
    regex.lastIndex = 10;
    output.textContent = "LastIndex property after setting: " +
regex.lastIndex;
}
```

This program uses the following properties and methods of the RegExp object:

- test(): Tests the regex pattern against a string.

- exec(): Executes the regex pattern against a string and returns an array containing the matches.

- source: Returns the source string of the regex pattern.

- flags: Returns the flags string of the regex pattern.

- lastIndex: Returns the index at which to start the next match.

7 g) write a program using date object properties and methods

HTML File (7g.html)

```
<!DOCTYPE html>
<html>
<head>
    <title>Date Object Properties and Methods</title>
</head>
<body>
    <h1 id="heading">Date Object Properties and Methods</h1>
    <button onclick="getDate()">Get Date</button>
    <button onclick="getDay()">Get Day</button>
    <button onclick="getFullYear()">Get Full Year</button>
    <button onclick="getHours()">Get Hours</button>
    <button onclick="getMilliseconds()">Get Milliseconds</button>
    <button onclick="getMinutes()">Get Minutes</button>
    <button onclick="getMonth()">Get Month</button>
    <button onclick="getSeconds()">Get Seconds</button>
    <button onclick="getTime()">Get Time</button>
    <button onclick="getTimezoneOffset()">Get Timezone
Offset</button>
    <button onclick="setDate()">Set Date</button>
    <button onclick="setFullYear()">Set Full Year</button>
    <button onclick="setHours()">Set Hours</button>
    <button onclick="setMilliseconds()">Set Milliseconds</button>
    <button onclick="setMinutes()">Set Minutes</button>
    <button onclick="setMonth()">Set Month</button>
    <button onclick="setSeconds()">Set Seconds</button>
    <button onclick="setTime()">Set Time</button>
    <p id="output"></p>
    <script src="7g.js"></script>
</body>  </html>
```

JavaScript File (7g.js)

```
let output = document.getElementById("output");
let date = new Date();
// Function to get the date
function getDate() {
	output.textContent = "Date: " + date.getDate();
}
// Function to get the day
function getDay() {
	output.textContent = "Day: " + date.getDay();
}
// Function to get the full year
function getFullYear() {
	output.textContent = "Full Year: " + date.getFullYear();
}
// Function to get the hours
function getHours() {
	output.textContent = "Hours: " + date.getHours();
}
// Function to get the milliseconds
function getMilliseconds() {
	output.textContent = "Milliseconds: " + date.getMilliseconds();
}
// Function to get the minutes
function getMinutes() {
	output.textContent = "Minutes: " + date.getMinutes();
}
// Function to get the month
function getMonth() {
```

```
        output.textContent = "Month: " + date.getMonth();
}
// Function to get the seconds
function getSeconds() {
        output.textContent = "Seconds: " + date.getSeconds();
}
// Function to get the time
function getTime() {
        output.textContent = "Time: " + date.getTime();
}
// Function to get the timezone offset
function getTimezoneOffset() {
        output.textContent = "Timezone Offset: " +
date.getTimezoneOffset();
}
// Function to set the date
function setDate() {
        date.setDate(15);
        output.textContent = "Date set to: " + date;
}
// Function to set the full year
function setFullYear() {
        date.setFullYear(2025);
        output.textContent = "Full Year set to: " + date;
}
// Function to set the hours
function setHours() {
        date.setHours(10);
        output.textContent = "Hours set to: " + date;
}
```

```
// Function to set the milliseconds
function setMilliseconds() {
    date.setMilliseconds(500);
    output.textContent = "Milliseconds set to: " + date;
}
// Function to set the minutes
function setMinutes() {
    date.setMinutes(30);
    output.textContent = "Minutes set to: " + date;
}
// Function to set the month
function setMonth() {
    date.setMonth(5);
    output.textContent = "Month set to: " + date;
}
// Function to set the seconds
function setSeconds() {
    date.setSeconds(45);
    output.textContent = "Seconds set to: " + date;
}
// Function to set the time
function setTime() {
    date.setTime(1643723400000);
    output.textContent = "Time set to: " + date;
}
```

This program uses the following properties and methods of the Date object:

- getDate(): Returns the day of the month.

- getDay(): Returns the day of the week.

- getFullYear(): Returns the year.

- getHours(): Returns the hour.

- getMilliseconds(): Returns the milliseconds.

- getMinutes(): Returns the minutes.

- getMonth(): Returns the month.

- getSeconds(): Returns the seconds.

- getTime(): Returns the number of milliseconds since the Unix Epoch.

- getTimezoneOffset(): Returns the time difference between UTC and local time.

- setDate(): Sets the day of the month.

- setFullYear(): Sets the year.

- setHours(): Sets the hour.

- setMilliseconds(): Sets the milliseconds.

- setMinutes(): Sets the minutes.

7 h) write a program to explain user defined object by using properties, methods, accessors, constructors and display

HTML File (7h.html)

```
<!DOCTYPE html>
<html>
<head>
    <title>User Defined Object</title>
</head>
<body>
    <h1 id="heading">User Defined Object</h1>
    <button onclick="createObject()">Create Object</button>
    <button onclick="displayObject()">Display Object</button>
    <button onclick="accessProperties()">Access Properties</button>
    <button onclick="callMethod()">Call Method</button>
    <button onclick="useAccessor()">Use Accessor</button>
    <p id="output"></p>
    <script src="7h.js"></script>
</body>
</html>
```

JavaScript File (7h.js)

```
let output = document.getElementById("output");
// Constructor function
function Person(name, age) {
    this.name = name;
    this.age = age;
}
// Method
Person.prototype.sayHello = function() {
    return "Hello, my name is " + this.name + " and I am " + this.age + " years old.";
}
// Accessor (getter)
Object.defineProperty(Person.prototype, "fullName", {
    get: function() {
        return this.name + " " + this.age;
    }
});
// Create an object
let person;
function createObject() {
    person = new Person("Madhu Kumar", 35);
    output.textContent = "Object created successfully!";
}
// Display object
function displayObject() {
    if (person) {
        output.textContent = "Name: " + person.name + ", Age: " + person.age;
    } else {
```

```
        output.textContent = "Object not created yet!";
    }
}
// Access properties
function accessProperties() {
    if (person) {
        output.textContent = "Name: " + person.name + ", Age: " +
person.age;
    } else {
        output.textContent = "Object not created yet!";
    }
}
// Call method
function callMethod() {
    if (person) {
        output.textContent = person.sayHello();
    } else {
        output.textContent = "Object not created yet!";
    }
}
// Use accessor
function useAccessor() {
    if (person) {
        output.textContent = "Full Name: " + person.fullName;
    } else {
        output.textContent = "Object not created yet!";
    }
}
```

This program explains the following concepts:

- Constructor: A special function used to initialize objects when they are created. In this example, the Person constructor function is used to create a new Person object.

- Properties: Data members of an object. In this example, the name and age properties are defined in the Person constructor function.

- Methods: Functions that belong to an object. In this example, the sayHello method is defined in the Person prototype.

- Accessors: Special functions that allow you to access and modify properties of an object. In this example, the fullName accessor is defined using the Object.defineProperty method.

- Display: The program displays the object's properties and the result of calling the sayHello method using the output paragraph element.

8. Java Script Conditional Statements and Loops

8 a. Write a program which asks the user to enter three integers, obtains the numbers from the user and outputs HTML text that displays the larger number followed by the words "LARGER NUMBER" in an information message dialog. If the numbers ale equal, output HTML text as "EQUAL NUMBERS.

HTML File (8a.html)

```
<!DOCTYPE html>
<html>
<head>
    <title>Larger Number</title>
</head>
<body>
    <h1 id="heading">Larger Number</h1>
    <input id="num1" type="number" placeholder="Enter first number">
    <input id="num2" type="number" placeholder="Enter second number">
    <input id="num3" type="number" placeholder="Enter third number">
    <button onclick="findLargerNumber()">Find Larger Number</button>
    <p id="output"></p>
    <script src="8a.js"></script>
</body>
</html>
```

JavaScript File (8a.js)

```
let output = document.getElementById("output");
function findLargerNumber() {
    let num1 = parseInt(document.getElementById("num1").value);
    let num2 = parseInt(document.getElementById("num2").value);
    let num3 = parseInt(document.getElementById("num3").value);
```

```
    if (num1 > num2 && num1 > num3) {
        output.textContent = num1 + " LARGER NUMBER";
    } else if (num2 > num1 && num2 > num3) {
        output.textContent = num2 + " LARGER NUMBER";
    } else if (num3 > num1 && num3 > num2) {
        output.textContent = num3 + " LARGER NUMBER";
    } else {
        output.textContent = "EQUAL NUMBERS";
    }
}
```

This program uses the following concepts:

- Input elements: The program uses three input elements to obtain three numbers from the user.

- Button element: The program uses a button element to trigger the findLargerNumber function when clicked.

- parseInt function: The program uses the parseInt function to convert the input values from strings to integers.

- Conditional statements: The program uses conditional statements (if and else if) to compare the numbers and determine the larger number.

- Output element: The program uses a p element to display the output message.

8 b. Write a program to display week days using switch case.

HTML File (8b.html)

```
<!DOCTYPE html>
<html>
<head>
    <title>Weekdays using Switch Case</title>
</head>
<body>
    <h1 id="heading">Weekdays using Switch Case</h1>
    <input id="day" type="number" placeholder="Enter day number (1-7)">
    <button onclick="displayDay()">Display Day</button>
    <p id="output"></p>
    <script src="8b.js"></script>
</body>
</html>
```

JavaScript File (8b.js)

```
let output = document.getElementById("output");
function displayDay() {
    let dayNumber = parseInt(document.getElementById("day").value);
    switch (dayNumber) {
        case 1:
            output.textContent = "Monday";
            break;
        case 2:
            output.textContent = "Tuesday";
            break;
        case 3:
            output.textContent = "Wednesday";
            break;
        case 4:
```

```
                output.textContent = "Thursday";
                break;
            case 5:
                output.textContent = "Friday";
                break;
            case 6:
                output.textContent = "Saturday";
                break;
            case 7:
                output.textContent = "Sunday";
                break;
            default:
                output.textContent = "Invalid day number";
        }
}
```

This program uses the following concepts:

- Input element: The program uses an input element to obtain the day number from the user.

- Button element: The program uses a button element to trigger the displayDay function when clicked.

- Switch case statement: The program uses a switch case statement to determine the weekday based on the day number entered by the user.

- Output element: The program uses a p element to display the weekday.

8 c. write a program to print 1 to 10 numbers using for, while and do-while loops.

HTML File (8c.html)

```
<!DOCTYPE html>
<html>
<head>
    <title>Loops Example</title>
</head>
<body>
    <h1 id="heading">Loops Example</h1>
    <button onclick="printUsingFor()">Print using For Loop</button>
    <button onclick="printUsingWhile()">Print using While Loop</button>
    <button onclick="printUsingDoWhile()">Print using Do-While Loop</button>
    <p id="output"></p>
    <script src="8c.js"></script>
</body>
</html>
```

JavaScript File (8c.js)

```
let output = document.getElementById("output");
function printUsingFor() {
    output.textContent = "Using For Loop: ";
    for (let i = 1; i <= 10; i++) {
        output.textContent += i + " ";
    }
}
function printUsingWhile() {
    output.textContent = "Using While Loop: ";
    let i = 1;
    while (i <= 10) {
        output.textContent += i + " ";
        i++;
    }
}
function printUsingDoWhile() {
    output.textContent = "Using Do-While Loop: ";
    let i = 1;
    do {
        output.textContent += i + " ";
        i++;
    } while (i <= 10);
}
```

This program uses the following concepts:

- For loop: The for loop is used to iterate over a block of code for a specified number of times.

- While loop: The while loop is used to execute a block of code as long as a specified condition is true.

- Do-while loop: The do-while loop is used to execute a block of code once, and then repeatedly as long as a specified condition is true.

- Output element: The program uses a p element to display the output.

8 d. write a program to print data in object using for-in, for-each and for-of loops.

HTML File (8d.html)

```
<!DOCTYPE html>
<html>
<head>
    <title>Loops Example</title>
</head>
<body>
    <h1 id="heading">Loops Example</h1>
    <button onclick="printUsingForIn()">Print using For-In Loop</button>
    <button onclick="printUsingForEach()">Print using For-Each Loop</button>
    <button onclick="printUsingForOf()">Print using For-Of Loop</button>
    <p id="output"></p>
    <script src="8d.js"></script>
</body>
</html>
```

JavaScript File (8d.js)

```
let output = document.getElementById("output");
let person = {
    name: "Ramesh Kumar",
    age: 30,
    occupation: "Software Developer"
};
function printUsingForIn() {
    output.textContent = "Using For-In Loop: ";
    for (let key in person) {
        output.textContent += `${key}: ${person[key]}, `;
    }
}
function printUsingForEach() {
    output.textContent = "Using For-Each Loop: ";
    Object.keys(person).forEach(key => {
        output.textContent += `${key}: ${person[key]}, `;
    });
}
function printUsingForOf() {
    output.textContent = "Using For-Of Loop: ";
    for (let value of Object.values(person)) {
        output.textContent += `${value}, `;
    }
}
```

This program uses the following concepts:

- For-in loop: The for-in loop is used to iterate over the properties of an object.

- For-each loop: The forEach() method is used to execute a function for each element in an array. In this case, we use Object.keys() to get an array of the object's keys and then use forEach() to iterate over the keys.

- For-of loop: The for-of loop is used to iterate over the values of an iterable object. In this case, we use Object.values() to get an array of the object's values and then use for-of to iterate over the values.

- Output element: The program uses a p element to display the output.

8 e. Develop a program to determine whether a given number is an 'ARMSTRONG NUMBER, or not. [Eg: 153 is an Armstrong number, since sum of the cube of the digits is equal to the number i.e.,13 + 53+ 33 : 153.]

HTML File (8e.html)

```
<!DOCTYPE html>
<html>
<head>
    <title>Armstrong Number</title>
</head>
<body>
    <h1 id="heading">Armstrong Number</h1>
    <input id="num" type="number" placeholder="Enter a number">
    <button onclick="checkArmstrong()">Check</button>
    <p id="output"></p>
    <script src="8e.js"></script>
</body>
</html>
```

JavaScript File (8e.js)

```
let output = document.getElementById("output");
function checkArmstrong() {
    let num = parseInt(document.getElementById("num").value);
    let originalNum = num;
    let sum = 0;
    let digitCount = num.toString().length;
    while (num != 0) {
        let digit = num % 10;
        sum += Math.pow(digit, digitCount);
        num = Math.floor(num / 10);
```

```
    }
    if (sum == originalNum) {
        output.textContent = `${originalNum} is an Armstrong number`;
    } else {
        output.textContent = `${originalNum} is not an Armstrong number`;
    }
}
```

This program uses the following concepts:

- Input element: The program uses an input element to obtain the number from the user.

- Button element: The program uses a button element to trigger the checkArmstrong function when clicked.

- While loop: The program uses a while loop to extract each digit from the number and calculate the sum of the cubes of the digits.

- Math.pow function: The program uses the Math.pow function to calculate the cube of each digit.

- Output element: The program uses a p element to display the output.

8 f. Write a program to display the denomination of the amount deposited in the bank in terms of 100s, 50s, 20s, 10s, 5s, 2s & 1s. (Eg: If amount deposited is Rs.163, the output should be l-100s, l-50s, 1- 10s, 1-2s & 1-1s)

HTML File (8f.html)

```
<!DOCTYPE html>
<html>
<head>
    <title>Denomination Calculator</title>
</head>
<body>
    <h1 id="heading">Denomination Calculator</h1>
    <input id="amount" type="number" placeholder="Enter amount deposited">
    <button onclick="calculateDenomination()">Calculate</button>
    <p id="output"></p>
    <script src="8f.js"></script>
</body>
</html>
```

JavaScript File (8f.js)

```
let output = document.getElementById("output");
function calculateDenomination() {
	let amount = parseInt(document.getElementById("amount").value);
	let denominations = [100, 50, 20, 10, 5, 2, 1];
	let result = "";

	for (let i = 0; i < denominations.length; i++) {
		let count = Math.floor(amount / denominations[i]);
		amount %= denominations[i];

		if (count > 0) {
			result += `${count} - ${denominations[i]}s, `;
		}
	}
	output.textContent = "Denomination: " + result.slice(0, -2);
}
```

This program uses the following concepts:

- Input element: The program uses an input element to obtain the amount deposited from the user.
- Button element: The program uses a button element to trigger the calculateDenomination function when clicked.
- For loop: The program uses a for loop to iterate over the denominations array and calculate the count of each denomination.
- Math.floor function: The program uses the Math.floor function to calculate the count of each denomination.
- Modulus operator: The program uses the modulus operator (%) to calculate the remaining amount after subtracting the denomination.
- Output element: The program uses a p element to display the output.