BCSL504- WEB TECHNOLOGY LAB MANUAL

Laboratory Components

1. HTML Basics "Myfirstwebpage.html" - Fundamental HTML tags and structures
2. HTML Tables "Table.html" - Create a class timetable with advanced table features
3. External CSS Styling "style.css" - Demonstrate various CSS selectors and styles
4. HTML Forms with CSS "registration.html" - Input elements styled with CSS
5. Semantic HTML and CSS "newspaper.html" - Use of semantic elements with styling
6. JavaScript Calculator Simple calculator with multiple operations
7. JavaScript and JSON JSON manipulation, conversion, and hashing
8. PHP and Databases Visitor counter and database sorting with PHP
9. jQuery DOM Manipulation Content appending, animation, and color changing
10. Ajax Operations Ajax requests with and without jQuery, JSON handling

Experiment-01

Develop the HTML page named as “Myfirstwebpage.html”. Add the following tags with relevant content.

1. Set the title of the page as “My First Web Page”

2. Within the body use the following tags:

a) Moving text = “Basic HTML Tags”

b) Different heading tags (h1 to h6)

c) Paragraph

d) Horizontal line

e) Line Break

f) Block Quote

g) Pre tag

h) Different Logical Style (<b>, <u>, <sub>, <sup> etc.)

Program:

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

```
    <title>My First Web Page</title>
    <!-- <style>
        body {
            font-family: Arial, sans-serif;
            line-height: 1.6;
            margin: 0;
            padding: 20px;
        }
    </style> -->
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <marquee>Basic HTML Tags</marquee>
    <h1>This is Heading 1</h1>
    <h2>This is Heading 2</h2>
    <h3>This is Heading 3</h3>
    <h4>This is Heading 4</h4>
    <h5>This is Heading 5</h5>
    <h6>This is Heading 6</h6>
```

** This is a paragraph. It demonstrates the use of the paragraph tag in HTML. Paragraphs are used to group related content together. **

<hr>

** This is another paragraph.
This text appears on a new line due to the line break tag. **

<blockquote>

This is a block quote. It's often used to highlight quoted text from another source.

</blockquote>

<pre>

This is preformatted text.

It preserves both spaces and

line breaks, making it useful

for displaying code or ASCII art.

</pre>

** **

**Here are examples of logical styles:
**

**Bold text
**

**Italic text
**

**<u>Underlined text</u>
**

```
    <strong>Strong text</strong><br>

    <em>Emphasized text</em><br>

    Text with <sub>subscript</sub> and <sup>superscript</sup>

  </p>

</body>

</html>
```

Explanation

Step 1: Document Structure

<!DOCTYPE html>: This declaration tells the browser that this is an HTML5 document.

<html lang="en">: The root element of the HTML page. The "lang" attribute specifies that the language is English.

<head>: This section contains metadata about the document.

<body>: This section contains the visible content of the webpage.

Step 2: The Head Section

<meta charset="UTF-8">: Specifies the character encoding for the document (UTF-8 supports many languages).

<meta name="viewport" ...>: This tag helps with responsive design, making the page display properly on different devices.

<title>: Sets the title of the webpage, which appears in the browser tab.

<link rel="stylesheet" href="style.css">: Links to an external CSS file for styling the page.

Step 3: The Body Content

<marquee>: Creates scrolling text (Note: This tag is outdated and not recommended for modern websites).

Headings: <h1> to <h6> tags represent six levels of section headings, with <h1> being the highest (most important) and <h6> the lowest.

<p>: Defines a paragraph of text.

<hr>: Creates a horizontal line, often used to separate content.

: Inserts a single line break.

<blockquote>: Used for quoting content from another source.

<pre>: Defines preformatted text, which preserves both spaces and line breaks.

Step 4: Text Formatting

<b> and : Both typically display text in bold. is preferred as it adds semantic meaning of strong importance.

<i> and : Both typically display text in italics. is preferred as it adds semantic meaning of emphasis.

<u>: Underlines text (use with caution as it can be confused with hyperlinks).

<sub>: Defines subscript text.

<sup>: Defines superscript text.

Experiment-02

Develop the HTML page named as “Table.html” to display your class time table.

a) Provide the title as Time Table with table header and table footer, row-span and col-span etc.

b) Provide various colour options to the cells (Highlight the lab hours and elective hours with different

colours.)

c) Provide colour options for rows.

Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Time Table</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         line-height: 1.6;
         margin: 0;
```

```
    padding: 20px;
}
table {
    width: 100%;
    border-collapse: collapse;
}
th, td {
    border: 1px solid #ddd;
    padding: 8px;
    text-align: center;
}
th {
    background-color: #f2f2f2;
}
.lab-hours {
    background-color: #ffcccb;
}
.elective-hours {
    background-color: #90ee90;
}
```

```
        .lunch {
            background-color: #ffd700;
        }
        .odd-row {
            background-color: #f8f8f8;
        }
        tfoot {
            background-color: #e6e6e6;
            font-weight: bold;
        }
    </style>
</head>
<body>
    <table>
        <thead>
            <tr>
                <th colspan="7">Class Time Table</th>
            </tr>
            <tr>
                <th>Time</th>
```

```
        <th>Monday</th>
        <th>Tuesday</th>
        <th>Wednesday</th>
        <th>Thursday</th>
        <th>Friday</th>
        <th>Saturday</th>
    </tr>
</thead>
<tbody>
    <tr>
        <td>9:00 - 10:00</td>
        <td>Math</td>
        <td>English</td>
        <td>Science</td>
        <td>History</td>
        <td>Geography</td>
        <td rowspan="6">No Classes</td>
    </tr>
    <tr class="odd-row">
        <td>10:00 - 11:00</td>
```

```
    <td>Physics</td>
    <td>Chemistry</td>
    <td>Biology</td>
    <td class="elective-hours">Computer Science</td>
    <td class="elective-hours">Art</td>
</tr>
<tr>
    <td>11:00 - 12:00</td>
    <td class="lab-hours" colspan="2">Physics Lab</td>
    <td class="lab-hours" colspan="2">Chemistry Lab</td>
    <td class="lab-hours">Biology Lab</td>
</tr>
<tr class="odd-row">
    <td>12:00 - 13:00</td>
    <td class="lunch" colspan="5">Lunch Break</td>
</tr>
<tr>
    <td>13:00 - 14:00</td>
    <td>Literature</td>
    <td>Math</td>
```

```
            <td>English</td>
            <td>Physics</td>
            <td>Chemistry</td>
        </tr>
        <tr class="odd-row">
            <td>14:00 - 15:00</td>
            <td class="elective-hours">Music</td>
            <td class="elective-hours">Drama</td>
            <td class="lab-hours" colspan="3">Computer Lab</td>
        </tr>
    </tbody>
    <tfoot>
        <tr>
            <td colspan="7">* Lab hours are highlighted in pink, elective hours in light green</td>
        </tr>
    </tfoot>
  </table>
</body>
</html>
```

Explanation

Step 1: Document Structure

The document starts with the standard HTML5 declaration and structure.

The <head> section contains metadata and styling information.

The <body> section contains the visible content, which in this case is a table.

Step 2: Styling (CSS)

The <style> tag in the <head> section defines the appearance of the table and its elements.

It sets the font, adjusts spacing, and defines colors for different types of classes.

Specific classes are created for lab hours, elective hours, lunch, and alternating row colors.

Step 3: Table Structure

The table is created using the <table> tag.

It has three main parts: <thead> (table header), <tbody> (table body), and <tfoot> (table footer).

Step 4: Table Header (<thead>)

The first row spans all columns and displays "Class Time Table".

The second row shows the days of the week and a "Time" column.

Step 5: Table Body (<tbody>)

Each <tr> represents a row in the timetable.

The first column in each row shows the time slot.

Subsequent columns represent different subjects for each day.

Special formatting is applied to certain cells:

Lab hours are highlighted in pink.

Elective hours are highlighted in light green.

The lunch break spans across multiple columns.

The "Saturday" column spans multiple rows to show "No Classes".

Alternate rows have a slightly different background color for better readability.

Step 6: Special Cell Attributes

colspan is used to make cells span multiple columns (e.g., for labs and lunch).

rowspan is used to make the "Saturday" cell span multiple rows.

Classes are applied to cells for specific styling (e.g., "lab-hours", "elective-hours").

Step 7: Table Footer (<tfoot>)

The footer provides a legend explaining the color coding used in the table.

This HTML document demonstrates several important concepts:

Table structure and formatting

Use of CSS for styling

Cell spanning (both rows and columns)

Color coding for different types of classes

Responsive design considerations (though the table itself isn't responsive)

Experiment-03

Develop an external style sheet named as “style.css” and provide different styles for h2, h3, hr, p, div, span,

time, img & a tags. Apply different CSS selectors for tags and demonstrate the significance of each.

Program:

Style.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Sample Styled Page (No Div)</title>
   <link rel="stylesheet" href="style.css">
</head>
<body>
   <main id="main-content">
      <h2>Welcome to Our Styled Page</h2>
```

```
<p>This is a paragraph right after an h2. It demonstrates the adjacent sibling selector.</p>

<h3>Hover over me!</h3>

<hr>

<p lang="en">This paragraph has a lang attribute, demonstrating the attribute selector.</p>

<p>Here's a <span class="highlight">highlighted</span> word using the class selector.</p>

<section>
  <p>This paragraph is inside a section, showing the descendant selector.</p>
  <span>This span is a direct child of the section.</span>
</section>
```

```
<p>The current date and time: <time datetime="2023-08-31">August 31, 2023</time></p>

<p>Notice how the first letter of each paragraph is styled differently.</p>

<article class="special">
  <p>This paragraph is inside an article with class="special".</p>
</article>

<img src="https://d3njjcbhbojbot.cloudfront.net/api/utilities/v1/imageproxy/https://images.ctfassets.net/wp1lcwdav1p1/6z473u5f7WaFUnr9GxDEk2/085274c4a841bd2dc900ebca36c43c9c/GettyImages-1255905237.jpg?w=1500&h=680&q=60&fit=fill&f=faces&fm=jpg&fl=progressive&auto=format%2Ccompress&dpr=1&w=1000" alt="A placeholder image">

<p>Check out this <a href="https://searchcreators.org/">link</a> to see different link states.</p>
</main>
</body>
```

```
</html>
```

Style.css

```
/* Element Selector */
h2 {
    color: #2c3e50;
    font-family: 'Arial', sans-serif;
    border-bottom: 2px solid #3498db;
    padding-bottom: 10px;
}

/* Element Selector with Pseudo-class */
h3:hover {
    color: #e74c3c;
    cursor: pointer;
    transition: color 0.3s ease;
}
```

```
/* Element Selector */

hr {

   border: 0;

   height: 1px;

   background-image: linear-gradient(to right, rgba(0, 0, 0, 0), rgba(0, 0, 0, 0.75), rgba(0, 0, 0, 0));

}


/* Element Selector with Attribute */

p[lang] {

   font-style: italic;

}


/* Class Selector */

.highlight {

   background-color: #f1c40f;

   padding: 5px;

}
```

```
/* ID Selector */
#main-content {
    max-width: 800px;
    margin: 0 auto;
    padding: 20px;
    background-color: #ecf0f1;
}

/* Descendant Selector */
div p {
    line-height: 1.6;
    margin-bottom: 15px;
}

/* Child Selector */
div > span {
    font-weight: bold;
    color: #16a085;
}
```

```
/* Adjacent Sibling Selector */
h2 + p {
    font-size: 1.1em;
    color: #7f8c8d;
}

/* Attribute Selector */
time[datetime] {
    color: #8e44ad;
    font-weight: bold;
}

/* Pseudo-element Selector */
p::first-letter {
    font-size: 1.5em;
    font-weight: bold;
    color: #c0392b;
```

```
}



/* Multiple Selectors */
img, a {
    border: 1px solid #bdc3c7;
    padding: 5px;
}
/* Pseudo-class Selector for Links */
a:link, a:visited {
    color: #3498db;
    text-decoration: none;
}
a:hover, a:active {
    color: #e74c3c;
    text-decoration: underline;
}
/* Attribute Selector for Images */
img[alt] {
```

```
    max-width: 100%;
    height: auto;
}
/* Combining Selectors */
div.special p {
    text-indent: 20px;
    color: #27ae60;
}
```

Explanation

Step 1: HTML Structure

The document starts with the standard HTML5 declaration and basic structure.

The <head> section includes metadata, title, and a link to an external CSS file (style.css).

The <body> contains a <main> element with the id "main-content", which holds all the visible content.

Step 2: Main Content Structure

The main content includes various HTML elements like h2, h3, p, hr, section, article, img, and a.

These elements demonstrate different CSS selectors and properties.

Step 3: CSS Styling (from style.css)

Element Selectors:

h2: Styled with color, font-family, border-bottom, and padding.

h3: Has a hover effect changing color with a transition.

hr: Styled as a gradient line.

Attribute Selectors:

p[lang]: Applies italic style to paragraphs with a lang attribute.

time[datetime]: Styles time elements with a datetime attribute.

img[alt]: Ensures responsive sizing for images with alt text.

Class Selector:

.highlight: Applies background color and padding.

ID Selector:

#main-content: Sets max-width, margin, padding, and background color.

Descendant Selector:

div p: Styles paragraphs inside divs (note: no divs in this HTML, so this won't apply).

Child Selector:

div > span: Styles spans that are direct children of divs (also won't apply here).

Adjacent Sibling Selector:

h2 + p: Styles paragraphs immediately following h2 elements.

Pseudo-element Selector:

p::first-letter: Styles the first letter of each paragraph.

Multiple Selectors:

img, a: Applies border and padding to both images and links.

Pseudo-class Selectors for Links:

a:link, a:visited, a:hover, a:active: Different styles for various link states.

Combining Selectors:

div.special p: Would style paragraphs in divs with class "special" (not applicable here).

Step 4: Specific Content and Styling Examples

The h2 and following p demonstrate the adjacent sibling selector.

The h3 has a hover effect.

A paragraph with lang="en" shows the attribute selector in action.

The "highlight" class is applied to a span within a paragraph.

A section with a paragraph and span demonstrates descendant and child selectors (though the CSS for div won't apply).

The <time> element shows the datetime attribute selector.

An article with class="special" is included, but the CSS for div.special won't apply.

An image and a link are included to demonstrate various selectors and responsive design.

Experiment-04

Develop HTML page named as “registration.html” having variety of HTML input elements with background colors, table for alignment & provide font colors & size using CSS styles.

Program:

```
<!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: #f0f0f0;
      margin: 0;
      padding: 20px;
    }
    h1 {
```

```
    color: #333;
    text-align: center;
}
table {
    width: 100%;
    max-width: 600px;
    margin: 0 auto;
    background-color: #fff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
}
td {
    padding: 10px;
}
label {
    color: #555;
    font-weight: bold;
}
```

```
    input[type="text"], input[type="email"], input[type="password"],
select, textarea {
        width: 100%;
        padding: 8px;
        border: 1px solid #ddd;
        border-radius: 4px;
        box-sizing: border-box;
        font-size: 16px;
    }
    input[type="radio"], input[type="checkbox"] {
        margin-right: 5px;
    }
    input[type="submit"] {
        background-color: #4CAF50;
        color: white;
        padding: 10px 20px;
        border: none;
        border-radius: 4px;
        cursor: pointer;
```

```
            font-size: 18px;
        }
        input[type="submit"]:hover {
            background-color: #45a049;
        }
        .error {
            color: #ff0000;
            font-size: 14px;
        }
    </style>
</head>
<body>
    <h1>Registration Form</h1>
    <form action="#" method="post">
        <table>
            <tr>
                <td><label for="fullname">Full Name:</label></td>
                <td><input type="text" id="fullname" name="fullname"
required></td>
```

```
        </tr>
        <tr>
            <td><label for="email">Email:</label></td>
            <td><input type="email" id="email" name="email"
required></td>
        </tr>
        <tr>
            <td><label for="password">Password:</label></td>
            <td><input type="password" id="password" name="password"
required></td>
        </tr>
        <tr>
            <td><label for="confirm_password">Confirm
Password:</label></td>
            <td><input type="password" id="confirm_password"
name="confirm_password" required></td>
        </tr>
        <tr>
            <td><label>Gender:</label></td>
            <td>
```

```
            <input type="radio" id="male" name="gender" value="male"
required>
            <label for="male">Male</label>
            <input type="radio" id="female" name="gender"
value="female" required>
            <label for="female">Female</label>
            <input type="radio" id="other" name="gender" value="other"
required>
            <label for="other">Other</label>
          </td>
        </tr>
        <tr>
          <td><label for="birthdate">Date of Birth:</label></td>
          <td><input type="date" id="birthdate" name="birthdate"
required></td>
        </tr>
        <tr>
          <td><label for="country">Country:</label></td>
          <td>
            <select id="country" name="country" required>
```

```
                <option value="">Select a country</option>
                <option value="usa">India</option>
                <option value="uk">United Kingdom</option>
                <option value="canada">Canada</option>
                <option value="australia">Australia</option>
                <option value="other">Other</option>
            </select>
        </td>
    </tr>
    <tr>
        <td><label for="interests">Interests:</label></td>
        <td>
            <input type="checkbox" id="sports" name="interests[]"
value="sports">
            <label for="sports">Sports</label>
            <input type="checkbox" id="music" name="interests[]"
value="music">
            <label for="music">Music</label>
            <input type="checkbox" id="reading" name="interests[]"
value="reading">
```

```
            <label for="reading">Reading</label>
            <input type="checkbox" id="travel" name="interests[]"
value="travel">
            <label for="travel">Travel</label>
          </td>
        </tr>
        <tr>
          <td><label for="bio">Bio:</label></td>
          <td><textarea id="bio" name="bio" rows="4"></textarea></td>
        </tr>
        <tr>
          <td colspan="2" style="text-align: center;">
            <input type="submit" value="Register">
          </td>
        </tr>
      </table>
    </form>
</body>
</html>
```

Explanation

Step 1: Document Structure

The document starts with the standard HTML5 declaration and structure.

The <head> section contains metadata, title, and embedded CSS styles.

The <body> section contains the form.

Step 2: Styling (CSS)

The style section defines the appearance of the form and its elements.

It sets the font, colors, layout, and responsive design for the form.

Specific styles are applied to different input types, labels, and the submit button.

A class for error messages is defined but not used in the HTML.

Step 3: Form Structure

The form is created using the <form> tag with a placeholder action "#".

A table is used to structure the form layout.

Step 4: Form Fields

The form includes the following fields:

Full Name (text input)

Email (email input)

Password (password input)

Confirm Password (password input)

Gender (radio buttons)

Date of Birth (date input)

Country (select dropdown)

Interests (checkboxes)

Bio (textarea)

Submit button

Step 5: Input Attributes and Validation

Most fields are marked as required.

Appropriate input types are used (e.g., email for email address, date for birthdate).

The password field has a confirmation field.

The country dropdown includes a default "Select a country" option.

Step 6: Styling Details

The form has a white background with rounded corners and a shadow.

Input fields have consistent styling with padding and border-radius.

The submit button has a green color with a hover effect.

The layout is responsive, with a maximum width set for larger screens.

This registration form demonstrates several important concepts:

HTML form structure and various input types

Basic client-side form validation using HTML5 attributes

Responsive design using CSS

Styling form elements for a consistent and appealing look

Use of tables for form layout (though modern practices often prefer CSS grid or flexbox)

Experiment-05

Develop HTML page named as “newpaper.html” having variety of HTML semantic elements with background colors, text-colors & size for figure, table, aside, section, article, header, footer… etc

Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>The Daily Chronicle</title>
   <style>
      body {
         font-family: 'Georgia', serif;
         line-height: 1.6;
         color: #333;
         max-width: 1200px;
         margin: 0 auto;
         padding: 20px;
```

```
    background-color: #f4f4f4;
}
header {
    background-color: #1a1a1a;
    color: #fff;
    padding: 20px;
    text-align: center;
}
header h1 {
    margin: 0;
    font-size: 2.5em;
}
nav {
    background-color: #333;
    color: #fff;
    padding: 10px;
}
nav ul {
    list-style-type: none;
```

```
    padding: 0;
    margin: 0;
    display: flex;
    justify-content: center;
}
nav ul li {
    margin: 0 10px;
}
nav ul li a {
    color: #fff;
    text-decoration: none;
}
main {
    display: flex;
    margin-top: 20px;
}
section {
    flex: 2;
    margin-right: 20px;
```

```
}
article {
    background-color: #fff;
    padding: 20px;
    margin-bottom: 20px;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
}
article h2 {
    color: #1a1a1a;
    font-size: 1.8em;
}
aside {
    flex: 1;
    background-color: #e6e6e6;
    padding: 20px;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
}
figure {
    margin: 0;
```

```
    text-align: center;
}
figure img {
    max-width: 100%;
    height: auto;
}
figcaption {
    font-style: italic;
    color: #666;
    font-size: 0.9em;
}
table {
    width: 100%;
    border-collapse: collapse;
    margin-bottom: 20px;
}
th, td {
    border: 1px solid #ddd;
    padding: 10px;
```

```
        text-align: left;
    }
    th {
        background-color: #f2f2f2;
    }
    footer {
        background-color: #1a1a1a;
        color: #fff;
        text-align: center;
        padding: 10px;
        margin-top: 20px;
    }
  </style>
</head>
<body>
  <header>
    <h1>The Daily Chronicle</h1>
  </header>
```

```
<nav>
  <ul>
    <li><a href="#">Home</a></li>
    <li><a href="#">Politics</a></li>
    <li><a href="#">Technology</a></li>
    <li><a href="#">Sports</a></li>
    <li><a href="#">Entertainment</a></li>
  </ul>
</nav>

<main>
  <section>
    <article>
      <h2>Breaking News: Major Technological Breakthrough</h2>
      <p>Scientists have announced a groundbreaking discovery in the field of quantum computing, potentially revolutionizing the tech industry.</p>
      <figure>
        <img src="https://www.cnet.com/a/img/resize/c7cb26e927bebaa784fb55a01e71d7fecb15d2e3/hub/2019/06/26/3f76e99d-8055-46f3-8f27-558ee276b665/20180405-
```

```
ibm-q-quantum-computer-
02.jpg?auto=webp&fit=crop&height=675&width=1200" alt="Quantum
Computer">
            <figcaption>A state-of-the-art quantum computer at the research
facility</figcaption>
        </figure>
    </article>

    <article>
        <h2>Local Sports Team Wins Championship</h2>
        <p>In a thrilling match, our local team secured victory in the
national championship, bringing pride to our city.</p>
        <table>
            <tr>
                <th>Team</th>
                <th>Score</th>
            </tr>
            <tr>
                <td>Local Heroes</td>
                <td>3</td>
```

```
            </tr>
            <tr>
                <td>Visiting Challengers</td>
                <td>2</td>
            </tr>
        </table>
    </article>
</section>

<aside>
    <h3>Weather Update</h3>
    <p>Expect sunny skies with a high of 75°F (24°C) today.</p>

    <h3>Upcoming Events</h3>
    <ul>
        <li>City Festival - This Weekend</li>
        <li>Tech Conference - Next Month</li>
        <li>Charity Run - In Two Weeks</li>
    </ul>
```

```
    </aside>
  </main>

  <footer>
    <p>&copy; 2023 The Daily Chronicle. All rights reserved.</p>
  </footer>
</body>
</html>
```

Explanation

Step 1: Document Structure

The document uses the standard HTML5 structure with <!DOCTYPE html> declaration.

The <head> section contains metadata, title, and embedded CSS styles.

The <body> contains the main content of the webpage.

Step 2: Styling (CSS)

The style section defines the layout and appearance of the page.

It uses a responsive design with max-width and flexbox for layout.

Specific styles are applied to different sections like header, nav, main content, and footer.

The color scheme is primarily black, white, and shades of gray, typical for a newspaper.

Step 3: Header and Navigation

The <header> contains the newspaper title "The Daily Chronicle".

A <nav> section provides links to different sections of the newspaper.

Step 4: Main Content

The <main> section is divided into two parts:

<section>: Contains the main articles

- Two articles are present, each in its own <article> tag

- The first article includes an image with a caption using <figure> and <figcaption>

- The second article includes a table showing sports scores

<aside>: Contains supplementary information

- Weather update

- List of upcoming events

Step 5: Articles

First article: "Breaking News: Major Technological Breakthrough"

Includes an image of a quantum computer

Second article: "Local Sports Team Wins Championship"

Includes a table with the match score

Step 6: Aside Content

Weather update

List of upcoming events

Step 7: Footer

Contains copyright information for the newspaper.

Key Features:

Responsive design: The layout adjusts based on screen size.

Semantic HTML: Uses appropriate tags like <article>, <aside>, <figure>, etc.

Flexbox layout: Used for the main content area to create columns.

Styled tables: Used in the sports article to display scores.

External image: Incorporated in the first article.

Navigation menu: Provides links to different sections of the newspaper.

This webpage demonstrates several important web design concepts:

Layout structuring using semantic HTML5 elements

CSS styling for a clean, newspaper-like appearance

Responsive design principles

Integration of various content types (text, images, tables)

Use of flexbox for modern layout techniques.

Experiment-06

Apply HTML, CSS and JavaScript to design a simple calculator to perform the following operations: sum,product, difference, remainder, quotient, power, square-root and square.

Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Simple Calculator</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         display: flex;
         justify-content: center;
         align-items: center;
         height: 100vh;
         margin: 0;
```

```
    background-color: #f0f0f0;
}
.calculator {
    background-color: #fff;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
    padding: 20px;
    width: 300px;
}
#display {
    width: 100%;
    height: 40px;
    font-size: 1.5em;
    text-align: right;
    margin-bottom: 10px;
    padding: 5px;
    box-sizing: border-box;
}
.buttons {
```

```
    display: grid;
    grid-template-columns: repeat(4, 1fr);
    gap: 10px;
}
button  {
    padding: 10px;
    font-size: 1.2em;
    border: none;
    background-color: #e0e0e0;
    cursor: pointer;
    border-radius: 4px;
}
button:hover {
    background-color: #d0d0d0;
}
.operator {
    background-color: #f0a030;
    color: white;
}
```

```
        .operator:hover {
            background-color: #e09020;
        }
    </style>
</head>
<body>
    <div class="calculator">
        <input type="text" id="display" readonly>
        <div class="buttons">
            <button onclick="appendToDisplay('7')">7</button>
            <button onclick="appendToDisplay('8')">8</button>
            <button onclick="appendToDisplay('9')">9</button>
            <button class="operator"
onclick="setOperation('+')">&plus;</button>
            <button onclick="appendToDisplay('4')">4</button>
            <button onclick="appendToDisplay('5')">5</button>
            <button onclick="appendToDisplay('6')">6</button>
            <button class="operator" onclick="setOperation('-
')">&minus;</button>
            <button onclick="appendToDisplay('1')">1</button>
```

```
        <button onclick="appendToDisplay('2')">2</button>

        <button onclick="appendToDisplay('3')">3</button>

        <button class="operator"
onclick="setOperation('*')">&times;</button>

        <button onclick="appendToDisplay('0')">0</button>

        <button onclick="appendToDisplay('.')">.</button>

        <button class="operator" onclick="calculate()">&equals;</button>

        <button class="operator"
onclick="setOperation('/')">&divide;</button>

        <button class="operator" onclick="setOperation('%')">%</button>

        <button class="operator"
onclick="setOperation('^')">x<sup>y</sup></button>

        <button class="operator" onclick="squareRoot()">√</button>

        <button class="operator"
onclick="square()">x<sup>2</sup></button>

        <button onclick="clearDisplay()">C</button>

      </div>

   </div>


   <script>
```

```
let display = document.getElementById('display');

let currentValue = '';

let operation = '';

let previousValue = '';


function appendToDisplay(value) {

    currentValue += value;

    display.value = currentValue;

}


function clearDisplay() {

    currentValue = '';

    operation = '';

    previousValue = '';

    display.value = '';

}


function setOperation(op) {

    if (currentValue !== '') {
```

```
        if (previousValue !== '') {
            calculate();
        }
        operation = op;
        previousValue = currentValue;
        currentValue = '';
    }
}

function calculate() {
    if (previousValue !== '' && currentValue !== '') {
        let result;
        const prev = parseFloat(previousValue);
        const current = parseFloat(currentValue);
        switch(operation) {
            case '+':
                result = prev + current;
                break;
            case '-':
```

```
            result = prev - current;
            break;
        case '*':
            result = prev * current;
            break;
        case '/':
            result = prev / current;
            break;
        case '%':
            result = prev % current;
            break;
        case '^':
            result = Math.pow(prev, current);
            break;
    }
    display.value = result;
    previousValue = result.toString();
    currentValue = '';
    operation = '';
```

```
        }
    }

    function squareRoot() {
        if (currentValue !== '') {
            const result = Math.sqrt(parseFloat(currentValue));
            display.value = result;
            currentValue = result.toString();
        }
    }

    function square() {
        if (currentValue !== '') {
            const result = Math.pow(parseFloat(currentValue), 2);
            display.value = result;
            currentValue = result.toString();
        }
    }
</script>
```

</body>

</html>

Explanation

Step 1: Document Structure

The document uses the standard HTML5 structure.

The <head> section contains metadata, title, and embedded CSS styles.

The <body> contains the calculator interface and JavaScript code.

Step 2: Styling (CSS)

The calculator is centered on the page using flexbox.

The calculator has a white background with rounded corners and a shadow.

Buttons are arranged in a grid layout.

Different styles are applied to number buttons and operator buttons.

Step 3: HTML Structure

The calculator is contained in a div with class "calculator".

It has an input field for display and a div for buttons.

Buttons are created for numbers 0-9, decimal point, operators (+, -, *, /), equals, clear, and additional functions (%, ^, √, x^2).

Step 4: JavaScript Functionality

The script defines several functions:

appendToDisplay(): Adds numbers and decimal point to the display.

clearDisplay(): Clears the calculator.

setOperation(): Sets the current operation.

calculate(): Performs the calculation based on the set operation.

squareRoot(): Calculates the square root of the current value.

square(): Calculates the square of the current value.

Step 5: Calculator Logic

The calculator uses three main variables: currentValue, previousValue, and operation.

When a number is clicked, it's appended to the currentValue.

When an operation is clicked, it stores the currentValue as previousValue and sets the operation.

The equals button triggers the calculate() function, which performs the operation and displays the result.

Step 6: Advanced Operations

The calculator includes modulo (%), exponentiation (^), square root (√), and square (x^2) operations.

These operations are handled in the calculate() function or as separate functions (squareRoot() and square()).

Key Features:

Basic arithmetic operations: addition, subtraction, multiplication, division.

Advanced operations: modulo, exponentiation, square root, square.

Clear button to reset the calculator.

Responsive design that works on various screen sizes.

Visual feedback with button hover effects.

This calculator demonstrates several important web development concepts:

DOM manipulation using JavaScript.

Event handling for button clicks.

Use of CSS Grid for layout.

Implementing mathematical operations in JavaScript.

Creating a responsive user interface.

Experiment-07

Develop JavaScript program (with HTML/CSS) for:

a) Converting JSON text to JavaScript Object

b) Convert JSON results into a date

c) Converting From JSON To CSV and CSV to JSON

d) Create hash from string using crypto.createHash() method.

Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>JSON/CSV Converter and Hash Generator</title>
   <script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.1.1/crypto-js.min.js"></script>
   <style>
      body {
         font-family: Arial, sans-serif;
```

```
    line-height: 1.6;
    margin: 0;
    padding: 20px;
    background-color: #f4f4f4;
}
.container {
    max-width: 800px;
    margin: auto;
    background: white;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
}
h1 {
    color: #333;
}
textarea  {
    width: 100%;
    height: 100px;
```

```
    margin-bottom: 10px;
}
button {
    background-color: #4CAF50;
    color: white;
    padding: 10px 15px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    margin-right: 10px;
}
button:hover {
    background-color: #45a049;
}
#result {
    margin-top: 20px;
    padding: 10px;
    background-color: #e7e7e7;
    border-radius: 4px;
```

```
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>JSON/CSV Converter and Hash Generator</h1>

        <h2>a) Convert JSON to JavaScript Object</h2>
        <textarea id="jsonInput" placeholder="Enter JSON here"></textarea>
        <button onclick="convertJsonToObject()">Convert to Object</button>

        <h2>b) Convert JSON to Date</h2>
        <textarea id="jsonDateInput" placeholder='Enter JSON date string (e.g., {"date": "2023-05-15T12:00:00Z"})'></textarea>
        <button onclick="convertJsonToDate()">Convert to Date</button>

        <h2>c) Convert JSON to CSV and CSV to JSON</h2>
        <textarea id="dataInput" placeholder="Enter JSON or CSV here"></textarea>
        <button onclick="convertJsonToCsv()">JSON to CSV</button>
```

```
    <button onclick="convertCsvToJson()">CSV to JSON</button>


    <h2>d) Create Hash from String</h2>

    <textarea id="hashInput" placeholder="Enter string to
hash"></textarea>

    <button onclick="createHash()">Generate Hash</button>


    <div id="result"></div>

  </div>


  <script>

    function convertJsonToObject() {

      try {

        const jsonInput = document.getElementById('jsonInput').value;

        const jsObject = JSON.parse(jsonInput);

        document.getElementById('result').innerText = 'Converted Object: '
+ JSON.stringify(jsObject, null, 2);

      } catch (error) {

        document.getElementById('result').innerText = 'Error: ' +
error.message;
```

```
        }
    }

    function convertJsonToDate() {
        try {
            const jsonInput = document.getElementById('jsonDateInput').value;
            const jsObject = JSON.parse(jsonInput);
            const date = new Date(jsObject.date);
            document.getElementById('result').innerText = 'Converted Date: ' +
date.toString();
        } catch (error) {
            document.getElementById('result').innerText = 'Error: ' +
error.message;
        }
    }

    function convertJsonToCsv() {
        try {
            const jsonInput = document.getElementById('dataInput').value;
            const jsObject = JSON.parse(jsonInput);
```

```
            const headers = Object.keys(jsObject[0]);
            const csvRows = [
               headers.join(','),
               ...jsObject.map(row => headers.map(fieldName =>
JSON.stringify(row[fieldName])).join(','))
            ];
            const csvString = csvRows.join('\n');
            document.getElementById('result').innerText = 'Converted CSV:\n'
+ csvString;
         } catch (error) {
            document.getElementById('result').innerText = 'Error: ' +
error.message;
         }
      }

      function convertCsvToJson() {
         try {
            const csvInput = document.getElementById('dataInput').value;
            const lines = csvInput.split('\n');
            const headers = lines[0].split(',');
```

```
            const jsonArray = lines.slice(1).map(line => {
                const values = line.split(',');
                return headers.reduce((obj, header, index) => {
                    obj[header] = values[index];
                    return obj;
                }, {});
            });
            document.getElementById('result').innerText = 'Converted
JSON:\n' + JSON.stringify(jsonArray, null, 2);
        } catch (error) {
            document.getElementById('result').innerText = 'Error: ' +
error.message;
        }
    }

    function createHash() {
        try {
            const input = document.getElementById('hashInput').value;
            const hash = CryptoJS.SHA256(input);
```

```
            document.getElementById('result').innerText = 'Generated Hash
(SHA-256): ' + hash;
        } catch (error) {
            document.getElementById('result').innerText = 'Error: ' +
error.message;
        }
    }
  </script>
</body>
</html>
```

Explanation

This is a tool that helps convert data between different formats (JSON and CSV) and create hash values from text. It's designed to be user-friendly and educational.

Main features:

a) Convert JSON to JavaScript Object

b) Convert JSON to Date

c) Convert JSON to CSV and CSV to JSON

d) Create a Hash from a String

Structure of the page:

The page is divided into sections, each with a text area for input and a button to perform the conversion or operation.

How it works:

Users can paste their data into the text areas and click the corresponding buttons to see the results. The converted data or hash appears in a result section at the bottom of the page.

Technologies used:

HTML: For structuring the webpage

CSS: For styling and making the page look nice

JavaScript: For performing the conversions and interactions

CryptoJS library: For creating hash values

Now, let's explain each main feature in simple terms:

a) Convert JSON to JavaScript Object:

This takes JSON (a way to represent data as text) and turns it into a JavaScript object that can be used in code.

b) Convert JSON to Date:

This takes a date written in JSON format and converts it into a regular date that's easier to read and use.

c) Convert JSON to CSV and CSV to JSON:

JSON to CSV: This takes data in JSON format and turns it into CSV (Comma-Separated Values), which is like a simple spreadsheet.

CSV to JSON: This does the opposite, taking CSV data and turning it into JSON.

d) Create Hash from String:

This takes any text you enter and creates a unique "fingerprint" (hash) for that text using a method called SHA-256.

Experiment-08

a. Develop a PHP program (with HTML/CSS) to keep track of the number of visitors visiting the web page and to display this count of visitors, with relevant headings.

b. Develop a PHP program (with HTML/CSS) to sort the student records which are stored in the database using selection sort.

Program:

Index.php

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Visitor Counter</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         line-height: 1.6;
         margin: 0;
         padding: 20px;
```

```
    background-color: #f4f4f4;
}
.container {
    max-width: 600px;
    margin: auto;
    background: white;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
}
h1 {
    color: #333;
    text-align: center;
}
.counter {
    font-size: 24px;
    text-align: center;
    margin-top: 20px;
}
```

```
    </style>
</head>
<body>
    <div class="container">
        <h1>Welcome to Our Website</h1>
        <div class="counter">
            <?php
            $counterFile = 'visitor_count.txt';

            // Read the current count
            if (file_exists($counterFile)) {
                $count = (int)file_get_contents($counterFile);
            } else {
                $count = 0;
            }

            // Increment the count
            $count++;
```

```
        // Save the new count
        file_put_contents($counterFile, $count);

        // Display the count
        echo "<h2>Visitor Count</h2>";
        echo "<p>You are visitor number: $count</p>";
        ?>
      </div>
   </div>
</body>
</html>
```

student_records.php

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

```
<style>
    body {
        font-family: Arial, sans-serif;
        line-height: 1.6;
        margin: 0;
        padding: 20px;
        background-color: #f4f4f4;
    }
    .container {
        max-width: 800px;
        margin: auto;
        background: white;
        padding: 20px;
        border-radius: 5px;
        box-shadow: 0 0 10px rgba(0,0,0,0.1);
    }
    h1 {
        color: #333;
        text-align: center;
```

```
        }
        table {
            width: 100%;
            border-collapse: collapse;
            margin-top: 20px;
        }
        th, td {
            padding: 10px;
            border: 1px solid #ddd;
            text-align: left;
        }
        th {
            background-color: #f2f2f2;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Student Records</h1>
```

```
<?php

// Database connection details

$host = 'localhost';

$dbname = 'student_records';

$username = 'your_username';

$password = 'your_password';


try {

    $pdo = new PDO("mysql:host=$host;dbname=$dbname", $username,
$password);

    $pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);


    // Fetch student records

    $stmt = $pdo->query("SELECT * FROM students");

    $students = $stmt->fetchAll(PDO::FETCH_ASSOC);


    // Selection sort function

    function selectionSort(&$arr, $n) {

        for ($i = 0; $i < $n - 1; $i++) {
```

```
        $min_idx = $i;
        for ($j = $i + 1; $j < $n; $j++) {
            if ($arr[$j]['gpa'] < $arr[$min_idx]['gpa']) {
                $min_idx = $j;
            }
        }
        if ($min_idx != $i) {
            $temp = $arr[$i];
            $arr[$i] = $arr[$min_idx];
            $arr[$min_idx] = $temp;
        }
    }
}

// Sort students by GPA
selectionSort($students, count($students));

// Display sorted student records
echo "<table>";
```

```
        echo "<tr><th>ID</th><th>Name</th><th>GPA</th></tr>";
        foreach ($students as $student) {
            echo "<tr>";
            echo "<td>" . htmlspecialchars($student['id']) . "</td>";
            echo "<td>" . htmlspecialchars($student['name']) . "</td>";
            echo "<td>" . htmlspecialchars($student['gpa']) . "</td>";
            echo "</tr>";
        }
        echo "</table>";

    } catch(PDOException $e) {
        echo "Connection failed: " . $e->getMessage();
    }
    ?>
  </div>
</body>
</html>
```

Explanation

Visitor Counter:

This is a simple webpage that keeps track of how many visitors have accessed the site.

HTML structure for the page layout

CSS for styling (making it look nice)

PHP code to count and display visitors

How it works:

The page reads a number from a file (visitor_count.txt).

It adds 1 to this number.

It saves the new number back to the file.

It displays the new number on the page.

This gives visitors a sense of how many people have visited the site before them.

Student Record Sorter:

This page displays a sorted list of student records, ordered by their GPAs.

Key components:

HTML structure for the page layout

CSS for styling

PHP code to connect to a database, retrieve student records, sort them, and display them

How it works:

The page connects to a database containing student information.

It retrieves all student records from the database.

It uses a sorting algorithm called Selection Sort to order the students by their GPAs.

It displays the sorted list in a table on the webpage.

The sorting algorithm (Selection Sort):

This method goes through the list multiple times.

Each time, it finds the student with the lowest GPA that hasn't been placed in order yet.

It then puts this student at the beginning of the unsorted part of the list.

This process repeats until all students are in order.

Important learning points:

Database Interaction: The code demonstrates how to connect to and retrieve data from a MySQL database using PHP.

Sorting Algorithms: It implements the Selection Sort algorithm, which is a fundamental concept in computer science.

Web Development: The code shows how to combine HTML, CSS, and PHP to create dynamic web pages.

Security: The code uses htmlspecialchars() to prevent XSS (Cross-Site Scripting) attacks when displaying data.

Experiment-09

Develop jQuery script (with HTML/CSS) for:

a. Appends the content at the end of the existing paragraph and list.

b. Change the state of the element with CSS style using animate() method

c. Change the color of any div that is animated.

Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>jQuery Append, Animate, and Color Change Demo</title>
   <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
   <style>
      body {
         font-family: Arial, sans-serif;
         line-height: 1.6;
         margin: 0;
```

```
    padding: 20px;
    background-color: #f4f4f4;
}
.container {
    max-width: 800px;
    margin: auto;
    background: white;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
}
h1, h2 {
    color: #333;
}
.box {
    width: 100px;
    height: 100px;
    background-color: #3498db;
    margin: 20px 0;
```

```
        }
        button {
            padding: 10px 15px;
            background-color: #2ecc71;
            color: white;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            margin-right: 10px;
        }
        button:hover {
            background-color: #27ae60;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>jQuery Demonstration</h1>
```

```
    <h2>a. Append Content</h2>

    <p id="existingParagraph">This is an existing paragraph. </p>

    <ul id="existingList">

      <li>Existing item 1</li>

      <li>Existing item 2</li>

    </ul>

    <button id="appendButton">Append Content</button>


    <h2>b. Animate Element</h2>

    <div id="animateBox" class="box"></div>

    <button id="animateButton">Animate Box</button>


    <h2>c. Change Color of Animated Div</h2>

    <div id="colorBox" class="box"></div>

    <button id="colorAnimateButton">Animate and Change
Color</button>

  </div>


  <script>
```

```
$(document).ready(function() {
    // a. Append content
    $("#appendButton").click(function() {
        $("#existingParagraph").append("This content is appended.");
        $("#existingList").append("<li>Appended item</li>");
    });

    // b. Animate element
    $("#animateButton").click(function() {
        $("#animateBox").animate({
            width: "200px",
            height: "200px",
            opacity: 0.5
        }, 1000);
    });

    // c. Animate and change color
    $("#colorAnimateButton").click(function() {
        $("#colorBox").animate({
```

```
            width: "200px",
            height: "200px"
        }, {
            duration: 1000,
            step: function(now, fx) {
                if (fx.prop === "width") {
                    $(this).css("background-color", `rgb(${Math.round(now)},
52, 219)`);
                }
            }
        });
    });
  </script>
</body>
</html>
```

Explanation

Purpose of the webpage:

This is a demonstration page showing how to use jQuery, a popular JavaScript library, to manipulate web page elements dynamically. It includes examples of appending content, animating elements, and changing colors.

Structure of the page:

The page is divided into three main sections, each demonstrating a different jQuery functionality:

a) Append Content

b) Animate Element

c) Change Color of Animated Div

Technologies used:

HTML: For structuring the webpage

CSS: For styling the page elements

jQuery: A JavaScript library for easier DOM manipulation and animations

Key components and their functions:

a) Append Content:

This section has an existing paragraph and a list.

When you click the "Append Content" button, it adds text to the paragraph and a new item to the list.

How it works: jQuery selects the elements by their IDs and uses the .append() method to add new content.

b) Animate Element:

This section has a blue square (box).

Clicking the "Animate Box" button makes the box grow larger and more transparent.

How it works: jQuery uses the .animate() method to gradually change the box's width, height, and opacity over 1 second (1000 milliseconds).

c) Change Color of Animated Div:

This section also has a blue square.

Clicking the "Animate and Change Color" button makes the box grow larger while changing its color from blue to red.

How it works: jQuery animates the box's size and simultaneously updates its background color based on the current width during the animation.

jQuery setup:

The script uses $(document).ready(function() { ... }) to ensure that the jQuery code only runs after the page has fully loaded.

Event handling:

Each button has a click event handler set up using jQuery's .click() method.

When a button is clicked, it triggers the corresponding function to perform the animation or content append.

Experiment-10

Develop a JavaScript program with Ajax (with HTML/CSS) for:

a. Use ajax() method (without Jquery) to add the text content from the text file by sending ajax request.

b. Use ajax() method (with Jquery) to add the text content from the text file by sending ajax request.

c. Illustrate the use of getJSON() method in jQuery

d. Illustrate the use of parseJSON() method to display JSON values.

Program:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ajax Demo Program</title>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js"></scri
pt>
    <style>
```

```
body {
    font-family: Arial, sans-serif;
    line-height: 1.6;
    margin: 0;
    padding: 20px;
    background-color: #f4f4f4;
}
.container {
    max-width: 800px;
    margin: auto;
    background: white;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
}
h1 {
    color: #333;
}
h2 {
```

```
    color: #666;
}
button {
    background-color: #4CAF50;
    border: none;
    color:  white;
    padding: 10px 20px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
    margin: 4px 2px;
    cursor: pointer;
    border-radius: 4px;
}
pre {
    background-color: #f8f8f8;
    border: 1px solid #ddd;
    border-radius: 4px;
```

```
            padding: 10px;
            white-space: pre-wrap;
            word-wrap: break-word;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Ajax Demo Program</h1>

        <h2>a. Ajax-like operation without jQuery</h2>
        <button onclick="operationWithoutJQuery()">Perform Operation (without jQuery)</button>
        <pre id="result-a"></pre>

        <h2>b. Ajax-like operation with jQuery</h2>
        <button onclick="operationWithJQuery()">Perform Operation (with jQuery)</button>
        <pre id="result-b"></pre>
```

```
    <h2>c. jQuery-like getJSON() method</h2>
    <button onclick="getJSONOperation()">Get JSON</button>
    <pre id="result-c"></pre>

    <h2>d. jQuery parseJSON() method</h2>
    <button onclick="parseJSONExample()">Parse JSON</button>
    <pre id="result-d"></pre>
</div>

<script>
    // Simulated data
    const simulatedData = {
        text: "This is a sample text from a simulated server response.",
        json: {
            name: "John Doe",
            age: 30,
            city: "New York"
        }
    };
```

```
    // a. Ajax-like operation without jQuery
    function operationWithoutJQuery() {
        setTimeout(function() {
            document.getElementById("result-a").textContent =
simulatedData.text;
        }, 500);
    }

    // b. Ajax-like operation with jQuery
    function operationWithJQuery() {
        $.Deferred(function(deferred) {
            setTimeout(function() {
                deferred.resolve(simulatedData.text);
            }, 500);
        }).done(function(result) {
            $("#result-b").text(result);
        });
    }
```

```
    // c. jQuery-like getJSON() method
    function getJSONOperation() {
        $.Deferred(function(deferred) {
            setTimeout(function() {
                deferred.resolve(simulatedData.json);
            }, 500);
        }).done(function(result) {
            $("#result-c").text(JSON.stringify(result, null, 2));
        });
    }

    // d. jQuery parseJSON() method
    function parseJSONExample() {
        var jsonString = JSON.stringify(simulatedData.json);
        var jsonObject = $.parseJSON(jsonString);
        $("#result-d").text(JSON.stringify(jsonObject, null, 2));
    }
</script>
```

</body>

</html>

Explanation

- Purpose of the webpage:
- This is a demonstration page showing different methods of handling asynchronous operations and JSON data, both with and without jQuery. It's designed to help students understand the differences between these approaches.
- The page is divided into four main sections, each demonstrating a different technique:
- Ajax-like operation without jQuery
- Ajax-like operation with jQuery
- jQuery-like getJSON() method
- jQuery parseJSON() method
- Technologies used:
- HTML: For structuring the webpage
- CSS: For styling the page elements
- JavaScript: For performing operations without jQuery
- jQuery: A JavaScript library for easier Ajax and JSON handling

- Ajax-like operation without jQuery:
- This simulates an asynchronous operation using plain JavaScript.
- It uses setTimeout() to mimic a delay in receiving data from a server.
- The result is displayed using vanilla JavaScript DOM manipulation.

- Ajax-like operation with jQuery:
- This demonstrates how to use jQuery's Deferred object to handle asynchronous operations.
- It also uses setTimeout() to simulate a delay.
- The result is displayed using jQuery's DOM manipulation methods.
- jQuery-like getJSON() method:
- This simulates fetching JSON data from a server using jQuery's Deferred object.
- It demonstrates how to handle and display JSON data.
- jQuery parseJSON() method:
- This shows how to parse a JSON string into a JavaScript object using jQuery.
- It then displays the parsed object.
- The code uses a simulatedData object to mimic data that would typically come from a server.
- This allows the demonstration to work without an actual server connection.
- Asynchronous Operations:
- All the operations (except the parseJSON example) use setTimeout() to simulate the delay that would occur when fetching data from a real server.
- This helps students understand the concept of asynchronous programming.
- jQuery Usage:
- The code demonstrates how jQuery can simplify Ajax-like operations and JSON handling.
- It shows the difference between using vanilla JavaScript and jQuery for similar tasks.