

Introduction to Web Technology Lab

What is the Internet?

The internet is a global network of interconnected computers and servers that communicate with each other using standardized protocols (most commonly TCP/IP).

It is a vast infrastructure that allows for the transmission of data across networks worldwide.

- **Components:**
 - **Hardware:** Includes servers, routers, switches, cables, and data centers that facilitate the transmission of data.
 - **Protocols:** Sets of rules that define how data is transmitted over the network (e.g., TCP/IP, HTTP, FTP).
 - **Connectivity:** Encompasses various types of connections, including wired (Ethernet) and wireless (Wi-Fi, cellular) connections.
- **Functionality:** The internet allows devices to communicate with each other, enabling services like email, file sharing, online gaming, and video conferencing.

What is the Web (World Wide Web)?

The web, or World Wide Web (WWW), is a system of interlinked hypertext documents and multimedia content accessed via the internet using web browsers. It is an application that runs on the internet, providing a user-friendly interface for accessing information.

- **Components:**
 - **Web Pages:** Documents that can include text, images, videos, and links to other documents, typically written in HTML (HyperText Markup Language).
 - **Web Browsers:** Software applications (like Google Chrome, Firefox, and Safari) that allow users to access and interact with web pages.
 - **Web Servers:** Computers that store and serve web content to users over the internet.

- **Functionality:** The web enables users to access information, interact with applications, and engage in social networking, e-commerce, and more through a graphical interface.

Key Differences

Feature	Internet	Web
Nature	A global network of networks	A collection of interconnected documents and resources
Function	Facilitates communication and data transfer	Provides access to information through web pages
Protocols	Uses various protocols like TCP/IP, HTTP, FTP	Primarily relies on HTTP/HTTPS for communication
Access	Requires network connectivity (ISP)	Accessed via web browsers over the internet

What is web programming ?

Web programming, often referred to as web development, is the process of creating dynamic websites and web applications.

It encompasses a variety of technologies and programming languages used to build the front-end (client-side) and back-end (server-side) components of a web application.

Front-End Development

Front-end development involves everything that users interact with directly in their web browsers. It focuses on the visual and interactive aspects of a website.

- **HTML (HyperText Markup Language):** The standard markup language used to create the structure of web pages. HTML elements are the building blocks of web content.

- **CSS (Cascading Style Sheets):** A stylesheet language used to describe the presentation of a document written in HTML. CSS controls layout, colors, fonts, and overall visual design.
- **JavaScript:** A programming language that enables interactive features on web pages, such as form validation, animations, and dynamic content updates. JavaScript frameworks (like React, Angular, and Vue.js) are also widely used for building complex user interfaces.

Back-End Development

Back-end development involves server-side programming, managing the database, and handling the application logic that processes user requests.

- **Server-Side Languages:** Common languages include PHP, Python, Ruby, Node.js, and Java. These languages manage database interactions, user authentication, and server logic.
- **Databases:** Web applications often require data storage and retrieval. Databases can be relational (like MySQL, PostgreSQL) or non-relational (like MongoDB, Firebase). They store user data, application data, and other content.
- **Web Servers:** Web servers (such as Apache, Nginx, and Microsoft IIS) handle requests from clients (web browsers), serve web pages, and manage the server-side logic of applications.

Experiment-1

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 (, <u>, <sub>, <sup> etc.)

Step-by-Step Instructions: Open a text editor: You can use any text editor like Notepad, Sublime Text, Visual Studio Code, etc. Create a new HTML file: Save it with the name Myfirstwebpage.html.

Source Code:

```
<!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>
</head>
<body>

  <!-- Moving text -->
  <marquee>Basic HTML Tags</marquee>

  <!-- Different heading tags -->
  <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>

  <!-- Paragraph -->
  <p>This is a paragraph of text. HTML paragraphs are defined with the &lt;p> tag.</p>
```

<!-- Horizontal line -->

<hr>

<!-- Line break -->

<p>This is some text before a line break.
This is the text after the line break.</p>

<!-- Block quote -->

<blockquote>

"This is a blockquote. It is used to quote sections of text from another source."

</blockquote>

<!-- Preformatted text -->

<pre>

This is preformatted text.

It preserves both spaces and line breaks.

</pre>

<!-- Different logical styles -->

<p>This is bold text.</p>

<p>This is <i>italicized</i> text.</p>

<p>This is <u>underlined</u> text.</p>

<p>This is ^{superscript} text.</p>

<p>This is _{subscript} text.</p>

<p>This is emphasized text.</p>

<p>This is strong text.</p>

<p>This is <mark>highlighted</mark> text.</p>

<p>This is <small>small</small> text.</p>

<p>This is deleted text.</p>

<p>This is <ins>inserted</ins> text.</p>

<p>This is <code>inline code</code> text.</p>

</body>

</html>

Explanation of the HTML Code:

1. <title>: Sets the title of the webpage as "My First Web Page," which appears in the browser tab.
2. <marquee>: Displays moving text "Basic HTML Tags."
3. **Heading Tags** (<h1> to <h6>): Represents different levels of headings from largest (<h1>) to smallest (<h6>).
4. <p> (**Paragraph**): Adds a paragraph of text.
5. <hr> (**Horizontal Line**): Creates a horizontal line across the page.

6.
 (**Line Break**): Adds a line break without starting a new paragraph.
7. <blockquote>: Represents a section that is quoted from another source.
8. <pre>: Displays preformatted text where whitespace and line breaks are preserved.
9. **Logical Style Tags**:
 1. : Makes text bold.
 2. <i>: Makes text italic.
 3. <u>: Underlines text.
 4. <sub>: Makes text subscript (e.g., H₂O).
 5. <sup>: Makes text superscript (e.g., E=mc²).
 6. : Makes text bold and indicates strong importance.
 7. : Emphasizes text with italics.

Output:

This is an H1 heading

This is an H2 heading

This is an H3 heading

This is an H4 heading

This is an H5 heading

This is an H6 heading

This is a paragraph demonstrating the use of the paragraph tag in HTML.

This is a line of text before the break.

This is a line of text after the break.

This is a blockquote. It is used to display a quotation or excerpt from another source.

This is preformatted text.
It preserves spaces and line breaks.

This is **bold** text.

This is *italicized* text.

This is underlined text.

This is ^{superscript} text.

This is _{subscript} text.

This is *emphasized* text.

This is **strong** text.

This is highlighted text.

This is small text.

This is ~~deleted~~ text.

This is inserted text.

This is inline code text.

Experiment-2

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 colors.)

c) Provide colour options for rows.

Step-by-Step Instructions:

1. **Open a text editor:** Use any text editor like Notepad, Sublime Text, Visual Studio Code, etc.
2. **Create a new HTML file:** Save it with the name Table.html.
3. **Write the HTML code:** Below is the HTML code that includes the table structure, title, header, footer, row-span, col-span, and color options for cells and rows.

Source Code :

```
<!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>
</head>
<body>
  <h1 style="text-align: center;">Class Time Table</h1>

  <table border="1" width="100%" cellpadding="10">
    <!-- Table Header -->
    <thead>
      <tr>
        <th>Day</th>
        <th>9:00 - 10:00</th>
        <th>10:00 - 11:00</th>
        <th>11:00 - 12:00</th>
        <th>12:00 - 1:00</th>
        <th colspan="2">1:00 - 2:00</th>
        <th>2:00 - 3:00</th>
        <th>3:00 - 4:00</th>
      </tr>
```

</thead>

<!-- Table Body -->

<tbody>

<tr style="background-color: #f2f2f2;">

<td>Monday</td>

<td>Math</td>

<td style="background-color: #ffc107;">Lab</td>

<td>Physics</td>

<td>Computer Science</td>

<td rowspan="5" colspan="2" style="background-color: #e3f2fd; text-align:center;">Lunch Break</td>

<td style="background-color: #ffeb3b;">Elective</td>

<td>English</td>

</tr>

<tr style="background-color: #e3f2fd;">

<td>Tuesday</td>

<td>Biology</td>

<td>Math</td>

<td style="background-color: #ffc107;">Lab</td>

<td>Physics</td>

<td>History</td>

<td>Geography</td>

</tr>

<tr style="background-color: #f2f2f2;">

<td>Wednesday</td>

<td>English</td>

<td>Math</td>

<td style="background-color: #ffeb3b;">Elective</td>

<td>Computer Science</td>

<td>Physics</td>

<td style="background-color: #ffc107;">Lab</td>

</tr>

<tr style="background-color: #e3f2fd;">

<td>Thursday</td>

<td>Geography</td>

<td style="background-color: #ffc107;">Lab</td>

<td>Math</td>

<td>Biology</td>

<td>History</td>

<td>Physics</td>

</tr>

<tr style="background-color: #f2f2f2;">


```
<td>Friday</td>
<td>History</td>
<td>English</td>
<td>Computer Science</td>
<td style="background-color: #ffeb3b;">Elective</td>
<td>Biology</td>
<td>Math</td>
</tr>
</tbody>

<!-- Table Footer -->
<tfoot>
<tr>
<td colspan="9" style="text-align: center;">Note: Lab and Elective hours are
highlighted with different colors</td>
</tr>
</tfoot>
</table>
</body>
</html>
```

Explanation of the HTML Code:

1. `<title>`: Sets the title of the webpage as "Time Table," which appears in the browser tab.
2. `<style>`: Adds CSS styles directly within the HTML document to style the table, headers, footers, and cells. The styles include:
 - a) `table`: Styles the table to be centered, collapsed borders, and width set to 80%.
 - b) `th, tfoot td`: Styles for the table header and footer with background color, padding, and border.
 - c) `td`: Styles for the table cells with padding, text alignment, and border.
 - d) `.lab`: Applies a light red background color for lab hours.
 - e) `.elective`: Applies a light green background color for elective hours.
 - f) `tr:nth-child(even)` and `tr:nth-child(odd)`: Alternates row colors for better readability.
3. `<table>`: Defines the table structure.
4. `<thead>`: Contains the table header with column titles.
 - a) The first row in the header has a single cell that spans all columns (`colspan="7"`) for the table title.
5. `<tbody>`: Contains the table body with the timetable data.
 1. The class attributes (`.lab` and `.elective`) apply different colors to specific cells to highlight lab and elective hours.
 2. The `colspan` attribute merges multiple columns into a single cell for lab hours spanning two periods.

BCSL504| Web Technology Lab

3. The rowspan attribute merges multiple rows into a single cell for lab hours spanning two rows.

6. <tfoot>: Contains the table footer with a note explaining the highlighted cells.

Output :

Time Table

Day/Time	9:00 - 10:00	10:00 - 11:00	11:00 - 12:00	12:00 - 1:00	Lunch Break	2:00 - 3:00	3:00 - 4:00
Monday	Math	English	Physics Lab	Elective	Break	Elective	History
Tuesday	Elective	Biology	Math	Chemistry Lab		Geography	PE
Wednesday	History	Computer Lab	English	Math		Physics	Elective
Thursday	PE	History	Geography	Elective		Biology	Math
Friday	Biology Lab	Math	English	Physics		Elective	Chemistry
End of Timetable							

Experiment- 3

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.

Step-by-Step Instructions:

1. **Create the HTML file:** Save it as index.html.
2. **Create the CSS file:** Save it as style.css.
3. **Link the CSS file to the HTML file:** Use the <link> tag in the <head> section of the HTML file to link the external stylesheet.

Source Code :

Html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Sample HTML with External CSS</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>

  <h2>Welcome to My Website</h2>
  <p>This paragraph comes right after the heading.</p>

  <h3>About Me</h3>
  <p>I am a web developer with a passion for creating interactive websites.</p>
  <hr>

  <div id="special-section">
    <h2>Special Section</h2>
    <p>This section has a special background and border.</p>
  </div>

  <div>
    <h3>My Projects</h3>
    <p class="highlight">Highlighted Project: A Dynamic Web Application.</p>
    <p>Here, I describe my projects in detail.</p>
```

```
<p><span>Note:</span> The projects are listed below.</p>
</div>
```

```
<time datetime="2024-08-20">Published on August 20, 2024</time>
```

```
<p>This is an additional paragraph to demonstrate the <b>pseudo-element</b> before the
text.</p>
```

```

```

```
<p>Visit my <a href="https://example.com" target="_blank">portfolio</a> to see more.</p>
```

```
</body>
</html>
```

style.css

```
/* Style for h2 tags */
```

```
h2 {
  color: #2E8B57;
  font-family: 'Arial', sans-serif;
  text-align: center;
  border-bottom: 2px solid #2E8B57;
  padding-bottom: 10px;
}
```

```
/* Style for h3 tags */
```

```
h3 {
  color: #4682B4;
  font-family: 'Verdana', sans-serif;
  text-align: left;
  margin-left: 20px;
}
```

```
/* Style for hr tags */
```

```
hr {
  border: 1px solid #B22222;
  width: 80%;
  margin: 20px auto;
}
```

```
/* Style for p tags */
```

```
p {
```

```
font-size: 16px;
line-height: 1.5;
margin: 10px 20px;
color: #333333;
}

/* Style for div tags */
div {
    background-color: #F5F5F5;
    padding: 20px;
    border: 1px solid #DCDCDC;
    margin-bottom: 20px;
}

/* Style for span tags */
span {
    color: #FF4500;
    font-weight: bold;
    font-style: italic;
}

/* Style for time tags */
time {
    font-size: 14px;
    color: #696969;
}

/* Style for img tags */
img {
    border: 2px solid #708090;
    border-radius: 8px;
    max-width: 100%;
    height: auto;
}

/* Style for a (anchor) tags */
a {
    color: #1E90FF;
    text-decoration: none;
    font-weight: bold;
}

a:hover {
```

```
color: #FF4500;
text-decoration: underline;
}

/* CSS Selectors */

/* Universal selector */
* {
    box-sizing: border-box;
}

/* Class selector for highlighting important text */
.highlight {
    background-color: #FFFFE0;
    padding: 5px;
    border-radius: 3px;
}

/* ID selector for a special section */
#special-section {
    background-color: #FFFACD;
    padding: 15px;
    border: 2px solid #FFD700;
}

/* Grouping selector for tags that share styles */
h2, h3, p {
    margin: 10px;
    font-family: 'Georgia', serif;
}

/* Descendant selector for specific elements inside a div */
div p {
    color: #4B0082;
}

/* Child selector for immediate children of a div */
div > p {
    font-weight: bold;
}

/* Adjacent sibling selector for styling elements following another */
h2 + p {
```

```
color: #8B0000;
font-style: italic;
}

/* Attribute selector for targeting links that open in a new tab */
a[target="_blank"] {
    color: #8A2BE2;
}

/* Pseudo-class selector for first child elements */
p:first-child {
    color: #20B2AA;
}

/* Pseudo-element selector for adding text before content */
p::before {
    content: "Note: ";
    color: #FF4500;
    font-weight: bold;
}
```

Code Explanation of CSS Selectors:

Universal Selector (*):

1. Targets all elements on the page, used here to ensure all elements follow the box-sizing: border-box model.

Type Selectors (h2, h3, hr, etc.):

1. Apply styles to specific HTML tags like <h2>, <h3>, etc.

Class Selector (.highlight):

1. Targets elements with the class highlight. Useful for reusable styles.

ID Selector (#special-section):

1. Targets a specific element with the ID special-section. IDs should be unique on a page.

Grouping Selector (h2, h3, p):

1. Allows multiple elements to share the same styles, reducing code redundancy.

Descendant Selector (div p):

1. Styles <p> elements that are descendants of <div> elements.

Child Selector (div > p):

1. Styles <p> elements that are direct children of a <div>.

Adjacent Sibling Selector (h2 + p):

1. Targets the <p> element that directly follows an <h2>.

Attribute Selector (a[target="_blank"]):

1. Styles <a> tags that have a target="_blank" attribute, indicating links that open in a new tab.

Pseudo-class Selector (p:first-child):

1. Targets the first <p> element within its parent.

Pseudo-element Selector (p::before):

1. Inserts content before the actual content of the <p> element.

Explanation of the HTML Structure:

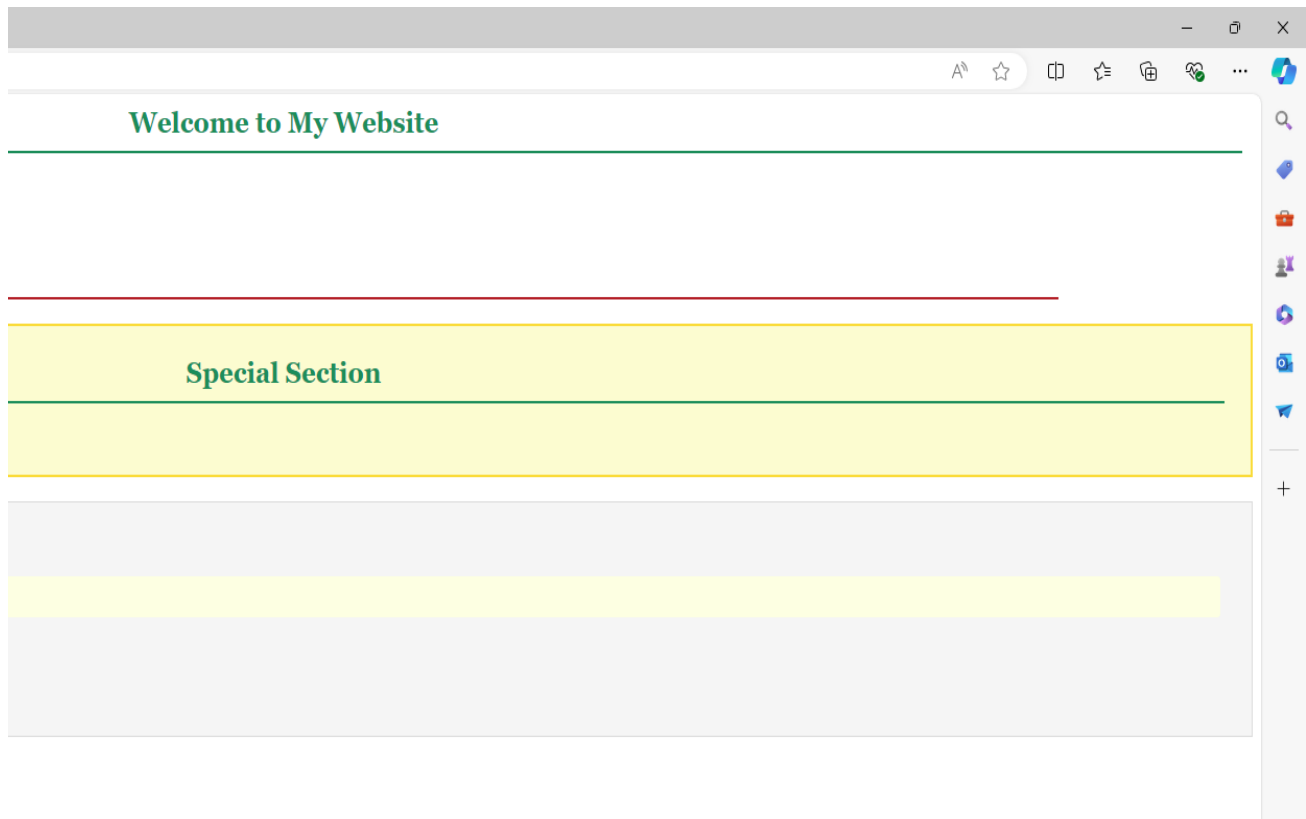
Linking the CSS:

1. The external stylesheet style.css is linked in the <head> section using <link>.

Elements Styled by the CSS:

1. **Heading (<h2> and <h3>):** The headings are styled with custom colors, alignment, and borders.
2. **Paragraphs (<p>):** Different paragraphs are used to demonstrate styling and various selectors.
3. **Horizontal Line (<hr>):** Displays a styled horizontal line.
4. **Div with ID Selector:** The <div> with the id="special-section" is styled with a unique background and border.
5. **Class Selector (.highlight):** The class .highlight is applied to a paragraph to demonstrate highlighted text.
6. **Span ():** A span inside a paragraph is styled to stand out.
7. **Time (<time>):** The <time> element is styled to display date information.
8. **Image ():** The image is styled with a border and rounded corners.
9. **Anchor (<a>):** Links are styled differently when hovered over and those that open in a new tab are highlighted.

Output:



Experiment-4

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.

Step-by-Step Instructions:

1. **Create the HTML file:** Open a text editor and save a new file as registration.html.
2. **Create the CSS styles:** Include CSS styles within the HTML file using the <style> tag to apply background colors, font colors, and sizes.

Source Code:

registration.html

```
<!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>
    /* General styles */
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
      margin: 0;
      padding: 20px;
    }
    h1 {
      text-align: center;
      color: #333333;
      font-size: 28px;
      margin-bottom: 20px;
    }
    table {
      width: 50%;
      margin: 0 auto;
      background-color: #ffffff;
```

```
border: 2px solid #ccc;
border-radius: 8px;
padding: 20px;
}

th, td {
padding: 10px;
text-align: left;
font-size: 16px;
}

th {
background-color: #4CAF50;
color: white;
font-size: 18px;
}

td {
color: #333333;
}

input[type="text"], input[type="email"], input[type="password"], input[type="date"],
input[type="number"], select {
width: 100%;
padding: 8px;
margin: 5px 0;
border: 1px solid #ccc;
border-radius: 4px;
background-color: #f9f9f9;
}

input[type="radio"], input[type="checkbox"] {
margin-right: 10px;
}

input[type="submit"], input[type="reset"] {
width: 100%;
padding: 10px;
margin-top: 10px;
border: none;
border-radius: 4px;
cursor: pointer;
font-size: 18px;
```

```
}

input[type="submit"] {
    background-color: #4CAF50;
    color: white;
}

input[type="submit"]:hover {
    background-color: #45a049;
}

input[type="reset"] {
    background-color: #f44336;
    color: white;
}

input[type="reset"]:hover {
    background-color: #e53935;
}
</style>
</head>
<body>

<h1>Registration Form</h1>

<form action="#" method="post">
    <table>
        <tr>
            <th colspan="2">Personal Information</th>
        </tr>
        <tr>
            <td>Full Name:</td>
            <td><input type="text" name="fullname" required></td>
        </tr>
        <tr>
            <td>Email:</td>
            <td><input type="email" name="email" required></td>
        </tr>
        <tr>
            <td>Password:</td>
            <td><input type="password" name="password" required></td>
        </tr>
        <tr>
            <td colspan="2"></td>
        </tr>
    </table>
</form>
```

```
<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"> Female
<input type="radio" name="gender" value="other"> Other
</td>
</tr>
<tr>
<td>Country:</td>
<td>
<select name="country" required>
<option value="india">India</option>
<option value="usa">USA</option>
<option value="canada">Canada</option>
<option value="australia">Australia</option>
</select>
</td>
</tr>
<tr>
<td>Phone Number:</td>
<td><input type="number" name="phone" required></td>
</tr>
<tr>
<th colspan="2">Preferences</th>
</tr>
<tr>
<td>Hobbies:</td>
<td>
<input type="checkbox" name="hobbies" value="reading"> Reading
<input type="checkbox" name="hobbies" value="traveling"> Traveling
<input type="checkbox" name="hobbies" value="sports"> Sports
<input type="checkbox" name="hobbies" value="music"> Music
</td>
</tr>
<tr>
<td>Receive Newsletter:</td>
<td><input type="checkbox" name="newsletter"> Yes</td>
</tr>
<tr>
```

```
<td colspan="2">
  <input type="submit" value="Register">
  <input type="reset" value="Reset">
</td>
</tr>
</table>
</form>

</body>
</html>
```

Explanation of the HTML and CSS Code:

HTML Structure:

1. **<table>**: The form is organized using a table for alignment.
2. **<thead>**: Contains the table header with a title.
3. **<tbody>**: Contains form elements such as text inputs, radio buttons, checkboxes, select dropdown, and buttons for submit and reset.
4. **<label>**: Used to describe each input field.
5. **Input Types**: Various input elements like text, email, password, date, radio, checkbox, and submit are used to create a comprehensive registration form.

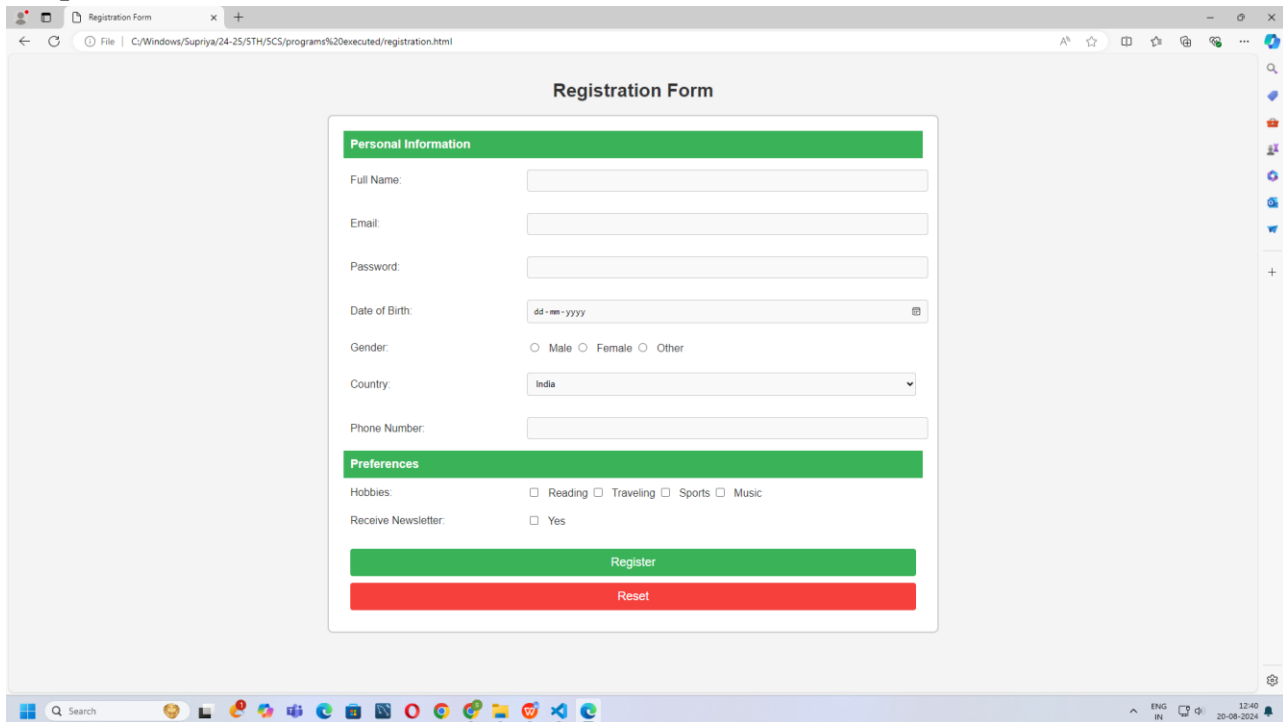
CSS Styling:

1. **Body Styling**: Sets the font, background color, text color, and padding for the entire page.
2. **Table Styling**: Styles the table with width, margin, border collapse, background color, and shadow.
3. **Form Element Styling**:
 1. **input[type="text"], input[type="email"], input[type="password"], input[type="date"], select**: Styles input fields with padding, margin, border, and font size.
 2. **Buttons**: Styles submit and reset buttons with background color, font size, and hover effects.
4. **Row and Cell Styling**: Applies padding, text alignment, and border for table cells and headers.
5. **Class and Element Selectors**: Used to style elements based on type and context, ensuring consistency and readability.

Instructions for Running the HTML File:

1. **Save the file**: Ensure the file is saved as registration.html.
2. **Open in a web browser**: Double-click the registration.html file or right-click and choose "Open with" to select a browser like Chrome, Firefox, Edge, etc.

Output:



The screenshot shows a web browser window with a single tab titled 'Registration Form'. The address bar shows the file path: 'C:/Windows/Supriya/24-25/5TH/SCS/programs%20executed/registration.html'. The page content is a registration form titled 'Registration Form'.

Registration Form

Personal Information

Full Name:

Email:

Password:

Date of Birth:

Gender: ☐ Male ☐ Female ☐ Other

Country:

Phone Number:

Preferences

Hobbies: ☐ Reading ☐ Traveling ☐ Sports ☐ Music

Receive Newsletter: ☐ Yes

Experiment-5

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

Step-by-Step Instructions

1. **Create the HTML file:** Open a text editor and save a new file as newspaper.html.
2. **Include CSS styles:** Use the <style> tag within the HTML file to define the background colors, text colors, and sizes for different semantic elements.

Source Code:

newspaper.html

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

    header {
      background-color: #4CAF50;
      color: white;
      text-align: center;
      padding: 20px;
    }

    nav {
      background-color: #333;
      color: white;
      text-align: center;
      padding: 10px;
    }
```



```
nav a {
  color: white;
  margin: 0 15px;
  text-decoration: none;
  font-size: 18px;
}

section {
  display: flex;
  padding: 20px;
}

article {
  flex: 3;
  background-color: #f4f4f4;
  padding: 20px;
  margin-right: 20px;
  border-radius: 5px;
}

aside {
  flex: 1;
  background-color: #f0e68c;
  padding: 20px;
  border-radius: 5px;
}

figure {
  text-align: center;
  margin: 20px 0;
}

figure img {
  max-width: 100%;
  height: auto;
  border-radius: 5px;
}

figure figcaption {
  font-size: 14px;
  color: #555;
}
```

```
table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
}

table, th, td {
    border: 1px solid #ddd;
}

th, td {
    padding: 10px;
    text-align: left;
}

th {
    background-color: #333;
    color: white;
}

td {
    background-color: #fff;
}

footer {
    background-color: #333;
    color: white;
    text-align: center;
    padding: 10px;
    position: fixed;
    width: 100%;
    bottom: 0;
}
</style>
</head>
<body>

<header>
    <h1>Daily News</h1>
    <p>Your source for the latest news</p>
</header>

<nav>
```

```
<a href="#">Home</a>
<a href="#">World</a>
<a href="#">Politics</a>
<a href="#">Technology</a>
<a href="#">Sports</a>
</nav>

<section>
  <article>
    <h2>Main Article</h2>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.</p>
    <figure>
      
      <figcaption>Image caption goes here</figcaption>
    </figure>
    <p>Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip
ex ea commodo consequat.</p>

    <h3>Table of Data</h3>
    <table>
      <tr>
        <th>Category</th>
        <th>Details</th>
      </tr>
      <tr>
        <td>World</td>
        <td>Global news updates</td>
      </tr>
      <tr>
        <td>Politics</td>
        <td>Political insights and analysis</td>
      </tr>
      <tr>
        <td>Technology</td>
        <td>Latest tech trends</td>
      </tr>
      <tr>
        <td>Sports</td>
        <td>Sports news and scores</td>
      </tr>
    </table>
  </article>
```

```
<aside>
  <h2>Related Articles</h2>
  <ul>
    <li><a href="#">Article 1</a></li>
    <li><a href="#">Article 2</a></li>
    <li><a href="#">Article 3</a></li>
  </ul>
</aside>
</section>

<footer>
  <p>&copy; 2024 Daily News. All rights reserved.</p>
</footer>

</body>
</html>
```

Explanation of the HTML and CSS Code

HTML Semantic Elements:

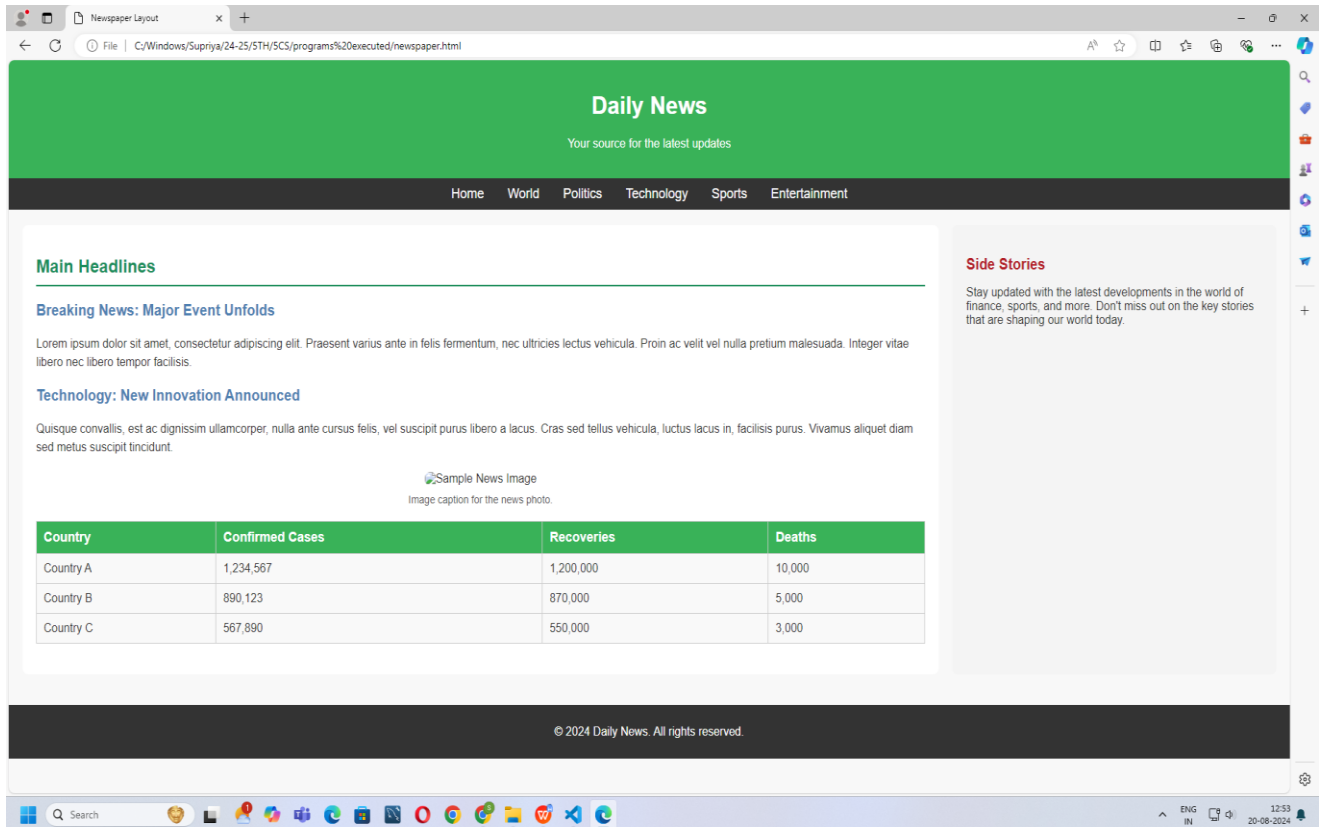
1. **<header>**: Represents the header section of the page, containing the newspaper title and tagline.
2. **<nav>**: Represents the navigation bar with links to different sections of the newspaper.
3. **<main>**: Contains the main content area, including articles and sidebars.
4. **<article>**: Represents a self-contained composition in the main content area.
5. **<section>**: Represents thematic grouping of content within the article.
6. **<figure>**: Contains an image related to the article content with a caption (**<figcaption>**).
7. **<table>**: Used to display top news stories in a tabular format.
8. **<aside>**: Represents content related to the main content, such as a sidebar with additional information.
9. **<footer>**: Represents the footer section of the page, containing copyright information and contact details.

CSS Styling:

1. **Background Colors**: Different background colors are used for various sections (header, nav, main, aside, footer) to differentiate them visually.
2. **Text Colors and Sizes**: Text color and font size are set for various elements to ensure readability and visual hierarchy.
3. **Element Styling**: Specific styles are applied to elements like **<figure>**, **<table>**, and **<aside>** to enhance their appearance.

4. **Flexbox Layout:** The main element uses Flexbox to create a two-column layout with the article and sidebar.

Output:




```
.calculator input[type="number"] {  
  width: 90%;  
  padding: 10px;  
  margin-bottom: 10px;  
  border: none;  
  border-radius: 5px;  
  font-size: 18px;  
}
```

```
.calculator input[type="button"] {  
  width: 45%;  
  padding: 10px;  
  margin: 5px;  
  border: none;  
  border-radius: 5px;  
  font-size: 18px;  
  cursor: pointer;  
  background-color: #4CAF50;  
  color: white;  
}
```

```
.calculator input[type="button"]:hover {  
  background-color: #45a049;  
}
```

```
.calculator #result {  
  margin-top: 20px;  
  font-size: 24px;  
  color: yellow;  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div class="calculator">
```

```
  <h2>Simple Calculator</h2>
```

```
  <input type="number" id="num1" placeholder="Enter first number">
```

```
  <input type="number" id="num2" placeholder="Enter second number (optional)">
```

```
<div>
```

```
  <input type="button" value="Sum" onclick="calculate('sum')">
```

```
  <input type="button" value="Product" onclick="calculate('product')">
```

```
  <input type="button" value="Difference" onclick="calculate('difference')">
```

```
<input type="button" value="Remainder" onclick="calculate('remainder')">
<input type="button" value="Quotient" onclick="calculate('quotient')">
<input type="button" value="Power" onclick="calculate('power')">
<input type="button" value="Square Root" onclick="calculate('sqrt')">
<input type="button" value="Square" onclick="calculate('square')">
</div>
```

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

```
<script>
function calculate(operation) {
    const num1 = parseFloat(document.getElementById('num1').value);
    const num2 = parseFloat(document.getElementById('num2').value);
    let result = "";

    switch(operation) {
        case 'sum':
            result = num1 + num2;
            break;
        case 'product':
            result = num1 * num2;
            break;
        case 'difference':
            result = num1 - num2;
            break;
        case 'remainder':
            result = num1 % num2;
            break;
        case 'quotient':
            result = num1 / num2;
            break;
        case 'power':
            result = Math.pow(num1, num2);
            break;
        case 'sqrt':
            result = Math.sqrt(num1);
            break;
        case 'square':
            result = num1 * num1;
            break;
        default:
            result = 'Invalid operation';
    }
}
```



```
}  
  
document.getElementById('result').innerText = `Result: ${result}`;  
}  
</script>  
</body>  
</html>
```

Explanation of the Code

HTML Structure:

1. The calculator is enclosed within a <div> with the class calculator.
2. Two input fields (num1 and num2) are provided for entering numbers.
3. Buttons are provided for each operation (+, -, *, /, %, ^, √, x²).
4. A result field is provided (result) to display the calculation output.
5. A "Clear" button clears all input and output fields.

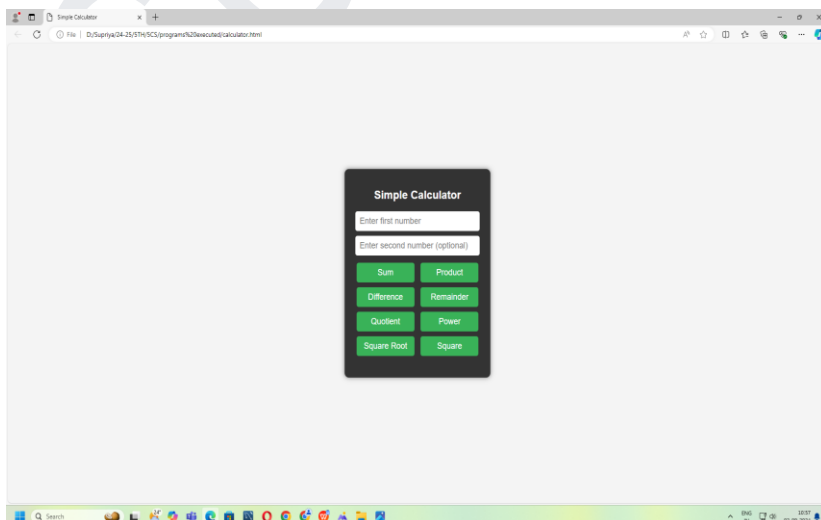
CSS Styling:

1. Basic styling is applied to center the calculator on the page and provide padding, margin, background color, and hover effects.
2. Flexbox is used to align elements and create a responsive layout.

JavaScript Functions:

1. The calculate() function takes an operation parameter to determine which calculation to perform.
2. It reads the values from the input fields, checks if they are valid numbers, and then performs the operation.
3. The result is displayed in the result input field.
4. The clearCalculator() function clears the input fields and result field.

Output:



Experiment-7

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**

Source Code:

Step-by-Step Example

1. **Create an HTML file named json_to_object.html.**

Source Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Convert JSON to JavaScript Object</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 20px;
      background-color: #f0f0f0;
    }
    .container {
      max-width: 600px;
      margin: 0 auto;
      padding: 20px;
      background-color: white;
      border-radius: 5px;
      box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    }
    textarea {
      width: 100%;
      height: 100px;
      padding: 10px;
      margin-bottom: 10px;
      font-size: 16px;
      border-radius: 5px;
```

```
border: 1px solid #ccc;
}
button {
padding: 10px 20px;
background-color: #007bff;
color: white;
border: none;
border-radius: 5px;
cursor: pointer;
font-size: 16px;
}
button:hover {
background-color: #0056b3;
}
pre {
background-color: #f7f7f7;
padding: 10px;
border-radius: 5px;
border: 1px solid #ddd;
font-size: 14px;
overflow-x: auto;
}
</style>
</head>
<body>
<div class="container">
<h2>Convert JSON to JavaScript Object</h2>
<textarea id="jsonInput" placeholder="Enter JSON text here...">{ "name": "John Doe",
"age": 30, "city": "New York" }</textarea>
<button onclick="convertJsonToObject()">Convert to Object</button>
<h3>Output:</h3>
<pre id="jsonOutput"></pre>
</div>
<script>
function convertJsonToObject() {
const jsonInput = document.getElementById('jsonInput').value;
try {
// Parse JSON string to JavaScript object
const jsonObject = JSON.parse(jsonInput);
// Convert the JavaScript object to a formatted JSON string for display
const formattedOutput = JSON.stringify(jsonObject, null, 2);

// Display the formatted JSON object
```

```
        document.getElementById('jsonOutput').textContent = formattedOutput;
    } catch (error) {
        // Display an error message if the JSON string is invalid
        document.getElementById('jsonOutput').textContent = 'Invalid JSON format. Please
check your input.';
    }
}
</script>
</body>
</html>
```

Explanation of the Code

HTML Structure:

1. A textarea element allows users to input JSON text. It comes pre-filled with a sample JSON object.
2. A button is provided to trigger the conversion process.
3. The pre element is used to display the formatted output or any error messages.

CSS Styling:

1. Basic CSS styling is added to enhance the user interface, making it more visually appealing and user-friendly.

JavaScript Functionality:

1. convertJsonToObject(): This function reads the JSON text from the textarea and attempts to parse it into a JavaScript object using JSON.parse().
2. If parsing is successful, it formats the JavaScript object using JSON.stringify() with indentation for better readability and displays the output.
3. If an error occurs during parsing (e.g., due to invalid JSON syntax), an error message is displayed.

Sample input:

```
{
  "name": "Alice",
  "age": 28,
  "city": "Wonderland",
  "isStudent": false
}
```

Output:

Convert JSON to JavaScript Object

```
{ "name": "John Doe", "age": 30, "city": "New York" }
```

Convert to Object

Output:

```
{  
  "name": "John Doe",  
  "age": 30,  
  "city": "New York"  
}
```

B)

Source Code:

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>Convert JSON to Date</title>  
  <style>  
    body {  
      font-family: Arial, sans-serif;  
      background-color: #f4f4f4;  
      padding: 20px;  
    }  
  
    #dateDisplay {  
      margin-top: 20px;  
      background-color: #333;  
      color: white;  
      padding: 10px;  
      border-radius: 5px;  
    }  
  </style>  
</head>  
<body>  
  <div id="dateDisplay">  
    <div id="date"></div>  
  </div>  
</body>  
</html>
```

```
</style>
</head>
<body>

  <h2>Convert JSON Results to Date</h2>
  <textarea id="jsonDataInput" rows="10" cols="50" placeholder="Enter JSON with date like
{"date": "2024-08-22T10:00:00Z"}"></textarea><br><br>
  <button onclick="convertToDate()">Convert to Date</button>

  <div id="dateDisplay"></div>

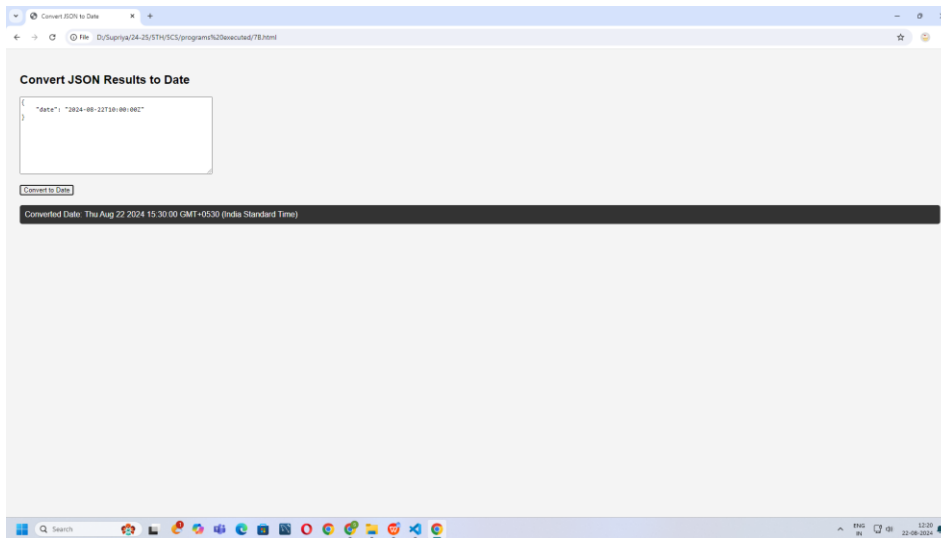
  <script>
    function convertToDate() {
      const jsonText = document.getElementById('jsonDataInput').value;
      try {
        const jsObject = JSON.parse(jsonText);
        const date = new Date(jsObject.date);
        document.getElementById('dateDisplay').innerHTML = `Converted Date:
${date.toString()}`;
      } catch (error) {
        document.getElementById('dateDisplay').innerHTML = "Invalid JSON format!";
      }
    }
  </script>

</body>
</html>
```

Sample input:

```
{
  "date": "2024-08-22T10:00:00Z"
}
```

Output:



C)

Source code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>JSON to CSV and CSV to JSON</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
      padding: 20px;
    }

    #csvJsonDisplay {
      margin-top: 20px;
      background-color: #333;
      color: white;
      padding: 10px;
      border-radius: 5px;
    }
  </style>
</head>
<body>

  <h2>Convert JSON to CSV and CSV to JSON</h2>

  <h3>JSON to CSV</h3>
  <textarea id="jsonToCsvInput" rows="10" cols="50" placeholder="Enter JSON array like
[{"name":"John","age":30},{"name":"Jane","age":25}]"></textarea><br><br>
```

```
<button onclick="convertJsonToCsv()">Convert to CSV</button>
```

```
<h3>CSV to JSON</h3>
```

```
<textarea id="csvToJsonInput" rows="10" cols="50" placeholder="Enter CSV text  
here..."></textarea><br><br>
```

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

```
<div id="csvJsonDisplay"></div>
```

```
<script>
```

```
function convertJsonToCsv() {  
  const jsonText = document.getElementById('jsonToCsvInput').value;  
  try {  
    const jsObject = JSON.parse(jsonText);  
    const headers = Object.keys(jsObject[0]);  
    const csv = [  
      headers.join(','),  
      ...jsObject.map(row => headers.map(header => row[header]).join(','))  
    ].join('\n');  
    document.getElementById('csvJsonDisplay').innerHTML = `  } catch (error) {  
    document.getElementById('csvJsonDisplay').innerHTML = "Invalid JSON format!";  
  }  
}
```

```
function convertCsvToJson() {  
  const csvText = document.getElementById('csvToJsonInput').value;  
  try {  
    const [headers, ...rows] = csvText.split('\n');  
    const headerArray = headers.split(',');  
    const jsonArray = rows.map(row => {  
      const values = row.split(',');  
      return headerArray.reduce((obj, header, index) => {  
        obj[header] = values[index];  
        return obj;  
      }, {});  
    });  
    document.getElementById('csvJsonDisplay').innerHTML =  
    `  } catch (error) {  
    document.getElementById('csvJsonDisplay').innerHTML = "Invalid CSV format!";  
  }  
}
```


</script>

</body>

</html>

Sample input:

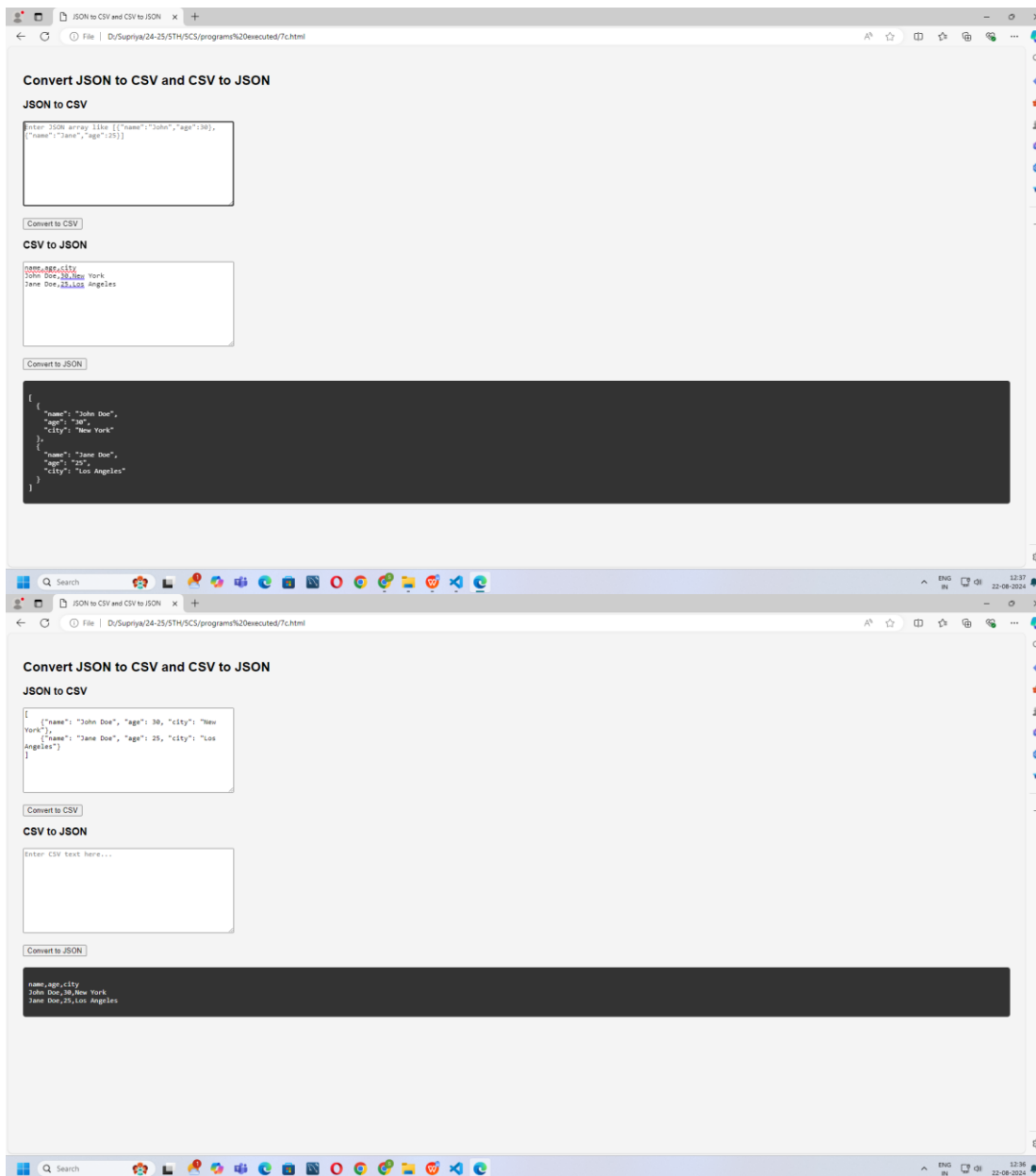
Sample Input for JSON to CSV

```
[  
  {"name": "John Doe", "age": 30, "city": "New York"},  
  {"name": "Jane Doe", "age": 25, "city": "Los Angeles"}  
]
```

Sample Input for CSV to JSON

```
name,age,city  
John Doe,30,New York  
Jane Doe,25,Los Angeles
```

Output:



D)

Step 1: Install Node.js

Ensure that Node.js is installed on your system. You can download and install it from the official Node.js website: <https://nodejs.org/>.

Step 2: Create a JavaScript File

Create a JavaScript file named `hashGenerator.js` with the following code:

Source Code:

```
// Import the crypto module
const crypto = require('crypto');
```

```
// Function to create a hash from a string
function createHashFromString(inputString) {
  // Use the createHash method and specify the algorithm ('sha256', 'sha512', etc.)
  const hash = crypto.createHash('sha256');

  // Update the hash content with the input string
  hash.update(inputString);

  // Calculate the digest in hexadecimal format
  const hashOutput = hash.digest('hex');

  return hashOutput;
}

// Test the function
const inputString = 'Hello, World!';
const hashResult = createHashFromString(inputString);

console.log(`Input String: ${inputString}`);
console.log(`SHA-256 Hash: ${hashResult}`);
```

Sample input:

```
const input = 'Hello, World!';
```

Explanation of the Code

Importing the Crypto Module:

1. The crypto module is imported using `require('crypto')`.

Creating a Hash:

1. The `createHash` function from the crypto module is used to create a new hash object. The algorithm for hashing is specified ('sha256' in this example, but you can use others like 'md5', 'sha512', etc.).

Updating the Hash Content:

1. The `update` method is used to specify the input string for which the hash is to be generated.

Calculating the Digest:

1. The `digest` method computes the hash digest and returns it in hexadecimal format ('hex').

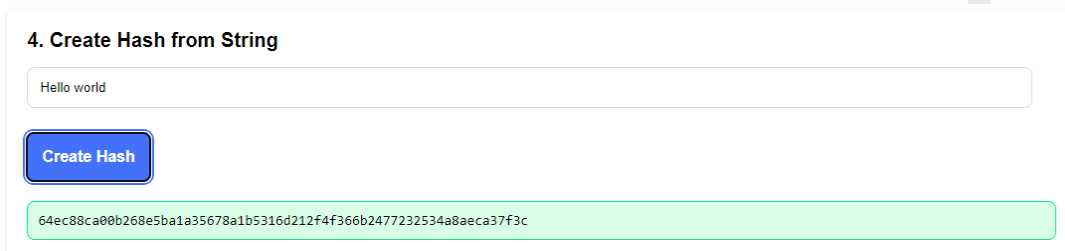
Testing the Function:

1. A sample input string 'Hello, World!' is passed to the function to generate its hash, and the result is printed to the console.

Step 3: Run the JavaScript File

To run the file, open a terminal or command prompt, navigate to the directory where hashGenerator.js is saved, and run the following command:

Output:



4. Create Hash from String

Hello world

Create Hash

64ec88ca00b268e5ba1a35678a1b5316d212f4f366b2477232534a8aeca37f3c

Experiment-8

- 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.

Source Code:

Experiment-9

Develop jQuery script (with HTML/CSS) for:

- Appends the content at the end of the existing paragraph and list.
- Change the state of the element with CSS style using animate() method
- Change the color of any div that is animated.

Source Code:

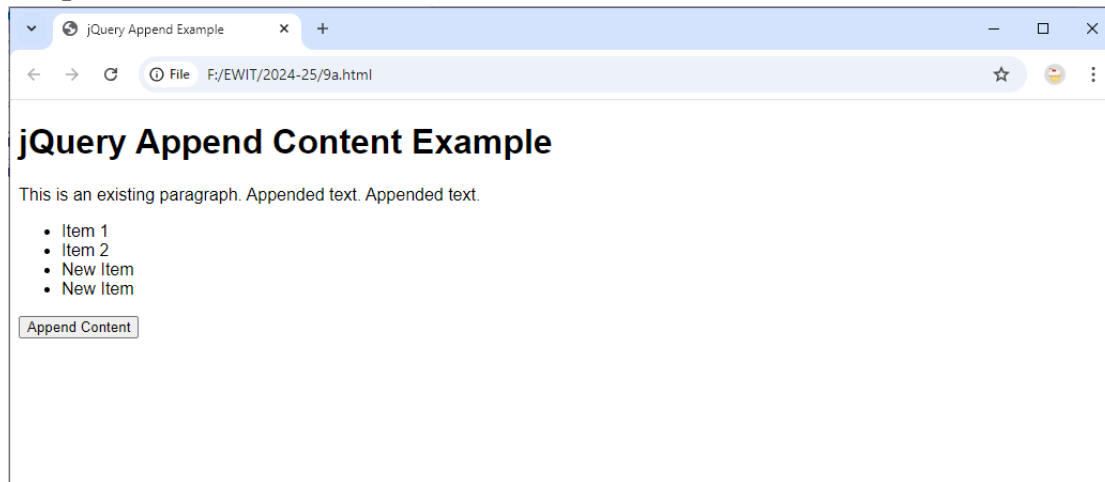
A)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>jQuery Append Example</title>
  <style>
    body {
      font-family: Arial, sans-serif;
    }
    #content {
      margin-top: 20px;
      padding: 10px;
      background-color: #e0f7fa;
      border-radius: 5px;
    }
  </style>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
  <h1>jQuery Append Content Example</h1>
  <p id="paragraph">This is an existing paragraph.</p>
  <ul id="list">
    <li>Item 1</li>
    <li>Item 2</li>
  </ul>
  <button id="appendContent">Append Content</button>

  <script>
    $(document).ready(function() {
      $('#appendContent').click(function() {
        $('#paragraph').append(' Appended text.');
```

```
        $('#list').append('<li>New Item</li>');  
    });  
});  
</script>  
</body>  
</html>
```

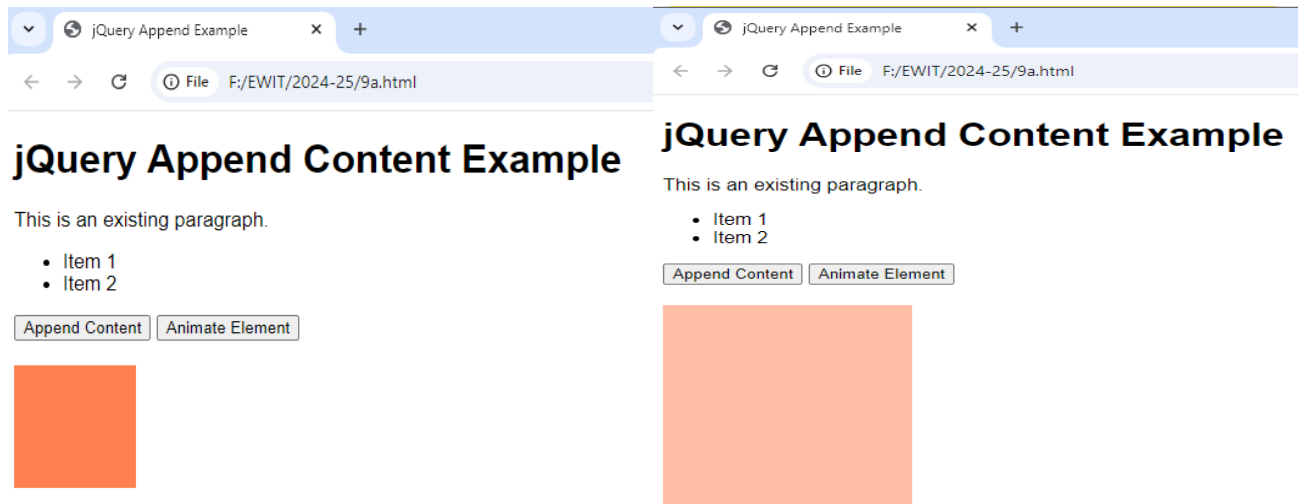
Output:



B)Additional Code for Animation:

```
<button id="animateElement">Animate Element</button>  
<div id="animateDiv" style="width: 100px; height: 100px; background-color: coral; margin-  
top: 20px;"></div>  
  
<script>  
    $('#animateElement').click(function() {  
        $('#animateDiv').animate({  
            width: '200px',  
            height: '200px',  
            opacity: 0.5  
        }, 1000);  
    });  
</script>
```

Output:



C)Additional Code for Changing Color

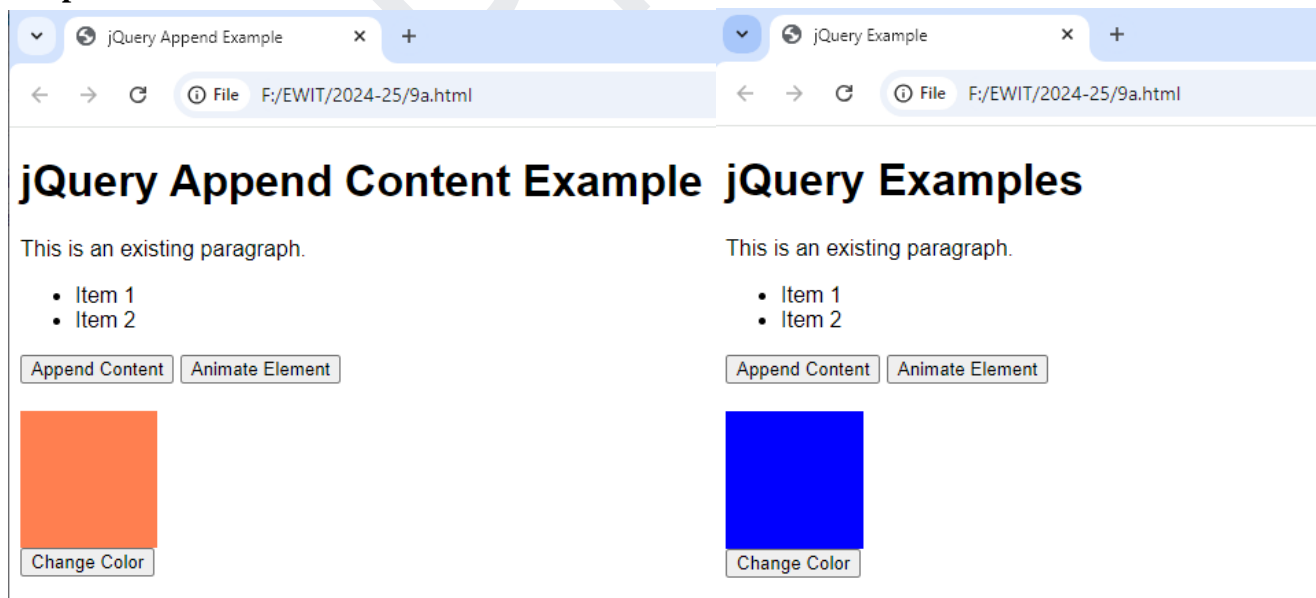
```
<button id="changeColor">Change Color</button>
```

```
<script>
```

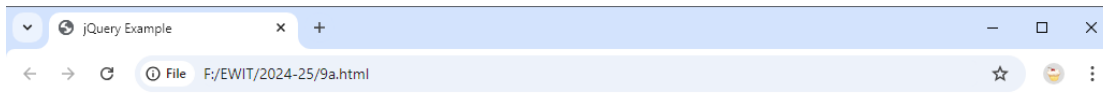
```
$('#changeColor').click(function() {  
    $('#animateDiv').animate({ backgroundColor: 'blue' }, 1000);  
});
```

```
</script>
```

Output:



Final Output:



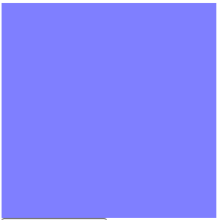
jQuery Examples

This is an existing paragraph. Appended text. Appended text. Appended text.

- Item 1
- Item 2
- New Item
- New Item
- New Item

Append Content

Animate Element



Change Color

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.**

Source Code:

A)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Vanilla JS AJAX Example</title>
  <style>
    body {
      font-family: Arial, sans-serif;
    }
    #content {
      margin-top: 20px;
      padding: 10px;
      background-color: #f4f4f4;
      border-radius: 5px;
    }
  </style>
</head>
<body>
  <h1>Load Content from a File with AJAX (Vanilla JS)</h1>
  <button id="loadContent">Load Content</button>
  <div id="content"></div>

  <script>
    document.getElementById('loadContent').addEventListener('click', function() {
      var xhr = new XMLHttpRequest();
      xhr.open('GET', 'content.txt', true);

      xhr.onload = function() {
        if (this.status == 200) {
          document.getElementById('content').innerHTML = this.responseText;
        }
      }
    });
  </script>
</body>
</html>
```

```
    }  
    };  
  
    xhr.send();  
  });  
</script>  
</body>  
</html>
```

Output:

B)

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>jQuery AJAX Example</title>  
  <style>  
    body {  
      font-family: Arial, sans-serif;  
    }  
    #content {  
      margin-top: 20px;  
      padding: 10px;  
      background-color: #e0f7fa;  
      border-radius: 5px;  
    }  
  </style>  
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>  
</head>  
<body>  
  <h1>Load Content from a File with AJAX (jQuery)</h1>  
  <button id="loadContent">Load Content</button>  
  <div id="content"></div>  
  
  <script>  
    $('#loadContent').on('click', function() {  
      $.ajax({  
        url: 'content.txt',  
        method: 'GET',
```

```
        success: function(data) {
            $('#content').html(data);
        },
        error: function() {
            $('#content').html('Error loading content.');
```

```
        }
    });
});
</script>
</body>
</html>
```

Output:

C)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>jQuery getJSON Example</title>
    <style>
        body {
            font-family: Arial, sans-serif;
        }
        #content {
            margin-top: 20px;
            padding: 10px;
            background-color: #f0f4c3;
            border-radius: 5px;
        }
    </style>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
    <h1>Load JSON Data with getJSON()</h1>
    <button id="loadContent">Load JSON</button>
    <div id="content"></div>

    <script>
        $('#loadContent').on('click', function() {
```

```
$.getJSON('data.json', function(data) {  
    var output = '<h2>' + data.title + '</h2>';  
    output += '<p>' + data.description + '</p>';  
    $('#content').html(output);  
});  
});  
</script>  
</body>  
</html>
```

data.json

```
{  
    "title": "JSON Example",  
    "description": "This is an example of loading JSON data using jQuery's getJSON() method."  
}
```

Output:

D)

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>parseJSON Example</title>  
    <style>  
        body {  
            font-family: Arial, sans-serif;  
        }  
        #content {  
            margin-top: 20px;  
            padding: 10px;  
            background-color: #f3e5f5;  
            border-radius: 5px;  
        }  
    </style>  
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>  
</head>  
<body>  
    <h1>Parsing JSON with parseJSON()</h1>
```

```
<button id="loadContent">Load and Parse JSON</button>
<div id="content"></div>

<script>
  $('#loadContent').on('click', function() {
    var jsonString = '{"name": "John Doe", "age": 30, "city": "New York"}';
    var parsedData = $.parseJSON(jsonString);

    var output = '<h2>Name: ' + parsedData.name + '</h2>';
    output += '<p>Age: ' + parsedData.age + '</p>';
    output += '<p>City: ' + parsedData.city + '</p>';
    $('#content').html(output);
  });
</script>
</body>
</html>
```

Output:

Viva Questions

1. What is the purpose of the `<title>` tag in an HTML document?

The `<title>` tag defines the title of the HTML document, displayed in the browser's title bar or tab. It's also used by search engines and when the page is bookmarked.

2. How does the `<marquee>` tag function, and what are its alternatives?

The `<marquee>` tag creates scrolling text or images. It's deprecated in HTML5 due to accessibility issues. Alternatives include using CSS animations or JavaScript to achieve similar effects.

3. Explain the difference between block-level and inline-level elements in HTML.

Block-level elements (e.g., `<div>`, `<p>`) take up the full width of the container and start on a new line.

Inline-level elements (e.g., ``, `<a>`) take up only as much width as their content and do not start on a new line.

4. What is the significance of the `<blockquote>` tag?

The `<blockquote>` tag is used for long quotations and typically indents the quoted text, distinguishing it from the rest of the content.

5. Can you differentiate between the `` and `` tags in HTML?

`` simply makes text bold with no added meaning.

`` not only makes text bold but also emphasizes its importance, providing semantic meaning for screen readers and search engines.

6. Why is the `<pre>` tag used, and how does it differ from the `<p>` tag?

The `<pre>` tag preserves whitespace, line breaks, and formatting as typed in the HTML document. It differs from the `<p>` tag, which collapses multiple spaces into a single space and doesn't preserve line breaks.

7. What are logical styles in HTML, and how do they enhance the presentation of text?

Logical styles (e.g., ``, ``) apply both stylistic formatting and semantic meaning. They help search engines and assistive technologies interpret the document structure and content more effectively.

8. How do you ensure that an HTML document is well-structured and accessible?

Use proper semantic tags (e.g., `<header>`, `<main>`, `<article>`), include alt attributes for images, use heading levels correctly, ensure proper nesting, and follow best practices for accessibility like using ARIA roles where needed.

9. What is the purpose of using `rowspan` and `colspan` in an HTML table?

`rowspan` merges cells vertically across multiple rows, and `colspan` merges cells horizontally across multiple columns. They are used to improve the table's layout, making it more readable by combining similar data points.

10. Explain the significance of table headers (`<thead>`) and footers (`<tfoot>`) in an HTML table.

`<thead>` defines the table's header section, typically used to label columns. `<tfoot>` defines the footer section, useful for summary rows or additional notes. These sections help organize data and improve table readability.

11. What is the difference between an ID selector and a class selector in CSS?

- An ID selector (`#id`) is unique and used for styling a single element. Each element can only have one ID, and an ID can only be used once per page.
- A class selector (`.class`) can be applied to multiple elements, allowing for reusable styles across different parts of the webpage.

12. How does the element selector work, and when is it most appropriate to use it?

The element selector (e.g., `p`, `h1`) targets all instances of a particular HTML element. It's most appropriate when you want to style every occurrence of a tag globally, like setting a universal font for all paragraphs (`p`).

13. Can you explain the significance of using an external stylesheet instead of inline or internal CSS?

An external stylesheet keeps CSS separate from HTML, promoting a cleaner, more maintainable structure. It allows for reuse across multiple HTML files and makes updating styles across a site easier, rather than embedding CSS in each document.

14. What is a group selector in CSS, and how does it help in reducing redundancy?

A group selector (e.g., `h1, h2, p`) applies the same styles to multiple elements, reducing redundancy in the code. Instead of repeating the same styles for each tag, grouping them makes the code more efficient and concise.

15. Explain how attribute selectors work and provide a practical example of their use.

Attribute selectors target elements based on attributes and their values. For example, `[type="text"]` selects input fields of type text:

EX: `input[type="text"] { background-color: #f0f0f0; }`

This applies styles only to specific input fields with the `type="text"` attribute.

16. How does the `hover` pseudo-class enhance the interactivity of links on a webpage?

The `:hover` pseudo-class enhances the interactivity of links by changing styles (like colour, background, etc.) when a user hovers over them. It's commonly used to indicate clickable elements and improve user experience:

Ex: `a:hover { text-decoration: underline; color: blue; }`

17. What are the advantages of using CSS to style HTML elements compared to inline styling?

CSS allows for separation of concerns (style from content), easier maintainability, and reuse across multiple elements or pages. It avoids inline clutter and provides flexibility through stylesheets, making it easier to update and manage designs globally.

18. How do you decide when to use an ID selector over a class selector?

Use an ID selector when styling a unique element on the page (e.g., a header, a specific section). Use a class selector for reusable styles across multiple elements (e.g., button styles, card designs).

19. What is the difference between `input[type="text"]` and `input[type="email"]`?

`input[type="text"]` accepts any text input, while `input[type="email"]` is designed to validate and accept email addresses.

20. Why is font size important in forms?

Font size ensures the readability of form elements. Proper sizing enhances user experience, making the form easier to fill out.

21. What is the purpose of the `placeholder` attribute in input fields?

The `placeholder` provides a hint inside the input field to guide the user on what type of information is expected (e.g., "Enter first name").

22. What is the purpose of using semantic elements like `<header>`, `<article>`, and `<aside>`?

Semantic elements define the structure and meaning of the content, making it more accessible for search engines and assistive technologies.

23. What is the use of the `<figure>` and `<figcaption>` tags?

The `<figure>` tag is used to encapsulate images, and `<figcaption>` provides a caption, making it easier to describe images or visual elements.

24. What is the purpose of the `<aside>` tag in this layout?

The `<aside>` tag holds content related to the main content, such as advertisements or additional information.

25. What is the purpose of `parseFloat` in this code?

`parseFloat` converts the input value (which is a string) into a floating-point number for mathematical operations.

26. Why is the switch-case structure used in this script?

The switch-case structure is used to handle multiple operations (like sum, product) based on the button the user clicks, making the code easier to manage.

27. What happens if you try to divide by zero in this calculator?

The program checks if the second number is zero before dividing and returns the message "Cannot divide by zero" to avoid mathematical errors.

28. How does the square root function work in this calculator?

It uses JavaScript's `Math.sqrt()` function to compute the square root of the first number.

29. Why are background and font colors applied in CSS rather than directly in HTML?

CSS separates content from design, making it easier to manage styles and ensuring the HTML structure focuses on content.

30. How does the `Math.pow` function work in JavaScript?

`Math.pow(num1, num2)` returns the value of `num1` raised to the power of `num2`.

31. What is the advantage of using `switch-case` instead of multiple `if-else` statements in this calculator?

`switch-case` is more readable and efficient when there are multiple distinct conditions like this, compared to chaining several `if-else` statements.

32. What is the purpose of the `file_exists()` function in PHP?

The `file_exists()` function is used to check if a file exists on the server before attempting to read or write to it. This helps to prevent errors when trying to access non-existent files.

33. How does the `file_get_contents()` function work in PHP?

`file_get_contents()` reads the entire content of a file into a string. For example it is used to retrieve the current visitor count from `counter.txt`.

34. What does the `file_put_contents()` function do?

`file_put_contents()` writes data to a file. It can either create a file if it doesn't exist or overwrite the contents of an existing file. In this program, it updates the visitor count in `counter.txt`.

35. How do you increment a value in PHP?

In PHP, you can increment a numeric value using the `++` operator. For example, `$current_count++` increases the value of `$current_count` by 1.

35. What is the role of the `.txt` file in this project?

The `.txt` file (`counter.txt`) is used to store the current number of visitors. It serves as persistent storage that PHP reads from and writes to, ensuring the visitor count is maintained between sessions.

36. How does PHP handle multiple users visiting the page simultaneously?

PHP handles multiple users by executing code on the server for each request independently. However, in the case of file handling (like updating `counter.txt`), race conditions may occur. Using file locking mechanisms or switching to database storage can prevent such issues.

37. What are the security concerns when writing to a file in PHP?

Some security concerns include:

- **File permission issues:** Incorrect permissions could allow unauthorized users to modify files.
- **Race conditions:** As mentioned, simultaneous access could corrupt data.
- **Injection attacks:** If user input is involved in file operations, it could lead to injection attacks like path traversal, which must be guarded against using proper validation.

38. Explain how you can handle click events in jQuery.

Answer: Click events in jQuery can be handled using the `.click()` method or the `.on('click', function() {})` method. These methods bind a function to the click event of a selected element, allowing for interactive user experiences.

39. What is the role of the `$(document).ready()` function?

Answer: The `$(document).ready()` function ensures that the DOM is fully loaded before executing any jQuery code. It helps prevent errors by ensuring that the elements are available for manipulation when the script runs.

40. What is AJAX, and how does it work?

Answer: AJAX (Asynchronous JavaScript and XML) is a technique used to send and receive data asynchronously without refreshing the entire web page. It works by using the `XMLHttpRequest` object or modern `fetch` API to send requests to the server, which then returns data (often in JSON format) that can be processed and displayed dynamically.

41. Explain the difference between the `XMLHttpRequest` and the `fetch` API.

Answer: `XMLHttpRequest` is an older API for making AJAX requests, while the `fetch` API is a modern alternative that is more powerful and flexible. `fetch` returns a Promise, making it easier to work with asynchronous code. Additionally, `fetch` supports more features, such as easier handling of response types.

42. How do you handle errors in AJAX requests?

Answer: Errors in AJAX requests can be handled using the `error` callback in jQuery's `$.ajax()` method or by using `.catch()` when working with Promises in the `fetch` API. It's essential to check the response status to determine if the request was successful.

43. What is the difference between `getJSON()` and `parseJSON()` in jQuery?

Answer: `getJSON()` is a shorthand method for making a GET request to fetch JSON data from a URL and automatically parses the response. `parseJSON()`, on the other hand, is used to convert a JSON string into a JavaScript object and is typically used when you have JSON data in string format.

44. What is the purpose of the `async` property in AJAX requests?

Answer: The `async` property determines whether the request is asynchronous or synchronous. When set to `true` (default), it allows the browser to continue processing other scripts while waiting for the response. If set to `false`, the browser will wait for the request to complete before continuing.

45. What happens if an AJAX request is made to a URL that does not exist?

Answer: If an AJAX request is made to a non-existent URL, the server will typically respond with a 404 Not Found status. In the error callback or `.catch()` block, you can handle this response and inform the user that the requested resource is not available.

46. What is the difference between the `fadeIn()` and `animate()` methods?

Answer: The `fadeIn()` method specifically fades an element into view by changing its opacity, while the `animate()` method can change any CSS properties, including size, position, and color, providing more flexibility in animations.