# ATME COLLEGE OF ENGINEERING

**13th KM Stone, Bannur Road, Mysore - 570 028**

ATME College of Engineering

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## (ACADEMIC YEAR 2026-27)

# LABORATORY MANUAL

**SUBJECT: WEB TECHNOLOGY LAB**

**SUBJECT CODE: BCSL504**

**SEMESTER: V**

**Outcome Based Education (OBE) and Choice Based Credit System (CBCS)**

**(Effective from the academic year 2023-24)**


| Composed By | Verified By | Approved By |
|---|---|---|
| Mr. NAGAPPA T N | Prof. Bhavya Dechamma K S | Dr. PUTTEGOWDA D |
| Programmer | Prof. Vanishree B S | Professor & Head, CSE |
| | Prof. Yashaswini Urs | |
| | Dr.Drakshayini K B | |
| | Faculty Coordinators | |

# INSTITUTIONAL MISSION AND VISION

## Objectives

- To provide quality education and groom top-notch professionals, entrepreneurs and leaders for different fields of engineering, technology and management.
- To open a Training-R & D-Design-Consultancy cell in each department, gradually introduce doctoral and postdoctoral programs, encourage basic & applied research in areas of social relevance, and develop the institute as a center of excellence.
- To develop academic, professional and financial alliances with the industry as well as the academia at national and transnational levels.
- To develop academic, professional and financial alliances with the industry as well as the academia at national and transnational levels.
- To cultivate strong community relationships and involve the students and the staff in local community service.
- To constantly enhance the value of the educational inputs with the participation of students, faculty, parents and industry.

## Vision

- Development of academically excellent, culturally vibrant, socially responsible and globally competent human resources.

## Mission

- To keep pace with advancements in knowledge and make the students competitive and capable at the global level.
- To create an environment for the students to acquire the right physical, intellectual, emotional and moral foundations and shine as torch bearers of tomorrow's society.
- To strive to attain ever-higher benchmarks of educational excellence.

# Department of Computer Science & Engineering

## Vision of the Department

- To develop highly talented individuals in Computer Science and Engineering to deal with real world challenges in industry, education, research and society.

## Mission of the Department

- To inculcate professional behavior, strong ethical values, innovative research capabilities and leadership abilities in the young minds & to provide a teaching environment that emphasizes depth, originality and critical thinking.
- Motivate students to put their thoughts and ideas adoptable by industry or to pursue higher studies leading to research.

**Program outcomes (POs)**

**Engineering Graduates will be able to:**

- **PO1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems
- **PO2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- **PO3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- **PO4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- **PO5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- **PO6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice

- **PO7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
- **PO8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- **PO9. Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- **PO10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- **PO11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- **PO12. Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

## Program Educational Objectives (PEO'S):

1. Empower students with a strong basis in the mathematical, scientific and engineering fundamentals to solve computational problems and to prepare them for employment, higher learning and R&D.
2. Gain technical knowledge, skills and awareness of current technologies of computer science engineering and to develop an ability to design and provide novel engineering solutions for software/hardware problems through entrepreneurial skills.
3. Exposure to emerging technologies and work in teams on interdisciplinary projects with effective communication skills and leadership qualities.
4. Ability to function ethically and responsibly in a rapidly changing environment by Applying innovative ideas in the latest technology, to become effective professionals in Computer Science to bear a life-long career in related areas.

## Program Specific Outcomes (PSOs)

1. Ability to apply skills in the field of algorithms, database design, web design, cloud computing and data analytics.
2. Apply knowledge in the field of computer networks for building network and internet based applications.

| Web Technology Lab | | Semester | 5 |
|---|---|---|---|
| Course Code | BCSL504 | CIE Marks | 50 |
| Teaching Hours/Week (L:T:P: S) | 0:0:2:0 | SEE Marks | 50 |
| Credits | 01 | Exam Hours | 100 |
| Examination type (SEE) | Practical | | |

**Course objectives:**

- Learn HTML 5 elements and their use.
- Use of CSS for enhanced user interface presentation.
- Gain knowledge of JavaScript, AJAX and jQuery for dynamic presentation.
- Use of PHP to build Web applications.
- Design and develop Websites and Web applications.

**List of Experiments:**

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

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 colours.)
   c) Provide colour options for rows.

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.

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.

5. 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.

6. 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.

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

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.

9. 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.

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.

**Programming Assignment (5 marks):**

Construct a Website (multiple Web pages) containing ‘Resume’ and Bio -data by using relevant HTML elements and appropriate styling for presentation with CSS/jQuery/JavaScript. Host the Website on a cloud platform.

**Programming Assignment (5 marks):**

Build a Web application with HTML, CSS, JavaScript, jQuery and PHP for online application/registration form. Form should accept the information and print/display on a browser with formatting/styling upon submission (Button click) on success. Host the application on a cloud platform.

**Course outcomes (Course Skill Set):**

At the end of the course, the student will be able to:

- Design the experiment for the given problem using HTML, Javascript and CSS.
- Develop the solution for the given real-world problem using jQuery, Ajax and PHP.
- Analyze the results and produce substantial written documentation.

**Assessment Details (both CIE and SEE)**

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together

**Continuous Internal Evaluation (CIE):**

CIE marks for the practical course are **50 Marks.**

The split-up of CIE marks for record/ journal and test are in the ratio **60:40.**

- Each experiment is to be evaluated for conduction with an observation sheet and record write-up. Rubrics for the evaluation of the journal/write-up for hardware/software experiments are designed by the faculty who is handling the laboratory session and are made known to students at the beginning of the practical session.
- Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks.
- Total marks scored by the students are scaled down to **30 marks** (60% of maximum marks).
- Weightage to be given for neatness and submission of record/write-up on time.
- Department shall conduct a test of 100 marks after the completion of all the experiments listed in the syllabus.
- In a test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce.
- The suitable rubrics can be designed to evaluate each student's performance and learning ability.
- The marks scored shall be scaled down to **20 marks** (40% of the maximum marks).

The Sum of scaled-down marks scored in the report write-up/journal and marks of a test is the total CIE marks scored by the student.

**Semester End Evaluation (SEE):**

- SEE marks for the practical course are 50 Marks.
- SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the Head of the Institute.
- The examination schedule and names of examiners are informed to the university before the conduction of the examination. These practical examinations are to be conducted between the schedule mentioned in the academic calendar of the University.
- All laboratory experiments are to be included for practical examination.
- (Rubrics) Breakup of marks and the instructions printed on the cover page of the answer script to be strictly adhered to by the examiners. **OR** based on the course requirement evaluation rubrics shall be decided jointly by examiners.
- Students can pick one question (experiment) from the questions lot prepared by the examiners jointly.
- Evaluation of test write-up/ conduction procedure and result/viva will be conducted jointly by examiners.

General rubrics suggested for SEE are mentioned here, writeup-20%, Conduction procedure and result in -60%, Viva-voce 20% of maximum marks. SEE for practical shall be evaluated for 100 marks and scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall be decided by the examiners)

Change of experiment is allowed only once and 15% of Marks allotted to the procedure part are to be made zero.

The minimum duration of SEE is 02 hours

**Suggested Learning Resources:**

**Books:**


1. Randy Connolly and Ricardo Hoar, Fundamentals of Web Development, 3rd edition, Pearson, 2021
2. Robert W Sebesta, Programming the World Wide Web, 8th Edition, Pearson Education, 2020.


**Web Links:**

- https://www.w3schools.com/html/default.asp
- https://www.w3schools.com/css/default.asp
- https://www.w3schools.com/js/js_examples.asp
- https://www.geeksforgeeks.org/javascript-examples/
- https://www.w3schools.com/php/default.asp
- https://www.w3schools.com/jquery/default.asp
- https://www.w3schools.com/js/js_ajax_intro.asp
- https://www.geeksforgeeks.org/jquery-tutorial/

# CONTENTS


| SL.NO. | EXPERIMENT NAME | PAGE NO. |
|---|---|---|
| 1 | **INTRODUCTION** | 1 |
| 2 | **Program 1**<br>Develop the HTML page named as “Myfirstwebpage.html”. Add the following tags with relevant content.<br>1. Set the title of the page as “My First Web Page”<br>2. Within the body use the following tags:<br>a) Moving text = “Basic HTML Tags”<br>b) Different heading tags (h1 to h6)<br>c) Paragraph<br>d) Horizontal line<br>e) Line Break<br>f) Block Quote<br>g) Pre tag<br>Different Logical Style (b>, <u>, <sub>, <sup> etc.) | 30 |
| 3 | **Program 2**<br>Develop the HTML page named as “Table.html” to display your class time table.<br>a) Provide the title as Time Table with table header and table footer, row-span and col-span etc.<br>b) Provide various colour options to the cells (Highlight the lab hours and elective hours with different colours.)<br>**c)** Provide colour options for rows. | 33 |
| 4 | **Program 3**<br>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. | 37 |
| 5 | **Program 4**<br>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. | 41 |
| 6 | **Program 5**<br>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. | 45 |
| 7 | **Program 6**<br>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. | 50 |

| | | |
|---|---|---|
| 8 | **Program 7**<br>Develop JavaScript program (with HTML/CSS) for:<br>a) Converting JSON text to JavaScript Object<br>b) Convert JSON results into a date<br>c) Converting From JSON To CSV and CSV to JSON<br>d) Create hash from string using crypto.createHash() method | 55 |
| 9 | **Program 8**<br>**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.<br>**b.** Develop a PHP program (with HTML/CSS) to sort the student records which are stored in the database using selection sort. | 60 |
| 10 | **Program 9**<br>Develop jQuery script (with HTML/CSS) for:<br>a) Appends the content at the end of the existing paragraph and list.<br>b) Change the state of the element with CSS style using animate() method<br>c) Change the color of any div that is animated. | 67 |
| 11 | **Program 10**<br>Develop a JavaScript program with Ajax (with HTML/CSS) for:<br>a) Use ajax() method (without Jquery) to add the text content from the text file by sending ajax request.<br>b) Use ajax() method (with Jquery) to add the text content from the text file by sending ajax request.<br>c) Illustrate the use of getJSON() method in jQuery<br>Illustrate the use of parseJSON() method to display JSON values. | 70 |
| 12 | **VIVA QUESTIONS AND ANSWERS** | 75 |

# Introduction to HTML

### 1. What is HTML?

HTML stands for HyperText Markup Language. It is the standard language used to create and design web pages. HTML structures content on the web, allowing browsers to interpret and display text, images, and other resources.

### 2. Basic HTML Document Structure

An HTML document has a specific structure. Here's a simple example:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Page Title</title>
</head>
<body>
    <h1>Welcome to My Website</h1>
    <p>This is a paragraph.</p>
</body>
</html>
```

- **`<!DOCTYPE html>`**: Declares the document type and HTML version (HTML5 in this case).
- **`<html>`**: Root element that contains all other HTML elements.
- **`<head>`**: Contains meta-information about the document, such as the title and character set.
- **`<body>`**: Contains the content of the web page, such as headings, paragraphs, images, and links.

### 3. HTML Elements and Tags

HTML is made up of elements, which are defined by tags. Tags are enclosed in angle brackets, e.g., `<tagname>`. Elements can have attributes to provide additional information. For example:

- **`<a href="https://example.com">`**: Creates a hyperlink.
- **`<img src="image.jpg" alt="Description">`**: Embeds an image.

# Common HTML Elements

### 1. Headings and Paragraphs

- **Headings**: `<h1>` to `<h6>` elements define headings, where `<h1>` is the largest and `<h6>` is the smallest.

  ```
  <h1>Main Heading</h1>
  <h2>Subheading</h2>
  ```

- **Paragraphs**: `<p>` element is used for paragraphs of text.

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

### 2. Links and Anchors

- **Anchor Tags**: `<a>` element is used to create hyperlinks.

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

  - **`href`** attribute specifies the URL of the link.
  - **`target="_blank"`** attribute can be used to open the link in a new tab.

### 3. Images

- **Image Tags**: `<img>` element is used to embed images.

```
<img src="image.jpg" alt="Description">
```

  - **`src`** attribute specifies the image source.
  - **`alt`** attribute provides alternative text if the image cannot be displayed.

## Lists and Tables

### 1. Lists

- **Ordered Lists**: `<ol>` element creates a numbered list.

```
<ol>
    <li>First item</li>
    <li>Second item</li>
</ol>
```

- **Unordered Lists**: `<ul>` element creates a bulleted list.

```
<ul>
    <li>Item one</li>
    <li>Item two</li>
</ul>
```

### 2. Tables

- **Table Structure**: Tables are created using `<table>`, with rows defined by `<tr>`, headers by `<th>`, and cells by `<td>`.

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

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

## Attributes and Forms

### 1. Attributes

Attributes provide additional information about elements. They are placed in the opening tag of an element and usually come in name-value pairs.

- **Example**: The `href` attribute of an `<a>` tag specifies the URL.

```
<a href="https://example.com">Link</a>
```

- **Common Attributes**: `id`, `class`, `style`, `title`.

### 2. Forms

Forms allow users to submit data to a server. Common form elements include:

- **Form Tag**: `<form>` is used to create a form.

```
<form action="/submit" method="post">
    <input type="text" name="username" placeholder="Enter your name">
    <input type="submit" value="Submit">
</form>
```

- **Input Fields**: `<input>` element allows various types of user input.

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

- **Text Areas**: `<textarea>` is used for multi-line text input.

```
<textarea name="comments" rows="4" cols="50">Enter your comments
here</textarea>
```

## HTML5 Features and Best Practices

### 1. HTML5 Elements

HTML5 introduces new semantic elements that make it easier to structure content:

- **Semantic Elements**:
  - **`<header>`**: Defines a header for a document or section.
  - **`<footer>`**: Defines a footer for a document or section.
  - **`<article>`**: Represents a self-contained piece of content.
  - **`<section>`**: Defines sections within a document.

```
<header>
    <h1>Page Header</h1>
</header>
```

```
<section>
    <article>
        <h2>Article Title</h2>
        <p>Article content goes here.</p>
    </article>
</section>
<footer>
    <p>Footer content</p>
</footer>
```

**2. Multimedia Elements**

HTML5 supports native multimedia elements:

- **Audio**: `<audio>` element for embedding audio content.

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

- **Video**: `<video>` element for embedding video content.

```
<video width="320" height="240" controls>
    <source src="movie.mp4" type="video/mp4">
    Your browser does not support the video tag.
</video>
```

**3. Best Practices**

- **Use Semantic HTML**: Helps with SEO and accessibility.
- **Validate Your HTML**: Use validators to ensure your code is correct.
- **Keep Your Code Clean and Organized**: Use indentation and comments for readability.

# Introduction to CSS

## 1. What is CSS?

CSS stands for Cascading Style Sheets. It is used to style and layout web pages. CSS allows you to separate content (HTML) from design (styles), making it easier to maintain and update the appearance of a website.

## 2. Basic CSS Syntax

CSS is made up of rules that consist of selectors and declarations.

- **Selector**: Targets the HTML element you want to style.
- **Declaration Block**: Contains one or more declarations enclosed in curly braces `{}`. Each declaration is a property-value pair.

**Example:**

```
selector {
    property: value;
}
```

**Example CSS Code:**

```
body {
    background-color: lightblue;
}

h1 {
    color: navy;
    text-align: center;
}
```

## 3. Including CSS in HTML

CSS can be included in three ways:

- **Inline CSS**: Applied directly within an HTML element's `style` attribute.

  ```
  <p style="color: blue;">This is a blue paragraph.</p>
  ```

- **Internal CSS**: Defined within the `<style>` tag inside the `<head>` section of the HTML document.

  ```
  <style>
      p {
          color: red;
      }
  </style>
  ```

- **External CSS**: Linked to an HTML document via the `<link>` tag. This is the most recommended approach for larger sites.

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

## Selectors and Properties

### 1. Selectors

Selectors are used to target HTML elements you want to style. Here are some common types:

- **Element Selector**: Targets all instances of a specific HTML element.

```
p {
    color: green;
}
```

- **Class Selector**: Targets elements with a specific `class` attribute. Prefixed with a dot `(.)`.

```
.class-name {
    font-size: 18px;
}
```

- **ID Selector**: Targets an element with a specific `id` attribute. Prefixed with a hash `#`.

```
#unique-id {
    background-color: yellow;
}
```

- **Attribute Selector**: Targets elements based on an attribute or its value.

```
input[type="text"] {
    border: 1px solid black;
}
```

- **Universal Selector**: Targets all elements on the page.

```
* {
    margin: 0;
    padding: 0;
}
```

### 2. Combining Selectors

You can combine selectors to target more specific elements.

- **Descendant Selector**: Targets elements nested within a specified element.

```
div p {
    color: purple;
}
```

- **Child Selector**: Targets direct children of a specified element.

```
ul > li {
    list-style-type: square;
}
```

- **Group Selector**: Applies the same styles to multiple selectors.

```
h1, h2, h3 {
    font-family: Arial, sans-serif;
}
```

## CSS Properties

### 1. Text Properties

- **`color`**: Sets the color of text.
```
p {
    color: #333;
}
```

- **`font-family`**: Specifies the font to be used.
```
body {
    font-family: 'Arial', sans-serif;
}
```

- **`font-size`**: Sets the size of the font.
```
h1 {
    font-size: 24px;
}
```

- **`text-align`**: Aligns text within an element.
```
p {
    text-align: center;
}
```

### 2. Box Model Properties

- **`margin`**: Sets the space outside an element's border.
```
div {
    margin: 20px;
}
```

- **`padding`**: Sets the space inside an element's border.
```
div {
    padding: 10px;
}
```
- **`border`**: Sets the border around an element.
```
div {
    border: 2px solid black;
}
```
- **`width`** and **`height`**: Set the dimensions of an element.

```
img {
    width: 100px;
    height: auto;
}
```

### 3. Background Properties

- **`background-color`**: Sets the background color of an element.

```
body {
    background-color: lightgray;
}
```

- **`background-image`**: Sets an image as the background.

```
div {
    background-image: url('background.jpg');
}
```

- **`background-size`**: Specifies the size of the background image.

```
div {
    background-size: cover;
}
```

## Advanced CSS Techniques

### 1. Responsive Design

Responsive design ensures that your website looks good on all devices and screen sizes.

- **Media Queries**: Apply different styles for different devices.

```
@media (max-width: 600px) {
    body {
        font-size: 14px;
    }
}
```

- **Flexible Grid Layouts**: Use percentages and flexible units like `vw` (viewport width) and `vh` (viewport height).

```
.container {
    width: 80%;
    margin: 0 auto;
}
```

### 2. Flexbox

Flexbox is a layout model that allows for more efficient and flexible layouts.

- **Basic Flexbox Example**:

```
.container {
    display: flex;
    justify-content: space-between;
}
```

- **Flex Properties**:
  - **`display: flex;`**: Defines a flex container.
  - **`flex-direction`**: Defines the direction of flex items.

```
.container {
    flex-direction: row; /* or column */
}
```

- **`justify-content`**: Aligns items along the main axis.

```
.container {
    justify-content: center;
}
```

### 3. Grid Layout

CSS Grid Layout provides a two-dimensional grid-based layout system.

- **Basic Grid Example**:

```
.grid-container {
    display: grid;
    grid-template-columns: 1fr 2fr;
    gap: 10px;
}

.grid-item {
    background-color: lightblue;
    padding: 20px;
}
```

- **Grid Properties**:
  - **`grid-template-columns`**: Defines the columns of the grid.
  - **`grid-template-rows`**: Defines the rows of the grid.

## Best Practices and Tools

### 1. Best Practices

- **Keep CSS Organized**: Use comments and maintain a consistent structure.

```
/* Layout Styles */
.container { ... }

/* Typography Styles */
h1 { ... }
```

- **Minimize Use of Inline Styles**: Use external or internal CSS for better maintainability.
- **Optimize Performance**: Minify CSS files and combine multiple stylesheets where possible.
- **Use Browser Developer Tools**: Inspect and debug styles in real-time.

### 2. CSS Preprocessors

- **Sass**: A CSS preprocessor that adds features like variables, nesting, and mixins.

```
$primary-color: #333;

body {
    color: $primary-color;
}
```

- **Less**: Similar to Sass, with its own set of features.

```
@primary-color: #333;

body {
    color: @primary-color;
}
```

### 3. CSS Frameworks

- **Bootstrap**: A popular framework that provides pre-designed components and a grid system.

```
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
```

- **Tailwind CSS**: A utility-first CSS framework that allows for highly customizable styling.

```
<link
href="https://cdn.jsdelivr.net/npm/tailwindcss@2.0.0/dist/tailwind.min.css" rel="stylesheet">
```

### 4. Resources

- **MDN Web Docs**: Comprehensive resource for CSS documentation and examples.
- **CSS-Tricks**: Tutorials and tips for CSS development.
- **Can I use**: Check browser support for CSS features

## Introduction to JavaScript

### 1. What is JavaScript?

JavaScript is a versatile, high-level programming language that is widely used for creating interactive and dynamic web content. It is an essential part of web development, along with HTML and CSS.

### 2. Adding JavaScript to HTML

JavaScript can be included in HTML documents in several ways:

- **Inline JavaScript**: Directly within an HTML element using the `onclick` attribute.

```
<button onclick="alert('Hello World!')">Click me</button>
```

- **Internal JavaScript**: Inside the `<script>` tag within the HTML `<head>` or `<body>` section.

```
<script>
    console.log('Hello World!');
</script>
```

- **External JavaScript**: In a separate `.js` file linked via the `<script>` tag.

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

### 3. Basic Syntax

- **Comments**: Used to annotate the code and are ignored by the interpreter.
  - Single-line comment: `// This is a comment`
  - Multi-line comment: `/* This is a multi-line comment */`
- **Variables**: Declared using `var`, `let`, or `const`.

```
let name = 'John'; // `let` allows reassignment
const age = 30;    // `const` does not allow reassignment
```

- **Data Types**:
  - **String**: Text (`"Hello"`, `'World'`)
  - **Number**: Numeric values (`42`, `3.14`)
  - **Boolean**: True or false (`true`, `false`)
  - **Object**: Collection of key-value pairs
  - **Array**: Ordered list of values (`[1, 2, 3]`)

## Operators and Control Structures

### 1. Operators

- **Arithmetic Operators**: Perform mathematical operations.

```
let sum = 5 + 3;      // Addition
```

```
let difference = 10 - 4; // Subtraction
let product = 4 * 3;  // Multiplication
let quotient = 12 / 3; // Division
```

- **Comparison Operators**: Compare values and return a boolean.

```
let isEqual = (5 === 5);  // Strict equality
let isGreater = (10 > 5); // Greater than
```

- **Logical Operators**: Perform logical operations.

```
let andResult = (true && false); // AND
let orResult = (true || false);  // OR
let notResult = !true;           // NOT
```

**2. Control Structures**

- **Conditional Statements**: Execute code based on conditions.

```
if (age >= 18) {
    console.log('Adult');
} else {
    console.log('Minor');
}
```

- **Switch Statement**: Alternative to multiple `if` statements.

```
switch (day) {
    case 1:
        console.log('Monday');
        break;
    case 2:
        console.log('Tuesday');
        break;
    default:
        console.log('Weekend');
}
```

- **Loops**: Repeats code multiple times.

```
// For loop
for (let i = 0; i < 5; i++) {
    console.log(i);
}

// While loop
let count = 0;
while (count < 5) {
    console.log(count);
    count++;
}

// Do-While loop
let num = 0;
do {
    console.log(num);
    num++;
} while (num < 5);
```

## Functions and Scope

### 1. Functions

Functions are reusable blocks of code that perform specific tasks.

- **Function Declaration**:

```
function greet(name) {
    return 'Hello, ' + name;
}

console.log(greet('Alice')); // Output: Hello, Alice
```

- **Function Expression**: Assigned to a variable.

```
const square = function(x) {
    return x * x;
}

console.log(square(4)); // Output: 16
```

- **Arrow Functions**: Shorter syntax for writing functions.

```
const add = (a, b) => a + b;
console.log(add(2, 3)); // Output: 5
```

### 2. Scope

Scope determines the visibility of variables.

- **Global Scope**: Variables declared outside any function.

```
let globalVar = 'Global';
function display() {
    console.log(globalVar); // Accessible here
}
```

- **Local Scope**: Variables declared inside a function.

```
function example() {
    let localVar = 'Local';
    console.log(localVar); // Accessible here
}
console.log(localVar); // Error: localVar is not defined
```

- **Block Scope**: Variables declared with `let` or `const` within a block (e.g., `if` or `for`).

```
if (true) {
    let blockVar = 'Block';
    console.log(blockVar); // Accessible here
}
console.log(blockVar); // Error: blockVar is not defined
```

## Objects and Arrays

### 1. Objects

Objects are collections of key-value pairs.

- **Creating an Object**:

```
let person = {
    name: 'John',
    age: 30,
    greet: function() {
        return 'Hello, ' + this.name;
    }
};

console.log(person.name); // Output: John
console.log(person.greet()); // Output: Hello, John
```

- **Accessing and Modifying Object Properties**:

```
person.age = 31; // Modify property
console.log(person['name']);  //  Access  property  using  bracket
notation
```

### 2. Arrays

Arrays are ordered lists of values.

- **Creating an Array**:

```
let numbers = [1, 2, 3, 4, 5];
```

- **Accessing Array Elements**:

```
console.log(numbers[0]); // Output: 1
```

- **Array Methods**:

```
numbers.push(6); // Add element to the end
numbers.pop();   // Remove element from the end
numbers.shift(); // Remove element from the beginning
numbers.unshift(0); // Add element to the beginning

numbers.forEach(num => console.log(num)); // Iterate over array
elements
```

## Advanced Concepts and Best Practices

### 1. DOM Manipulation

JavaScript interacts with the Document Object Model (DOM) to modify web page content and structure.

- **Selecting Elements**:

```
let element = document.getElementById('myElement');
let elements = document.getElementsByClassName('myClass');
```

- **Modifying Content**:

```
element.textContent = 'New Content';
```

- **Event Handling**:

```
element.addEventListener('click', () => {
    alert('Element clicked!');
});
```

### 2. Asynchronous JavaScript

Handle operations that take time (e.g., network requests) without blocking the main thread.

- **Callbacks**:

```
function fetchData(callback) {
    setTimeout(() => {
        callback('Data');
    }, 1000);
}

fetchData(data => console.log(data)); // Output after 1 second: Data
```

- **Promises**: Represent a value that may be available now, later, or never.

```
let promise = new Promise((resolve, reject) => {
    setTimeout(() => resolve('Data'), 1000);
});

promise.then(data => console.log(data)); // Output after 1 second:
Data
```

- **Async/Await**: Syntactic sugar over promises, making asynchronous code look synchronous.

```
async function fetchData() {
    let data = await promise;
    console.log(data); // Output after 1 second: Data
}

fetchData();
```

### 3. Error Handling

- **Try/Catch**: Handle errors gracefully.

```
try {
    let result = riskyOperation();
} catch (error) {
    console.log('An error occurred:', error);
}
```

### 4. Best Practices

- **Write Clean and Readable Code**: Use meaningful variable names and maintain consistent formatting.
- **Follow the DRY Principle**: Don't Repeat Yourself. Use functions and objects to avoid code duplication.
- **Use Strict Mode**: Enforce stricter parsing and error handling.

```
'use strict';
```

- **Test and Debug**: Regularly test and debug code to ensure functionality and performance.

## AJAX (Asynchronous JavaScript and XML)

AJAX is a technique used in web development to create interactive and dynamic web applications. It allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This enables a web page to update without requiring a full page reload, enhancing the user experience.

### 1. Understanding AJAX

**1.1 What is AJAX?** AJAX stands for Asynchronous JavaScript and XML. It's not a programming language but a set of web development techniques combining JavaScript and XML (or JSON). The key idea is to allow web applications to send and receive data from a server asynchronously without disrupting the current page.

### 1.2 How AJAX Works:

1. **Client-Side Request**: A client-side script (JavaScript) makes an HTTP request to the server.
2. **Server-Side Processing**: The server processes the request and sends back a response.
3. **Client-Side Update**: JavaScript processes the server's response and updates the web page dynamically.

### 1.3 Components of AJAX:

- **XMLHttpRequest Object**: The core JavaScript object used to send requests and receive responses.
- **JavaScript**: Used to handle the request and update the page.
- **Server-Side Script**: Typically written in languages like PHP, Python, or Node.js, responsible for processing requests.

### 2. Implementing AJAX

### 2.1 Using XMLHttpRequest

Here's a basic example of how to use `XMLHttpRequest` to fetch data from a server.

**Example Code:**

```
// Create a new XMLHttpRequest object
let xhr = new XMLHttpRequest();

// Configure it: GET-request for the URL /article/.../load
xhr.open('GET', 'https://api.example.com/data', true);

// Send the request over the network
xhr.send();

// This function is called when the request completes
xhr.onload = function() {
    if (xhr.status >= 200 && xhr.status < 300) {
        // Request was successful
        let response = JSON.parse(xhr.responseText);
```

```
        console.log(response);
        // Update the page with the response
        document.getElementById('output').textContent = response.message;
    } else {
        // Request failed
        console.error('Request failed with status:', xhr.status);
    }
};
```

### 2.2 Using Fetch API

The `fetch` API is a more modern alternative to `XMLHttpRequest`, providing a simpler and more flexible approach for handling network requests.

**Example Code:**

```
// Make a GET request to the server
fetch('https://api.example.com/data')
    .then(response => {
        if (!response.ok) {
            throw new Error('Network response was not ok');
        }
        return response.json();
    })
    .then(data => {
        console.log(data);
        // Update the page with the response
        document.getElementById('output').textContent = data.message;
    })
    .catch(error => {
        console.error('There  was  a  problem  with  the  fetch  operation:',
error);
    });
```

### 2.3 Handling Different Data Formats

While XML was traditionally used with AJAX, JSON is now the more common format for data interchange.

- **XML Example:**

```
xhr.responseType = 'document'; // Set response type to XML
xhr.onload = function() {
    let xmlDoc = xhr.responseXML;
    let                          element                          =
xmlDoc.getElementsByTagName('message')[0].textContent;
    console.log(element);
    document.getElementById('output').textContent = element;
};
```

- **JSON Example:**

```
xhr.responseType = 'json'; // Set response type to JSON
xhr.onload = function() {
    let jsonResponse = xhr.response;
    console.log(jsonResponse);
```

```
    document.getElementById('output').textContent =
jsonResponse.message;
};
```

### 2.4 Error Handling

Always include error handling to manage network issues or server errors:

- **XMLHttpRequest Error Handling:**

```
xhr.onerror = function() {
    console.error('Request failed');
};
```

- **Fetch API Error Handling:**

```
fetch('https://api.example.com/data')
    .then(response => {
        if (!response.ok) {
            throw new Error('Network response was not ok');
        }
        return response.json();
    })
    .catch(error => {
        console.error('There was a problem with the fetch
operation:', error);
    });
```

## 3. Practical Applications of AJAX

**3.1 Dynamic Content Loading** AJAX is commonly used to dynamically load content without refreshing the entire page, such as in infinite scrolling or live search features.

**3.2 Form Submission** Submit forms asynchronously to update parts of a web page or provide real-time feedback without refreshing.

**3.3 Real-Time Updates** Use AJAX for real-time applications like chat applications, live notifications, and stock price updates.

**3.4 Enhancing User Experience** AJAX can be used to improve the user experience by providing faster interactions and reducing wait times for content updates.

## 4. Summary and Best Practices

### 4.1 Summary

- AJAX allows web pages to be updated asynchronously.
- Use `XMLHttpRequest` or `fetch` API to make asynchronous requests.
- Handle responses in JSON or XML format and ensure proper error handling.

### 4.2 Best Practices

- Minimize the number of requests to reduce server load.
- Use asynchronous operations to avoid blocking the user interface.
- Handle errors gracefully and provide user feedback.
- Consider security implications, such as Cross-Site Request Forgery (CSRF) and Cross-Origin Resource Sharing (CORS).

AJAX is a powerful technique that, when used effectively, can greatly enhance the responsiveness and interactivity of web applications.

## jQuery

**jQuery** is a fast, lightweight, and feature-rich JavaScript library designed to simplify HTML document traversal, event handling, and animation. It provides a more straightforward syntax for common JavaScript tasks and makes it easier to work with the DOM (Document Object Model), events, and AJAX.

### 1. Introduction to jQuery

**1.1 What is jQuery?** jQuery is a cross-browser JavaScript library that simplifies HTML document manipulation, event handling, and animation. It allows developers to write less code to accomplish common tasks, improving productivity and code readability.

### 1.2 Why Use jQuery?

- **Simplified Syntax**: Provides a more concise syntax for common JavaScript tasks.
- **Cross-Browser Compatibility**: Handles inconsistencies across different browsers.
- **Plugins**: Offers a vast number of plugins to extend its functionality.

**1.3 Including jQuery in Your Project** You can include jQuery in your project either by downloading it or using a Content Delivery Network (CDN).

- **Using a CDN**:

```
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
```

- **Downloading jQuery**: Download from the official website and include it in your project directory.

```
<script src="path/to/jquery.min.js"></script>
```

### 2. Basic jQuery Operations

### 2.1 Selecting Elements

jQuery uses CSS-style selectors to target HTML elements.

- **Selecting by ID**:

```
$('#myElement'); // Selects the element with id="myElement"
```

- **Selecting by Class**:

```
$('.myClass'); // Selects all elements with class="myClass"
```

- **Selecting by Tag**:

```
$('div'); // Selects all <div> elements
```

### 2.2 Manipulating Elements

jQuery simplifies DOM manipulation.

- **Changing Content**:

```
$('#myElement').text('New Content'); // Sets the text content
$('#myElement').html('<strong>New Content</strong>'); // Sets HTML content
```

- **Modifying Attributes**:

```
$('#myElement').attr('src', 'new-image.jpg'); // Changes the 'src' attribute
```

- **Adding/Removing Classes**:

```
$('#myElement').addClass('active'); // Adds the class "active"
$('#myElement').removeClass('active'); // Removes the class "active"
```

### 2.3 Event Handling

jQuery simplifies event handling, making it easy to add event listeners.

- **Click Event**:

```
$('#myButton').click(function() {
    alert('Button clicked!');
});
```

- **Hover Event**:

```
$('#myElement').hover(
    function() { $(this).addClass('hover'); },  // Mouse enter
    function() { $(this).removeClass('hover'); } // Mouse leave
);
```

### 2.4 Animations

jQuery provides methods to create animations and effects.

- **Show/Hide Elements**:

```
$('#myElement').hide(); // Hides the element
$('#myElement').show(); // Shows the element
```

- **Fading In/Out**:

```
$('#myElement').fadeIn(); // Fades in the element
$('#myElement').fadeOut(); // Fades out the element
```

- **Sliding Up/Down**:

```
$('#myElement').slideUp(); // Slides up (hides) the element
$('#myElement').slideDown(); // Slides down (shows) the element
```

### 3. AJAX with jQuery

jQuery simplifies making asynchronous requests with AJAX.

#### 3.1 Basic AJAX Request

- **GET Request**:

```
$.get('https://api.example.com/data', function(data) {
    console.log(data);
});
```

- **POST Request**:

```
$.post('https://api.example.com/submit',    {    key:    'value'    },
function(response) {
    console.log(response);
});
```

#### 3.2 AJAX Methods

jQuery provides several methods for performing AJAX requests.

- **$.ajax() Method**: A more flexible method for making requests.

```
$.ajax({
    url: 'https://api.example.com/data',
    method: 'GET',
    success: function(response) {
        console.log(response);
    },
    error: function(xhr, status, error) {
        console.error('Error:', error);
    }
});
```

- **$.getJSON() Method**: Specifically for JSON data.

```
$.getJSON('https://api.example.com/data', function(data) {
    console.log(data);
});
```

## 4. jQuery Plugins and Best Practices

### 4.1 jQuery Plugins

jQuery's extensibility is enhanced through plugins. Plugins are reusable pieces of code that extend jQuery's functionality. Examples include:

- **Slider Plugins**: For creating image sliders or carousels.
- **Form Validation Plugins**: For validating user input on forms.

### 4.2 Best Practices

- **Use $() Selectors Wisely**: Minimize the number of selectors to improve performance.
- **Avoid Global Namespace Pollution**: Use IIFE (Immediately Invoked Function Expressions) to encapsulate code.
- **Use Modern Alternatives**: Consider using vanilla JavaScript or modern libraries/frameworks if jQuery's features are no longer necessary.

### 4.3 Migrating from jQuery

If transitioning to a framework like React or Vue, consider gradually migrating from jQuery to avoid rewriting code.

## PHP (Hypertext Preprocessor)

PHP is a widely-used open-source scripting language designed for web development but is also used as a general-purpose language. It enables the creation of dynamic and interactive web applications and is often embedded within HTML.

### 1. Introduction to PHP

**1.1 What is PHP?** PHP stands for Hypertext Preprocessor. It is a server-side scripting language designed primarily for web development but also used as a general-purpose language. PHP scripts are executed on the server, and the result is sent to the client's browser as plain HTML.

**1.2 Key Features of PHP:**

- **Server-Side Execution**: PHP code is executed on the server, generating HTML that is sent to the client.
- **Embedded in HTML**: PHP code can be embedded directly within HTML code.
- **Database Interaction**: PHP has built-in support for interacting with various databases, especially MySQL.
- **Cross-Platform**: Runs on multiple operating systems, including Windows, Linux, and macOS.

**1.3 Setting Up PHP:** To run PHP code, you need:

- A web server (e.g., Apache, Nginx).
- PHP installed on the server.
- Optionally, a database system like MySQL.

**Example Installation Using XAMPP:**

- Download and install XAMPP from apachefriends.org.
- Start the Apache server and MySQL database from the XAMPP control panel.

### 2. Basic PHP Syntax

**2.1 PHP Tags:** PHP code is enclosed within `<?php` and `?>` tags.

**Example:**

```
<?php
echo "Hello, World!";
?>
```

### 2.2 Variables:

- **Declaring Variables**: Variables in PHP start with a dollar sign `$` and do not require explicit type declaration.

```
$name = "Alice"; // String
$age = 30;       // Integer
```

- **Variable Types**:
  - **String**: `"Hello"`
  - **Integer**: `42`
  - **Float**: `3.14`
  - **Boolean**: `true`, `false`
  - **Array**: `array(1, 2, 3)`
  - **Object**: Instances of classes
  - **NULL**: Represents a variable with no value.

### 2.3 Concatenation:

```
$fullName = $firstName . " " . $lastName;
```

### 2.4 Comments:

- **Single-Line Comment**: `// This is a comment`
- **Multi-Line Comment**: `/* This is a multi-line comment */`

### 2.5 Echo and Print:

```
echo "Hello, World!"; // Outputs text to the browser
print "Hello, World!"; // Similar to echo but returns 1 on success
```

## 3. Control Structures

### 3.1 Conditional Statements:

- **If Statement:**

```
if ($age >= 18) {
    echo "Adult";
} else {
    echo "Minor";
}
```

- **Switch Statement:**

```
switch ($day) {
    case 1:
        echo "Monday";
        break;
    case 2:
        echo "Tuesday";
        break;
```

```
        default:
            echo "Weekend";
    }
```

### 3.2 Loops:

- **For Loop:**

```
for ($i = 0; $i < 5; $i++) {
    echo $i;
}
```

- **While Loop:**

```
$count = 0;
while ($count < 5) {
    echo $count;
    $count++;
}
```

- **Do-While Loop:**

```
$num = 0;
do {
    echo $num;
    $num++;
} while ($num < 5);
```

## 4. Functions

### 4.1 Defining Functions:

```
function greet($name) {
    return "Hello, " . $name;
}

echo greet("Alice"); // Output: Hello, Alice
```

### 4.2 Function Parameters and Return Values:

- **Default Parameters:**

```
function multiply($a, $b = 2) {
    return $a * $b;
}
```

- **Variable-Length Arguments:**

```
function sum(...$numbers) {
    return array_sum($numbers);
}
```

## 5. Arrays

### 5.1 Indexed Arrays:

```
$colors = array("Red", "Green", "Blue");
echo $colors[1]; // Output: Green
```

### 5.2 Associative Arrays:

```
$person = array("first_name" => "John", "last_name" => "Doe");
echo $person["first_name"]; // Output: John
```

### 5.3 Multidimensional Arrays:

```
$matrix = array(
    array(1, 2, 3),
    array(4, 5, 6),
    array(7, 8, 9)
);
echo $matrix[1][2]; // Output: 6
```

## 6. Working with Forms

### 6.1 Handling Form Data:

- **GET Method:**

```
<form method="get" action="form.php">
    <input type="text" name="username">
    <input type="submit">
</form>

// form.php
<?php
$username = $_GET['username'];
echo "Username: " . htmlspecialchars($username);
?>
```

- **POST Method:**

```
<form method="post" action="form.php">
    <input type="text" name="username">
    <input type="submit">
</form>

// form.php
<?php
$username = $_POST['username'];
echo "Username: " . htmlspecialchars($username);
?>
```

### 6.2 Validating Form Data:

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST['username'];
    if (empty($username)) {
        echo "Username is required";
    } else {
        echo "Username: " . htmlspecialchars($username);
    }
}
```

## 7. Database Interaction

### 7.1 Connecting to MySQL with MySQLi:

```
$mysqli = new mysqli("localhost", "username", "password", "database");

// Check connection
if ($mysqli->connect_error) {
    die("Connection failed: " . $mysqli->connect_error);
}
```

### 7.2 Executing Queries:

- **Selecting Data:**

```
$result = $mysqli->query("SELECT * FROM users");
while ($row = $result->fetch_assoc()) {
    echo $row['username'];
}
```

- **Inserting Data:**

```
$mysqli->query("INSERT INTO users (username, password) VALUES
('Alice', 'password123')");
```

### 7.3 Closing the Connection:

```
$mysqli->close();
```

### 7.4 Using PDO (PHP Data Objects) for Database Interaction:

- **Connecting to a Database:**

```
$pdo = new PDO('mysql:host=localhost;dbname=database', 'username',
'password');
```

- **Executing Queries:**

```
$stmt = $pdo->query("SELECT * FROM users");
while ($row = $stmt->fetch()) {
    echo $row['username'];
}
```

## 8. Error Handling

### 8.1 Error Reporting:

```
error_reporting(E_ALL);
ini_set('display_errors', 1);
```

### 8.2 Handling Errors:

- **Try-Catch Block:**

```
try {
    // Code that may throw an exception
    throw new Exception("An error occurred");
} catch (Exception $e) {
    echo 'Caught exception: ', $e->getMessage(), "\n";
}
```

### 8.3 Custom Error Handlers:

```
function customError($errno, $errstr) {
    echo "Error [$errno]: $errstr";
}
set_error_handler("customError");
```

**Program 1**

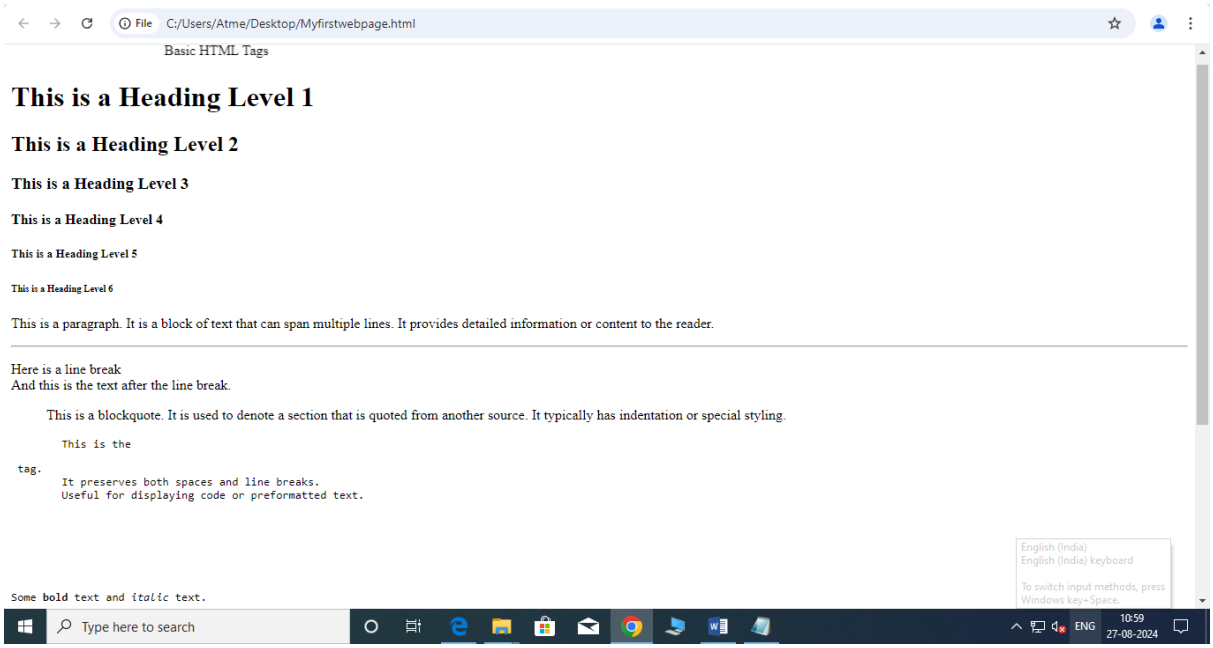
1. Develop the HTML page named as “Myfirstwebpage.html”. Add the following tags with relevant content
   1. Set the title of the page as “My First Web Page”
   2. Within the body use the following tags:
      a) Moving text = “Basic HTML Tags”
      b) Different heading tags (h1 to h6)
      c) Paragraph
      d) Horizontal line
      e) Line Break
      f) Block Quote
      g) Pre tag
      h) Different Logical Style (<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>
</head>
<body>
  <!-- Moving text -->
  <marquee>Basic HTML Tags</marquee>

  <!-- Different heading tags -->
  <h1>This is a Heading Level 1</h1>
  <h2>This is a Heading Level 2</h2>
  <h3>This is a Heading Level 3</h3>
  <h4>This is a Heading Level 4</h4>
  <h5>This is a Heading Level 5</h5>
  <h6>This is a Heading Level 6</h6>

  <!-- Paragraph -->
  <p>This is a paragraph. It is a block of text that can span multiple lines. It provides detailed information or content to the reader.</p>

  <!-- Horizontal line -->
  <hr>

  <!-- Line Break -->
  <p>Here is a line break<br>And this is the text after the line break.</p>
```

```
<!-- Block Quote -->
  <blockquote>
    <p>This is a blockquote. It is used to denote a section that is quoted from another source. It typically has indentation or special styling.</p>
  </blockquote>

  <!-- Pre tag -->
  <pre>
    This is the <pre> tag.
    It preserves both spaces and line breaks.
    Useful for displaying code or preformatted text.
  </pre>

  <!-- Different Logical Style -->
  <p>Some <b>bold</b> text and <i>italic</i> text.</p>
  <p>Here is some <u>underlined</u> text.</p>
  <p>Here is a subscript: H<sub>2</sub>O</p>
  <p>Here is a superscript: E = mc<sup>2</sup></p>

</body>
</html>
```

## Output



**Explanation of the Tags Used:**

1. **`<title>`**: Sets the title of the page that appears on the browser tab.
2. **`<marquee>`**: Creates scrolling text (Note: `<marquee>` is obsolete and not recommended for modern web practices, but included here as per the requirement).
3. **`<h1> to <h6>`**: Represent different levels of headings.
4. **`<p>`**: Defines a paragraph.
5. **`<hr>`**: Inserts a horizontal line.
6. **`<br>`**: Adds a line break.
7. **`<blockquote>`**: Used for longer quotations or to indicate a block of quoted text.
8. **`<pre>`**: Displays preformatted text, preserving both spaces and line breaks.
9. **Logical styles**: `<b>`, `<i>`, `<u>`, `<sub>`, and `<sup>` are used for bold, italic, underline, subscript, and superscript text, respectively.

**Program 2**

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 colours.)
   c) Provide colour options for rows.

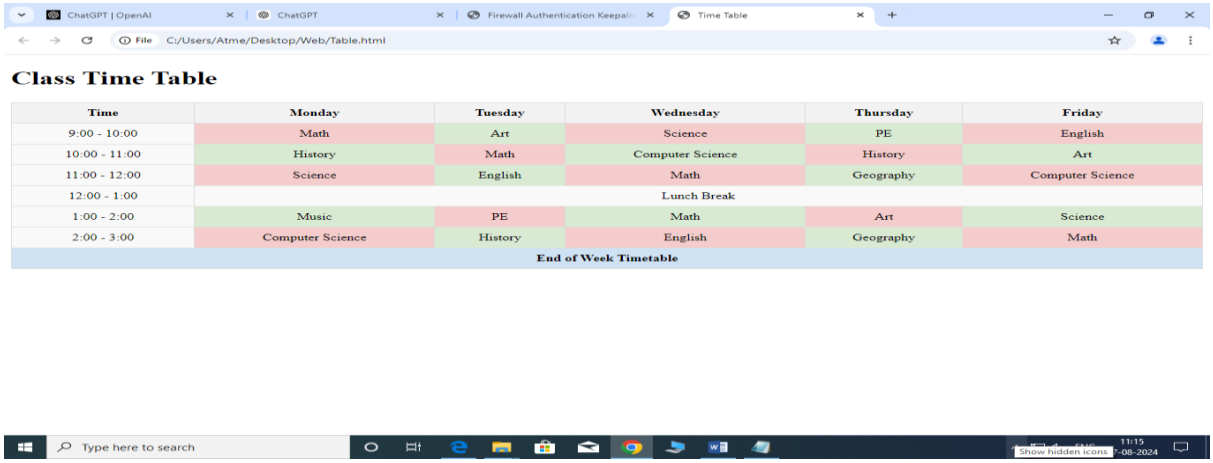
**Program**

```
<!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>
    table {
      width: 100%;
      border-collapse: collapse;
    }
    th, td {
      border: 1px solid #ddd;
      padding: 8px;
      text-align: center;
    }
    th {
      background-color: #f2f2f2;
    }
    .lab-hour {
      background-color: #f4cccc; /* Light red for lab hours */
    }
    .elective-hour {
      background-color: #d9ead3; /* Light green for elective hours */
    }
    .highlight-row {
      background-color: #f9f9f9; /* Light grey for rows */
    }
    .header {
      background-color: #b6d7a8; /* Light green for header */
      font-weight: bold;
    }
    .footer {
      background-color: #cfe2f3; /* Light blue for footer */
      font-weight: bold;
    }
  </style>
```

```
</head>
<body>
  <h1>Class Time Table</h1>

  <table>
    <!-- Table Header -->
    <thead>
      <tr class="header">
        <th>Time</th>
        <th>Monday</th>
        <th>Tuesday</th>
        <th>Wednesday</th>
        <th>Thursday</th>
        <th>Friday</th>
      </tr>
    </thead>
    <!-- Table Body -->
    <tbody>
      <tr class="highlight-row">
        <td>9:00 - 10:00</td>
        <td class="lab-hour">Math</td>
        <td class="elective-hour">Art</td>
        <td class="lab-hour">Science</td>
        <td class="elective-hour">PE</td>
        <td class="lab-hour">English</td>
      </tr>
      <tr class="highlight-row">
        <td>10:00 - 11:00</td>
        <td class="elective-hour">History</td>
        <td class="lab-hour">Math</td>
        <td class="elective-hour">Computer Science</td>
        <td class="lab-hour">History</td>
        <td class="elective-hour">Art</td>
      </tr>
      <tr class="highlight-row">
        <td>11:00 - 12:00</td>
        <td class="lab-hour">Science</td>
        <td class="elective-hour">English</td>
        <td class="lab-hour">Math</td>
        <td class="elective-hour">Geography</td>
        <td class="lab-hour">Computer Science</td>
      </tr>
      <tr class="highlight-row">
        <td>12:00 - 1:00</td>
        <td colspan="5">Lunch Break</td>
      </tr>
      <tr class="highlight-row">
        <td>1:00 - 2:00</td>
```

```
            <td class="elective-hour">Music</td>
            <td class="lab-hour">PE</td>
            <td class="elective-hour">Math</td>
            <td class="lab-hour">Art</td>
            <td class="elective-hour">Science</td>
          </tr>
          <tr class="highlight-row">
            <td>2:00 - 3:00</td>
            <td class="lab-hour">Computer Science</td>
            <td class="elective-hour">History</td>
            <td class="lab-hour">English</td>
            <td class="elective-hour">Geography</td>
            <td class="lab-hour">Math</td>
          </tr>
        </tbody>
        <!-- Table Footer -->
        <tfoot>
          <tr class="footer">
            <td colspan="6">End of Week Timetable</td>
          </tr>
        </tfoot>
      </table>

</body>
</html>
```

## Output




| Time | Monday | Tuesday | Wednesday | Thursday | Friday |
|---|---|---|---|---|---|
| 9:00 - 10:00 | Math | Art | Science | PE | English |
| 10:00 - 11:00 | History | Math | Computer Science | History | Art |
| 11:00 - 12:00 | Science | English | Math | Geography | Computer Science |
| 12:00 - 1:00 | Lunch Break | | | | |
| 1:00 - 2:00 | Music | PE | Math | Art | Science |
| 2:00 - 3:00 | Computer Science | History | English | Geography | Math |
| End of Week Timetable | | | | | |

**Explanation of the HTML and CSS:**

1. **`<title>`**: Sets the title of the page as "Time Table."
2. **Table Header (`<thead>`)**:
   - Contains the column titles for the timetable.
3. **Table Body (`<tbody>`)**:
   - Contains the timetable data.
   - Uses `class="highlight-row"` to style alternating rows.
   - Utilizes classes like `lab-hour` and `elective-hour` to apply different background colors for lab and elective hours.
   - Uses `colspan` to merge cells across multiple columns for the lunch break.
4. **Table Footer (`<tfoot>`)**:
   - Provides a footer row that spans all columns with a summary or note about the table.
5. **CSS Styles**:
   - Define the appearance of the table, including borders, padding, and background colors for different types of hours and rows.

**Program 3**

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.

**Program**

**style.css**

```
/* Universal selector for basic styling */
* {
   margin: 0;
   padding: 0;
   box-sizing: border-box;
}

/* Styling for h2 tag */
h2 {
   color: #2c3e50; /* Dark blue color */
   font-family: Arial, sans-serif;
   font-size: 24px;
   margin-bottom: 20px;
}

/* Styling for h3 tag */
h3 {
   color: #16a085; /* Teal color */
   font-family: 'Courier New', Courier, monospace;
   font-size: 20px;
   text-transform: uppercase;
}

/* Styling for horizontal rule */
hr {
   border: none;
   height: 2px;
   background-color: #3498db; /* Blue color */
   margin: 20px 0;
}

/* Styling for p tag */
p {
   color: #34495e; /* Grey color */
   font-family: Georgia, serif;
   font-size: 16px;
   line-height: 1.6;
   margin-bottom: 15px;
```

```
}

/* Styling for div tag */
div {
   background-color: #ecf0f1; /* Light grey background */
   border: 1px solid #bdc3c7; /* Border color */
   padding: 15px;
   margin-bottom: 20px;
   border-radius: 5px;
}

/* Styling for span tag */
span {
   color: #e74c3c; /* Red color */
   font-weight: bold;
}

/* Styling for time tag */
time {
   color: #9b59b6; /* Purple color */
   font-style: italic;
}

/* Styling for img tag */
img {
   max-width: 100%;
   height: auto;
   border: 2px solid #3498db; /* Blue border */
   border-radius: 10px;
}

/* Styling for a tag */
a {
   color: #2980b9; /* Blue color */
   text-decoration: none;
   font-weight: bold;
}

a:hover {
   text-decoration: underline;
   color: #1f618d; /* Darker blue on hover */
}
```

**Explanation of the CSS Styles and Selectors:**

1. **Universal Selector (`*`)**:
   - Applies basic margin, padding, and box-sizing to all elements, ensuring consistency across different elements.
2. **Element Selector (`h2`, `h3`, `hr`, `p`, `div`, `span`, `time`, `img`, `a`)**:
   - Targets specific HTML elements to apply general styling.
3. **`h2`**:
   - Sets color, font-family, font-size, and margin for `h2` headers.
4. **`h3`**:
   - Applies color, font-family, font-size, and text transformation to `h3` headers.
5. **`hr`**:
   - Styles horizontal rules with background color, height, and margin.
6. **`p`**:
   - Defines color, font-family, font-size, line-height, and margin for paragraphs.
7. **`div`**:
   - Adds background color, border, padding, margin, and border-radius to `div` elements.
8. **`span`**:
   - Styles `span` elements with color and font-weight.
9. **`time`**:
   - Applies color and font-style to `time` elements.
10. **`img`**:
    - Sets maximum width, auto height, border, and border-radius for images, ensuring responsiveness.
11. **`a`**:
    - Styles links with color, removes underline by default, and adds bold font weight.
    - `a:hover` pseudo-class changes the link appearance when hovered over, adding underline and changing the color.

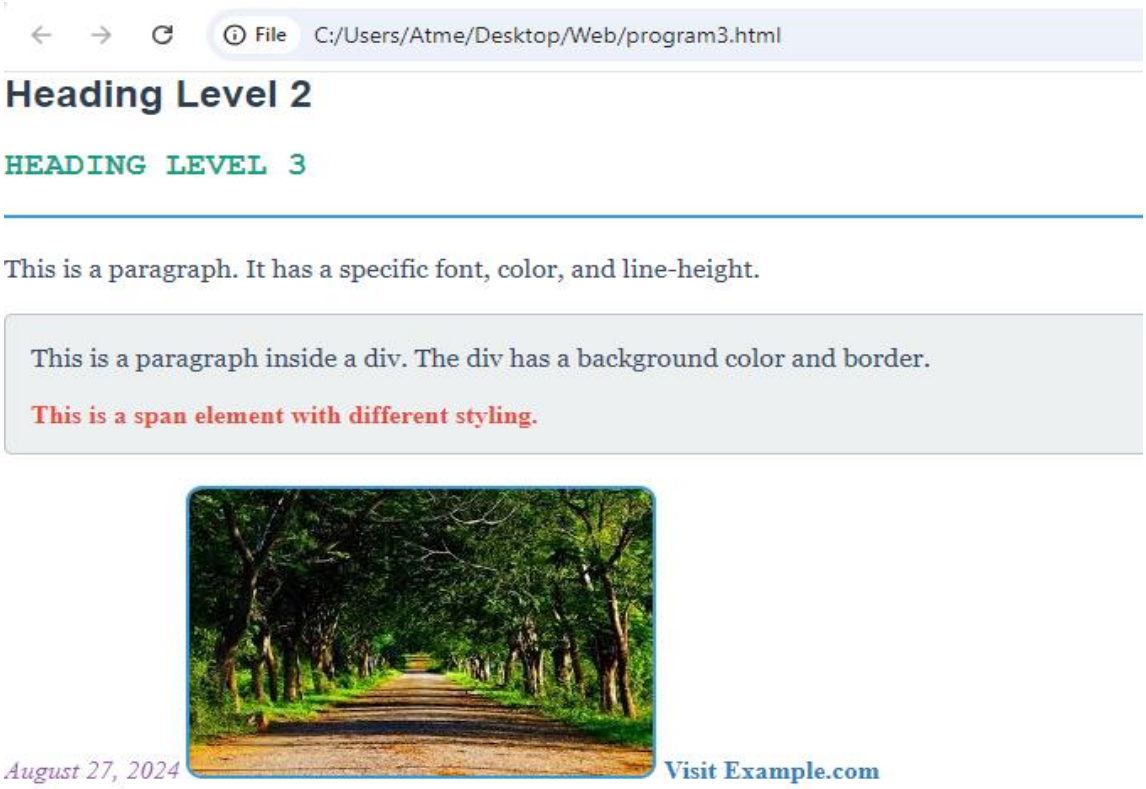
### Applying the Styles

To use these styles in an HTML document, link the `style.css` file in the `<head>` section of your HTML file:

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Styled Page</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h2>Heading Level 2</h2>
  <h3>Heading Level 3</h3>
  <hr>
  <p>This is a paragraph. It has a specific font, color, and line-height.</p>
```

```
  <div>
    <p>This is a paragraph inside a div. The div has a background color and border.</p>
    <span>This is a span element with different styling.</span>
  </div>
  <time datetime="2024-08-27">August 27, 2024</time>
  <img src="example.jpg" alt="Example Image">
  <a href="https://www.example.com">Visit Example.com</a>
</body>
</html>
```

**Output**



**Program 4**

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.

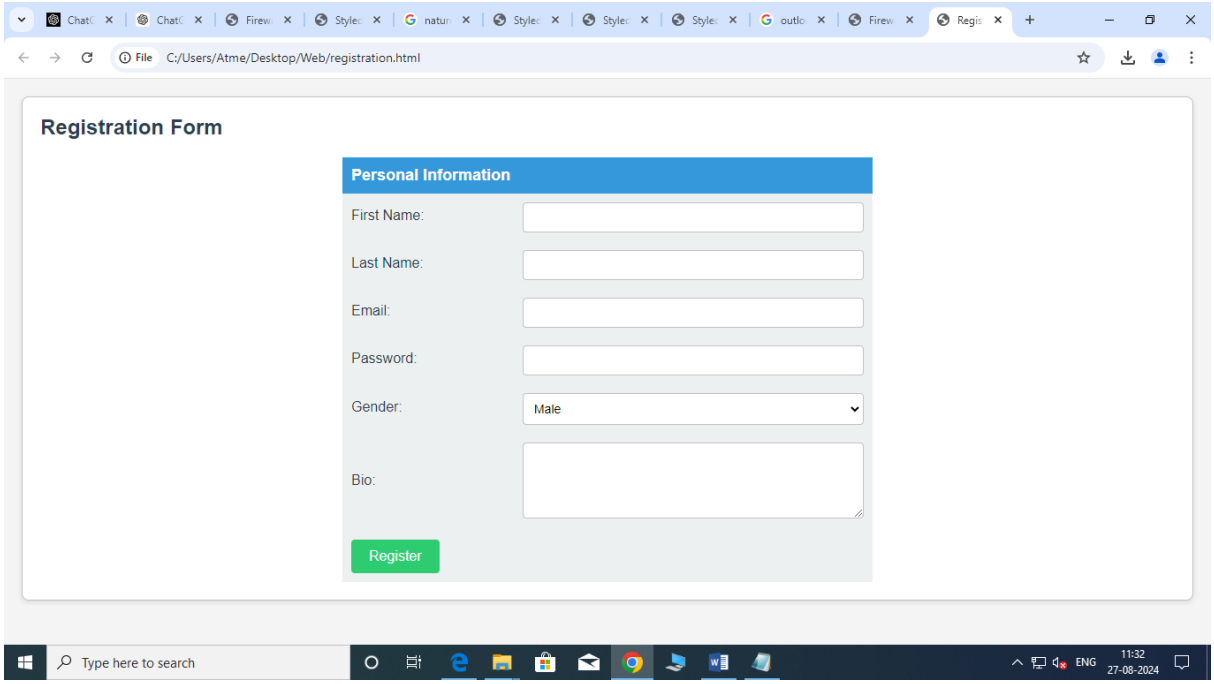
**Program**

```
<!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>
  <link rel="stylesheet" href="style.css">
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
      margin: 0;
      padding: 20px;
    }
    table {
      width: 100%;
      max-width: 600px;
      margin: 0 auto;
      border-collapse: collapse;
    }
    th, td {
      padding: 10px;
      text-align: left;
    }
    th {
      background-color: #3498db;
      color: #fff;
      font-size: 18px;
    }
    td {
      background-color: #ecf0f1;
    }
    label {
      display: block;
      margin-bottom: 5px;
      font-size: 16px;
      color: #2c3e50;
    }
    input[type="text"],
    input[type="email"],
```

```
      input[type="password"],
      select,
      textarea {
         width: 100%;
         padding: 8px;
         border: 1px solid #bdc3c7;
         border-radius: 4px;
         font-size: 14px;
      }
      input[type="submit"] {
         background-color: #2ecc71;
         color: #fff;
         border: none;
         padding: 10px 20px;
         border-radius: 4px;
         font-size: 16px;
         cursor: pointer;
      }
      input[type="submit"]:hover {
         background-color: #27ae60;
      }
      .form-container {
         background-color: #fff;
         padding: 20px;
         border-radius: 8px;
         box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
      }
      .form-title {
         font-size: 24px;
         color: #2c3e50;
         margin-bottom: 20px;
      }
   </style>
</head>
<body>
   <div class="form-container">
      <h1 class="form-title">Registration Form</h1>
      <form action="#" method="post">
         <table>
            <tr>
               <th colspan="2">Personal Information</th>
            </tr>
            <tr>
               <td><label for="first-name">First Name:</label></td>
               <td><input type="text" id="first-name" name="first-name" required></td>
            </tr>
            <tr>
               <td><label for="last-name">Last Name:</label></td>
```

```
            <td><input type="text" id="last-name" name="last-name" 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="gender">Gender:</label></td>
            <td>
              <select id="gender" name="gender">
                <option value="male">Male</option>
                <option value="female">Female</option>
                <option value="other">Other</option>
              </select>
            </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">
              <input type="submit" value="Register">
            </td>
          </tr>
        </table>
      </form>
    </div>
  </body>
  </html>
```

## Output




## Explanation of the HTML and CSS:

1. **HTML Structure**:
   - The form is placed inside a `div` with the class `form-container` for styling.
   - A table is used for layout alignment, with each row containing labels and corresponding input fields.
   - Various input types (`text`, `email`, `password`, `select`, `textarea`) are included to demonstrate different form elements.
2. **CSS Styles**:
   - **`body`**: Sets a background color and padding for the entire page.
   - **`table`**: Centers the table and sets its width. Uses `border-collapse` to ensure borders are collapsed into a single line.
   - **`th`**: Styles table headers with a background color and white text.
   - **`td`**: Styles table data cells with a light grey background.
   - **`label`**: Styles labels with font size and color.
   - **`input`, `select`, `textarea`**: Applies styles to form elements including padding, border, and font size.
   - **`input[type="submit"]`**: Styles the submit button with a green background, white text, and hover effect.
   - **`.form-container`**: Adds padding, background color, border radius, and box shadow to the form container for a card-like appearance.
   - **`.form-title`**: Styles the form title with font size and color.

**Program 5**

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.

**Program**

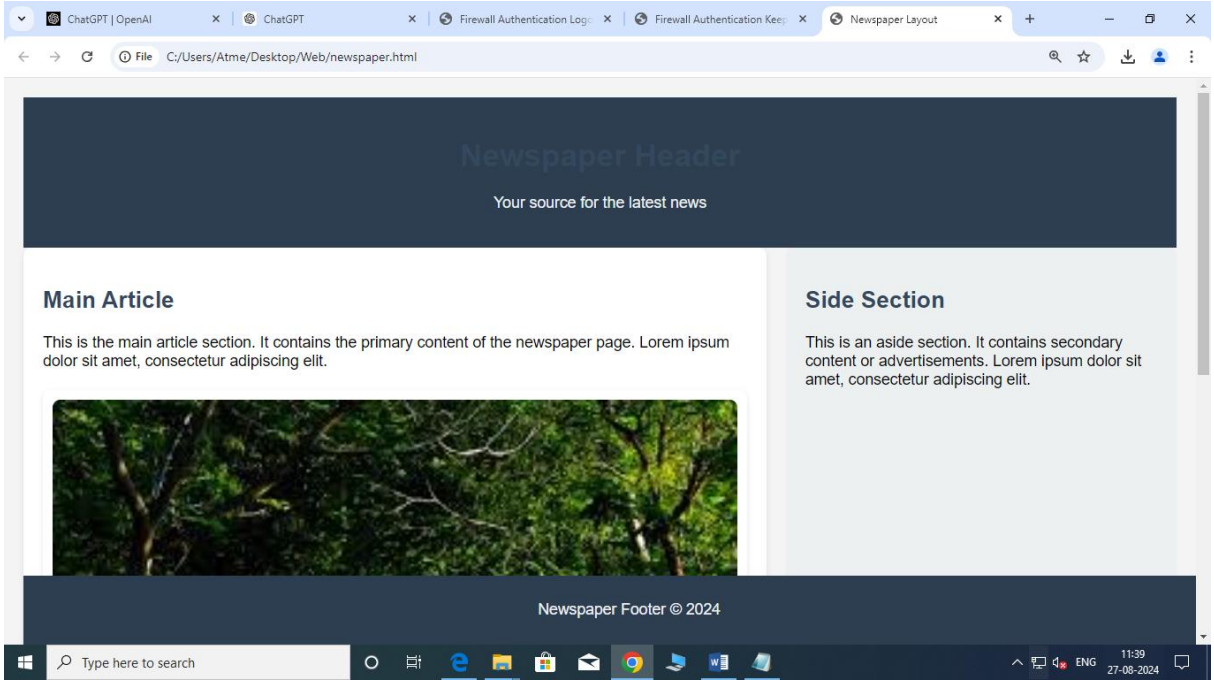
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Newspaper Layout</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
      margin: 0;
      padding: 20px;
    }
    header {
      background-color: #2c3e50;
      color: #ecf0f1;
      padding: 20px;
      text-align: center;
    }
    footer {
      background-color: #2c3e50;
      color: #ecf0f1;
      padding: 10px;
      text-align: center;
      position: fixed;
      bottom: 0;
      width: 100%;
    }
    main {
      display: flex;
      gap: 20px;
      margin-bottom: 60px; /* To ensure footer does not overlap content */
    }
    article {
      flex: 2;
      background-color: #fff;
      padding: 20px;
      border-radius: 8px;
      box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    }
```

```
    aside {
      flex: 1;
      background-color: #ecf0f1;
      padding: 20px;
      border-radius: 8px;
    }
    section {
      margin-bottom: 20px;
    }
    h1, h2, h3 {
      color: #34495e;
    }
    table {
      width: 100%;
      border-collapse: collapse;
      margin-top: 20px;
    }
    table, th, td {
      border: 1px solid #bdc3c7;
    }
    th {
      background-color: #3498db;
      color: #fff;
      padding: 10px;
    }
    td {
      background-color: #fff;
      padding: 10px;
      text-align: left;
    }
    figure {
      margin: 0;
      padding: 0;
    }
    figcaption {
      font-size: 14px;
      color: #7f8c8d;
      text-align: center;
    }
    .figure-container {
      background-color: #fff;
      padding: 10px;
      border-radius: 8px;
      box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
      margin: 20px 0;
    }
  </style>
</head>
```

```
<body>
  <header>
    <h1>Newspaper Header</h1>
    <p>Your source for the latest news</p>
  </header>

  <main>
    <article>
      <section>
        <h2>Main Article</h2>
        <p>This is the main article section. It contains the primary content of the newspaper page. Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
        <figure class="figure-container">
          <img src="example.jpg" alt="Example Image" style="width: 100%; border-radius: 8px;">
          <figcaption>Figure 1: Example image caption.</figcaption>
        </figure>
        <table>
          <caption>Sample Table</caption>
          <thead>
            <tr>
              <th>Header 1</th>
              <th>Header 2</th>
              <th>Header 3</th>
            </tr>
          </thead>
          <tbody>
            <tr>
              <td>Data 1</td>
              <td>Data 2</td>
              <td>Data 3</td>
            </tr>
            <tr>
              <td>Data 4</td>
              <td>Data 5</td>
              <td>Data 6</td>
            </tr>
          </tbody>
        </table>
      </section>
    </article>

    <aside>
      <h2>Side Section</h2>
      <p>This is an aside section. It contains secondary content or advertisements. Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
    </aside>
  </main>
```

```
  <footer>
    <p>Newspaper Footer © 2024</p>
  </footer>
</body>
</html>
```

## Output




## Explanation of the HTML and CSS:

1. **HTML Structure**:
   - **`<header>`**: Contains the main heading and subtitle of the newspaper.
   - **`<footer>`**: Provides footer information, fixed at the bottom of the page.
   - **`<main>`**: Uses flexbox to layout the `article` and `aside` side by side.
   - **`<article>`**: Represents the main content area of the page.
   - **`<aside>`**: Contains secondary content or sidebar information.
   - **`<section>`**: Used within the `article` to group related content.
   - **`<figure>`** and **`<figcaption>`**: Used to include an image with a caption.
   - **`<table>`**: Contains tabular data with headers and a caption.

2. **CSS Styles**:
   - **`body`**: Sets a background color, font, and padding for the entire page.
   - **`header`** and **`footer`**: Define background colors, text colors, and padding. The footer is positioned fixed at the bottom of the page.
   - **`main`**: Utilizes flexbox to create a responsive layout for the main content and sidebar.

- **`article`** and **`aside`**: Style the primary content and sidebar with background colors, padding, and shadows.
- **`table`**: Styles the table with borders, background colors for headers, and padding.
- **`figure`** and **`figcaption`**: Apply styling to the figure and caption, including background color and text color.

**Program 6**

6. 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.

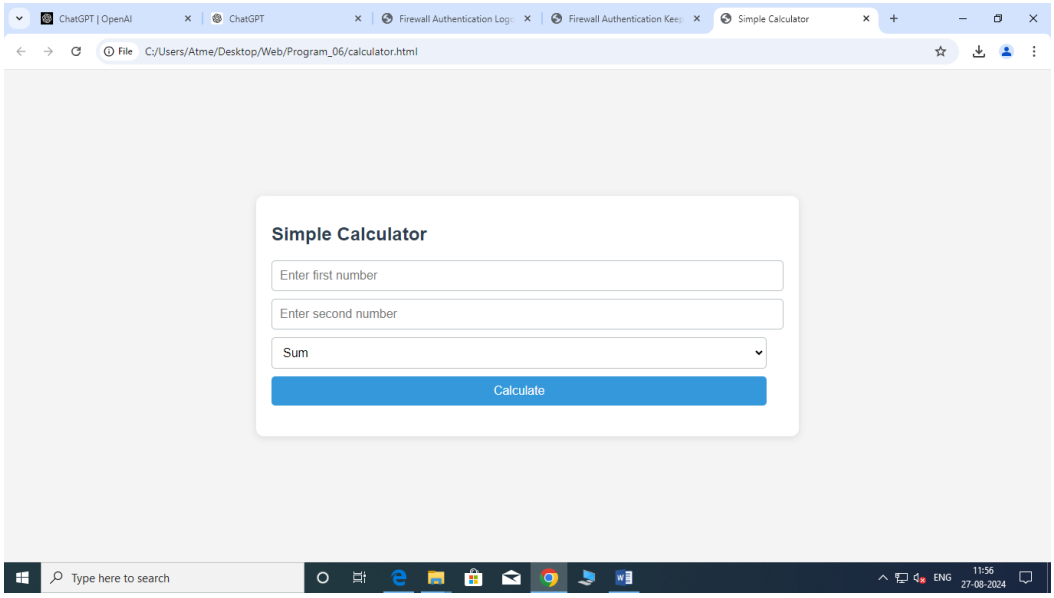
**Program**

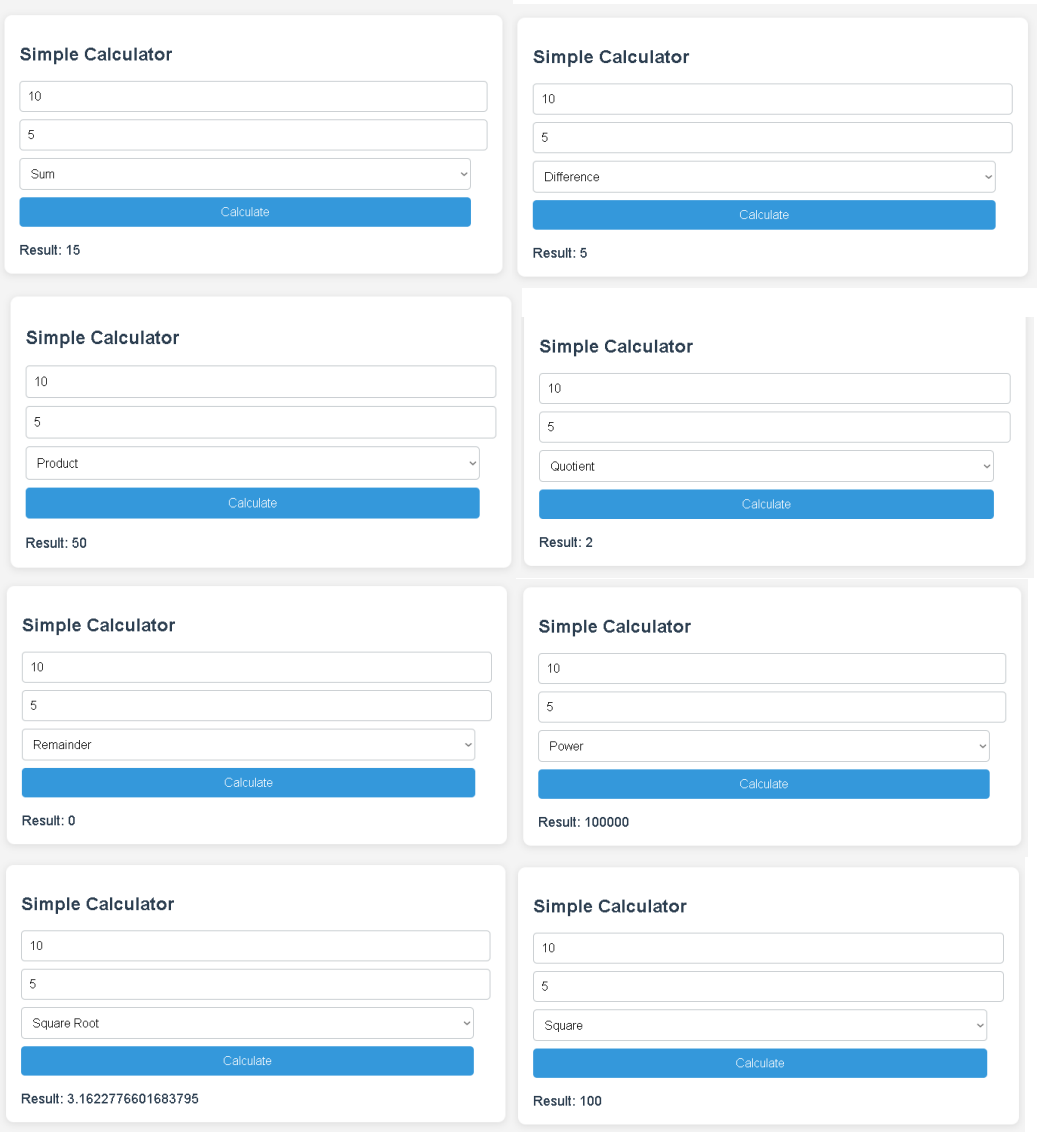
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Simple Calculator</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      background-color: #f4f4f4;
      margin: 0;
    }
    .calculator {
      background-color: #fff;
      padding: 20px;
      border-radius: 10px;
      box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
    }
    .calculator h1 {
      font-size: 24px;
      color: #2c3e50;
      margin-bottom: 20px;
    }
    .calculator input,
    .calculator select,
    .calculator button {
      margin-bottom: 10px;
      padding: 10px;
      border: 1px solid #bdc3c7;
      border-radius: 5px;
      font-size: 16px;
      width: calc(100% - 22px); /* Adjust for padding and border */
    }
    .calculator button {
      background-color: #3498db;
      color: #fff;
      border: none;
```

```
        cursor: pointer;
      }
      .calculator button:hover {
        background-color: #2980b9;
      }
      .calculator .result {
        font-size: 18px;
        color: #2c3e50;
        margin-top: 10px;
      }
    </style>
  </head>
  <body>
    <div class="calculator">
      <h1>Simple Calculator</h1>
      <input type="number" id="num1" placeholder="Enter first number">
      <input type="number" id="num2" placeholder="Enter second number">
      <select id="operation">
        <option value="sum">Sum</option>
        <option value="difference">Difference</option>
        <option value="product">Product</option>
        <option value="quotient">Quotient</option>
        <option value="remainder">Remainder</option>
        <option value="power">Power</option>
        <option value="sqrt">Square Root</option>
        <option value="square">Square</option>
      </select>
      <button onclick="calculate()">Calculate</button>
      <div class="result" id="result"></div>
    </div>

    <script>
      function calculate() {
        // Get values from inputs
        const num1 = parseFloat(document.getElementById('num1').value);
        const num2 = parseFloat(document.getElementById('num2').value);
        const operation = document.getElementById('operation').value;

        // Result variable
        let result;

        // Perform calculation based on selected operation
        switch (operation) {
          case 'sum':
            result = num1 + num2;
            break;
          case 'difference':
            result = num1 - num2;
```

```
            break;
          case 'product':
            result = num1 * num2;
            break;
          case 'quotient':
            result = num1 / num2;
            break;
          case 'remainder':
            result = num1 % num2;
            break;
          case 'power':
            result = Math.pow(num1, num2);
            break;
          case 'sqrt':
            result = Math.sqrt(num1);
            break;
          case 'square':
            result = Math.pow(num1, 2);
            break;
          default:
            result = 'Invalid operation';
        }

        // Display result
        document.getElementById('result').textContent = 'Result: ' + result;
      }
  </script>
</body>
</html>
```

## Output


ChatGPT | OpenAI
ChatGPT
Firewall Authentication Log
Firewall Authentication Keep
Simple Calculator
File C:/Users/Atme/Desktop/Web/Program_06/calculator.html
Simple Calculator
Enter first number
Enter second number
Sum
Calculate
Type here to search
ENG
11:56
27-08-2024


Simple Calculator
10
5
Sum
Calculate
Result: 15
Simple Calculator
10
5
Difference
Calculate
Result: 5
Simple Calculator
10
5
Product
Calculate
Result: 50
Simple Calculator
10
5
Quotient
Calculate
Result: 2
Simple Calculator
10
5
Remainder
Calculate
Result: 0
Simple Calculator
10
5
Power
Calculate
Result: 100000
Simple Calculator
10
5
Square Root
Calculate
Result: 3.1622776601683795
Simple Calculator
10
5
Square
Calculate
Result: 100

**Explanation:**

1. **HTML**:
   - The HTML structure includes:
     - Two input fields for numbers.
     - A `<select>` dropdown to choose the operation.
     - A `<button>` to trigger the calculation.
     - A `<div>` to display the result.
2. **CSS**:
   - Styles are applied to center the calculator on the page, style the form elements, and add hover effects to the button.
   - The `.calculator` class styles the container, adding padding, border-radius, and box-shadow for a clean look.
3. **JavaScript**:
   - The `calculate` function retrieves the values from the input fields and the selected operation from the dropdown.
   - It performs the calculation based on the chosen operation using a `switch` statement.
   - The result is then displayed in the `<div>` with the `id="result"`.

This simple calculator performs all the specified operations, and the JavaScript handles the logic for each operation based on user input.

**Program 7**

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

**Program**

```
<!DOCTYPE html>
<html lang="en">
        <head>
          <meta charset="UTF-8">
          <meta name="viewport" content="width=device-width, initial-scale=1.0">
          <title>JSON and CSV Converter</title>
          <style>
            body {
              font-family: Arial, sans-serif;
              margin: 20px;
            }
            .container {
              margin-bottom: 20px;
            }
            textarea, input[type="text"], button {
              display: block;
              width: 100%;
              margin-bottom: 10px;
              padding: 8px;
              box-sizing: border-box;
            }
            button {
              background-color: #4CAF50;
              color: white;
              border: none;
              cursor: pointer;
            }
            button:hover {
              background-color: #45a049;
            }
            .result {
              white-space: pre-wrap;
              background: #f4f4f4;
              padding: 10px;
            }
          </style>
        </head>
        <body>
```

```
        <div class="container">
          <h2>Convert JSON to JavaScript Object</h2>
          <textarea id="jsonText" placeholder='Enter JSON here'></textarea>
          <button onclick="convertJsonToObject()">Convert JSON</button>
          <div id="jsonObject" class="result"></div>
        </div>

        <div class="container">
          <h2>Convert JSON to Date</h2>
          <input type="text" id="jsonDate" placeholder='Enter JSON Date string'>
          <button onclick="convertJsonToDate()">Convert to Date</button>
          <div id="dateResult" class="result"></div>
        </div>

        <div class="container">
          <h2>Convert JSON to CSV and CSV to JSON</h2>
          <textarea id="jsonCsvText" placeholder='Enter JSON for CSV conversion'></textarea>
          <button onclick="convertJsonToCsv()">JSON to CSV</button>
          <div id="csvResult" class="result"></div>

          <textarea id="csvJsonText" placeholder='Enter CSV text'></textarea>
          <button onclick="convertCsvToJson()">CSV to JSON</button>
          <div id="jsonResult" class="result"></div>
        </div>

        <div class="container">
          <h2>Create Hash from String</h2>
          <input type="text" id="stringInput" placeholder='Enter string to hash'>
          <button onclick="createHash()">Create Hash</button>
          <div id="hashResult" class="result"></div>
        </div>

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

## JavaScript (app.js)

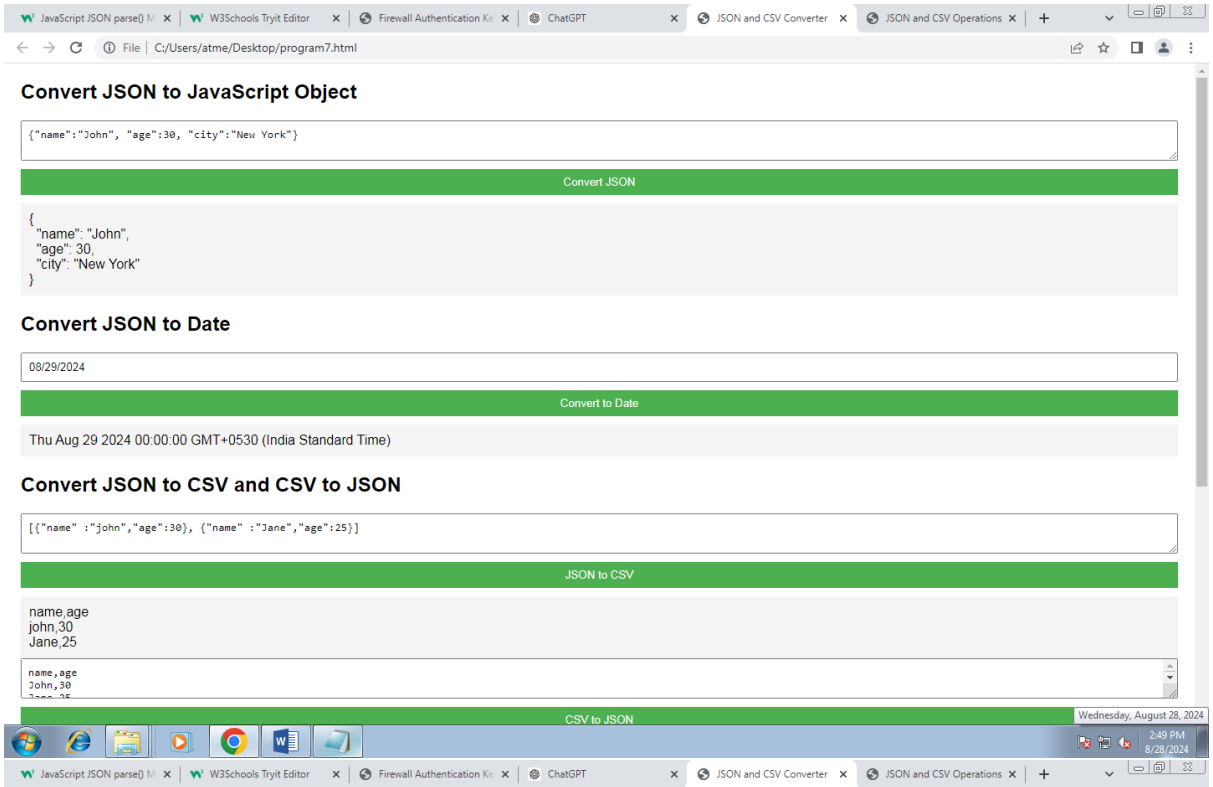
Now let's add the JavaScript code to handle each of the tasks.

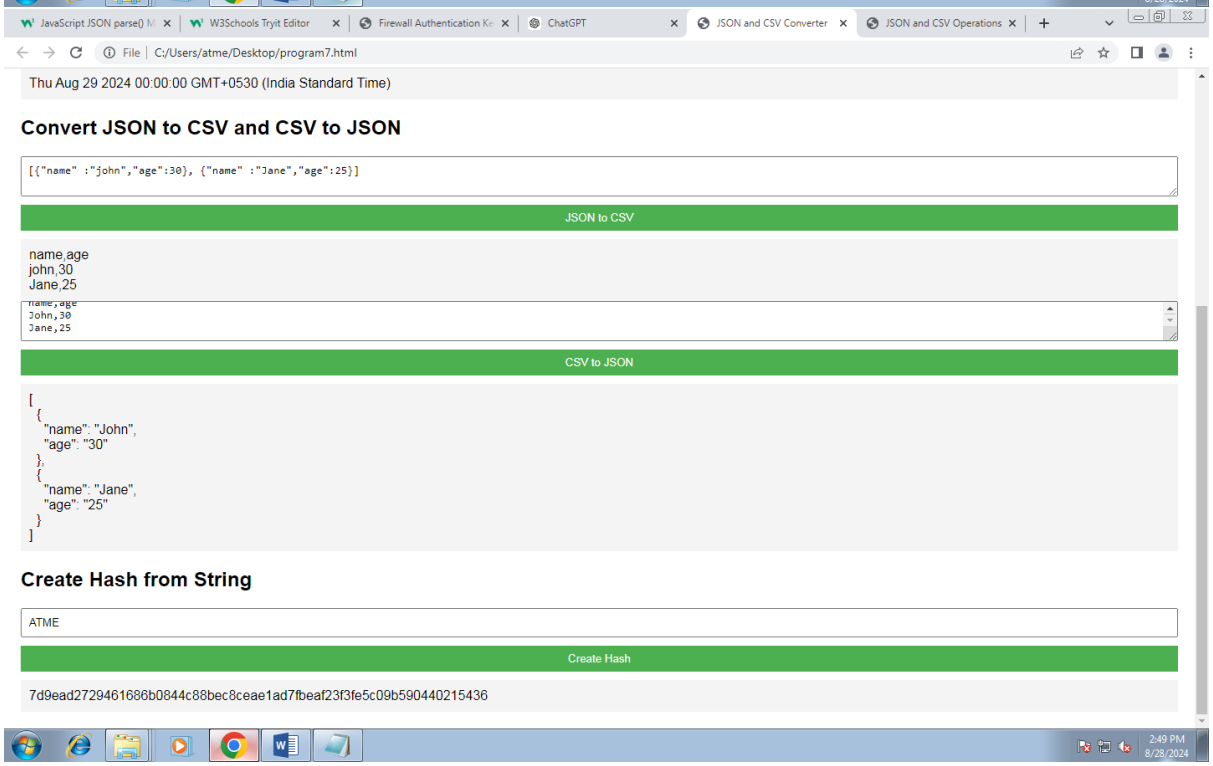
```
      }// Convert JSON text to JavaScript Object
      function convertJsonToObject() {
        const jsonText = document.getElementById('jsonText').value;
        try {
          const obj = JSON.parse(jsonText);
          document.getElementById('jsonObject').textContent = JSON.stringify(obj, null, 2);
        } catch (e) {
```

```
        document.getElementById('jsonObject').textContent = 'Invalid JSON';
    }
}

// Convert JSON date string to Date
function convertJsonToDate() {
    const jsonDate = document.getElementById('jsonDate').value;
    try {
        const date = new Date(jsonDate);
        document.getElementById('dateResult').textContent = date.toString();
    } catch (e) {
        document.getElementById('dateResult').textContent = 'Invalid Date';
    }
}

// Convert JSON to CSV
function convertJsonToCsv() {
    const jsonText = document.getElementById('jsonCsvText').value;
    try {
        const data = JSON.parse(jsonText);
        const csv = Object.keys(data[0]).join(',') + '\n' +
                data.map(row => Object.values(row).join(',')).join('\n');
        document.getElementById('csvResult').textContent = csv;
    } catch (e) {
        document.getElementById('csvResult').textContent = 'Invalid JSON';
    }
}

// Convert CSV to JSON
function convertCsvToJson() {
    const csvText = document.getElementById('csvJsonText').value;
    const rows = csvText.split('\n').map(row => row.split(','));
    const headers = rows.shift();
    const json = rows.map(row => {
        let obj = {};
        row.forEach((value, index) => obj[headers[index]] = value);
        return obj;
    });
    document.getElementById('jsonResult').textContent = JSON.stringify(json, null, 2);
}

// Create hash from string
async function createHash() {
    const crypto = window.crypto || window.msCrypto; // For IE11
    const str = document.getElementById('stringInput').value;
    const encoder = new TextEncoder();
    const data = encoder.encode(str);
    const hashBuffer = await crypto.subtle.digest('SHA-256', data);
```

```
const hashArray = Array.from(new Uint8Array(hashBuffer));
const hashHex = hashArray.map(b => b.toString(16).padStart(2, '0')).join('');
document.getElementById('hashResult').textContent = hashHex;
```

## Output



**Explanation:**

1. **Convert JSON to JavaScript Object**: The function `convertJsonToObject` parses the JSON text and displays it in a formatted way. If the JSON is invalid, an error message is shown.
2. **Convert JSON Date to JavaScript Date**: The function `convertJsonToDate` creates a JavaScript `Date` object from the JSON date string and displays it.
3. **Convert JSON to CSV**: The function `convertJsonToCsv` parses JSON data into a CSV format. It assumes that the input JSON is an array of objects.
4. **Convert CSV to JSON**: The function `convertCsvToJson` parses CSV data into JSON. It assumes the first row of the CSV contains headers.
5. **Create Hash from String**: The function `createHash` uses the Web Crypto API to generate a SHA-256 hash of the input string.

## Program 8

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.

## Program 8a

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.

### Step 1: Create the PHP Script

First, create a PHP file named `program_8a.php`. This script will handle the counting of visitors and display the count on the web page.

```
<?php
// Define the path to the file where the count will be stored
$countFile = 'count.txt';

// Check if the count file exists
if (!file_exists($countFile)) {
   // If the file doesn't exist, create it and initialize the count to 0
   file_put_contents($countFile, '0');
}

// Read the current count from the file
$count = (int)file_get_contents($countFile);

// Increment the count
$count++;

// Save the updated count back to the file
file_put_contents($countFile, $count);

// HTML output
?>
<!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;
         display: flex;
         justify-content: center;
         align-items: center;
```

```
        height: 100vh;
        background-color: #f4f4f4;
        margin: 0;
    }
    .container {
        text-align: center;
        background-color: #fff;
        padding: 20px;
        border-radius: 8px;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }
    h1 {
        color: #333;
    }
    .count {
        font-size: 2em;
        color: #007bff;
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>Visitor Counter</h1>
    <p class="count">You are visitor number: <?php echo $count; ?></p>
  </div>
</body>
</html>
```

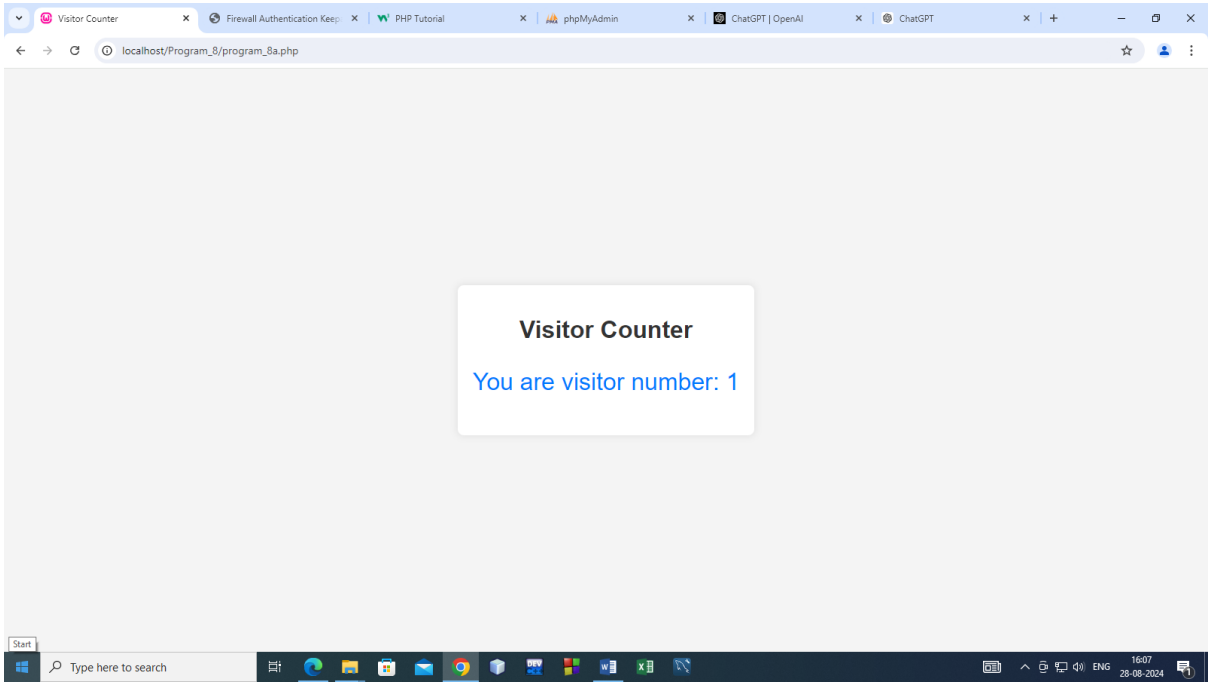
**Step 2: Explanation**

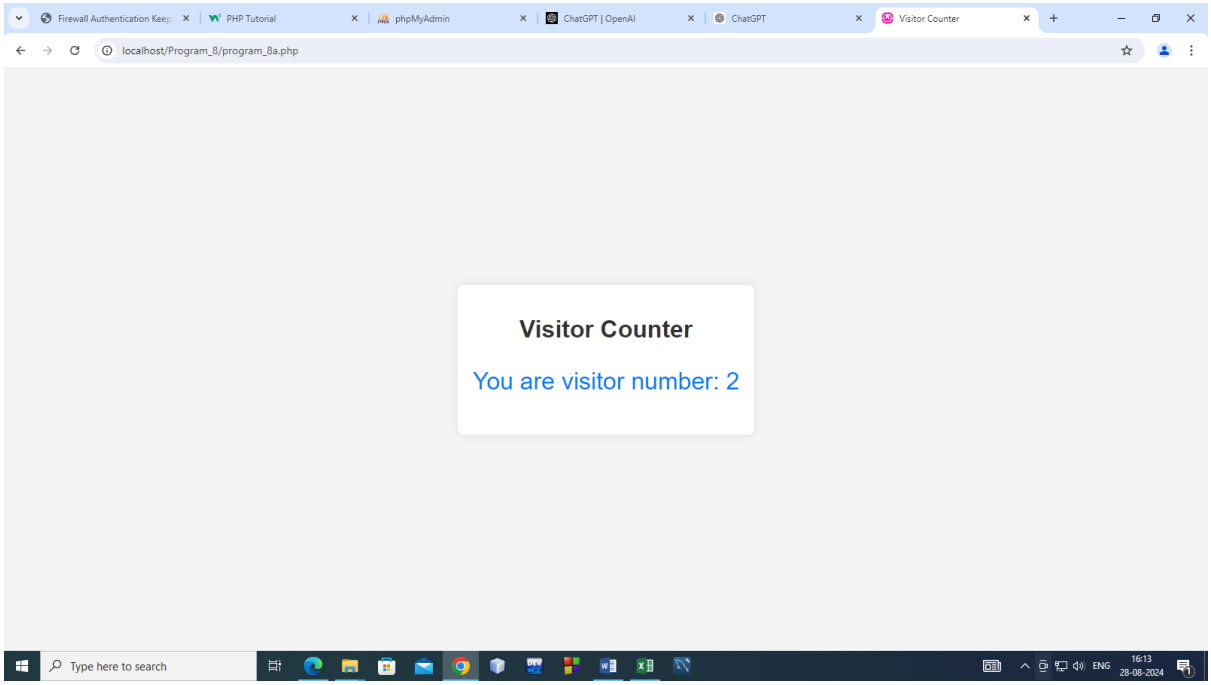
1. **File Handling**: The PHP script uses `count.txt` to keep track of the number of visitors. If this file does not exist, it creates it and initializes the count to 0. Then, it reads the current count, increments it, and writes the updated count back to the file.
2. **HTML/CSS**:
   - **HTML**: The page has a simple structure with a heading and a paragraph that displays the visitor count.
   - **CSS**: Basic styles are applied to center the content and make the page look clean. The container has a box shadow for a subtle effect, and the visitor count is styled to stand out.

**Step 3: Upload and Test**

1. **Upload Files**: Place `program_8a.php` on your web server. Ensure that `count.txt` is writable by the web server (check permissions if needed).
2. **Access Your Page**: Open your web browser and navigate to the location where `program_8a.php` is hosted. You should see the visitor counter and the count will increment with each page load.

## Output


Visitor Counter
localhost/Program_8/program_8a.php
Visitor Counter
You are visitor number: 1
Type here to search


Visitor Counter
localhost/Program_8/program_8a.php
Visitor Counter
You are visitor number: 2
Type here to search

## Program 8b

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

## Steps

1. **Set up your database**: You'll need a MySQL database with a table for student records. For simplicity, let's assume you have a database named `web_lab_students` and a table named `students` with fields `Stu_name`, `USN` and `Sem`.
2. **Create the PHP script**: This script will connect to the database, fetch student records, sort them using selection sort, and then display the results.
3. **Write the HTML/CSS**: This part will handle the presentation of the sorted data.

### 1. Database Setup

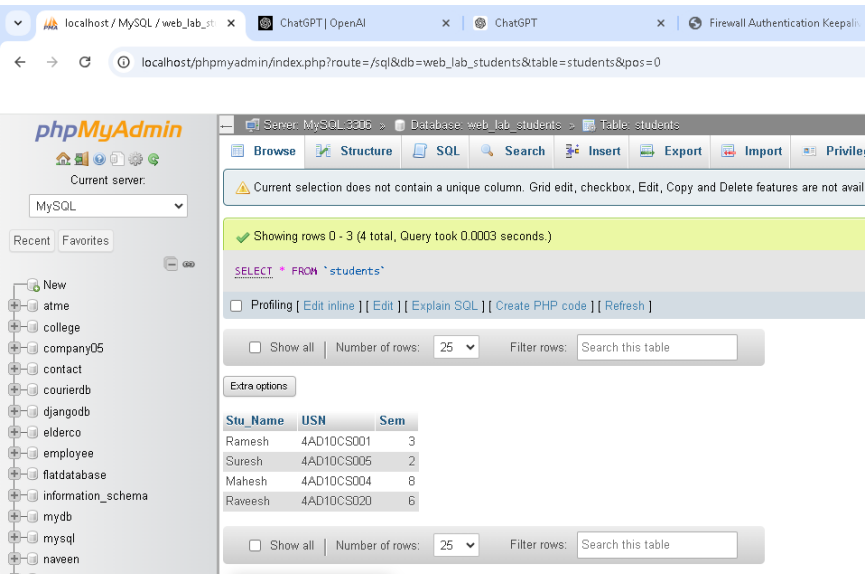
Assuming your database `web_lab_students` and table `students` are already created, here is an example of how the table might look:

CREATE DATABASE `web_lab_students`;

USE `web_lab_students`;

CREATE TABLE `students` (
`Stu_Name` VARCHAR(100),
`USN` VARCHAR(20),
`Sem` INT );

INSERT INTO students (Stu_Name, USN, Sem) VALUES(‘Ramesh’, ‘4AD10CS001’,3)
INSERT INTO students (Stu_Name, USN, Sem) VALUES(‘Suresh’, ‘4AD10CS005’,2)
INSERT INTO students (Stu_Name, USN, Sem) VALUES(‘Mahesh’, ‘4AD10CS004’,8)
INSERT INTO students (Stu_Name, USN, Sem) VALUES(‘Raveesh’, ‘4AD10CS020’,6)



## 2. PHP Script (with Selection Sort)

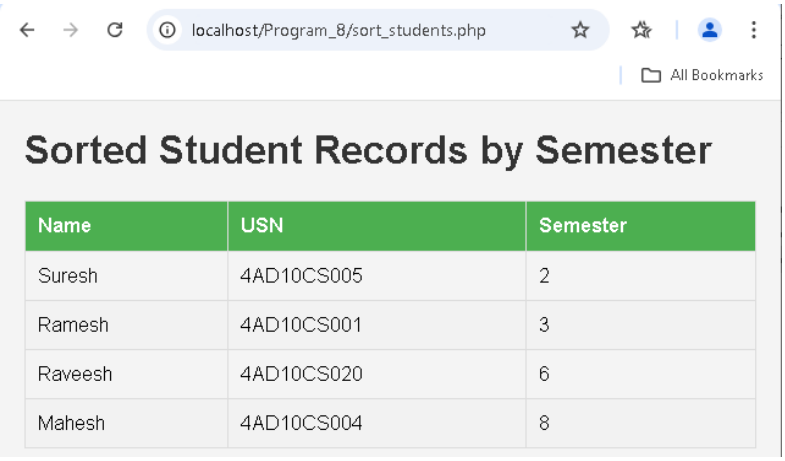
Create a file named `sort_students.php` with the following content:

```
<?php
// Database configuration
$host = 'localhost';
$dbname = 'web_lab_students';
$username = 'root'; // Replace with your MySQL username
$password = 'root'; // Replace with your MySQL password

// Create a new PDO instance
try {
   $pdo = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);
   $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
   die("Could not connect to the database: " . $e->getMessage());
}

// Fetch student records from the database
$sql = 'SELECT * FROM students';
$stmt = $pdo->query($sql);
$students = $stmt->fetchAll(PDO::FETCH_ASSOC);

// Function to perform selection sort
function selectionSort(&$array) {
   $n = count($array);
   for ($i = 0; $i < $n - 1; $i++) {
      $minIndex = $i;
      for ($j = $i + 1; $j < $n; $j++) {
         if ($array[$j]['Sem'] < $array[$minIndex]['Sem']) {
            $minIndex = $j;
         }
      }
      if ($minIndex != $i) {
         $temp = $array[$i];
         $array[$i] = $array[$minIndex];
         $array[$minIndex] = $temp;
      }
   }
}

// Sort students by semester using selection sort
selectionSort($students);
?>
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sorted Student Records</title>
    <style>
      body {
        font-family: Arial, sans-serif;
        margin: 20px;
        background-color: #f4f4f4;
      }
      table {
        width: 100%;
        border-collapse: collapse;
      }
      th, td {
        padding: 10px;
        border: 1px solid #ddd;
        text-align: left;
      }
      th {
        background-color: #4CAF50;
        color: white;
      }
      tr:nth-child(even) {
        background-color: #f2f2f2;
      }
      h1 {
        color: #333;
      }
    </style>
  </head>
  <body>
    <h1>Sorted Student Records by Semester</h1>
    <table>
      <thead>
        <tr>
          <th>Name</th>
          <th>USN</th>
          <th>Semester</th>
        </tr>
      </thead>
      <tbody>
        <?php foreach ($students as $student): ?>
          <tr>
            <td><?php echo htmlspecialchars($student['Stu_Name']); ?></td>
            <td><?php echo htmlspecialchars($student['USN']); ?></td>
            <td><?php echo htmlspecialchars($student['Sem']); ?></td>
          </tr>
        <?php endforeach; ?>
```

```
        </tbody>
      </table>
    </body>
    </html>
```

## Output




| Name | USN | Semester |
|---|---|---|
| Suresh | 4AD10CS005 | 2 |
| Ramesh | 4AD10CS001 | 3 |
| Raveesh | 4AD10CS020 | 6 |
| Mahesh | 4AD10CS004 | 8 |

## Explanation

1. **Database Connection**: Establish a connection to the MySQL database using PDO.
2. **Fetching Data**: Retrieve all student records from the `students` table.
3. **Selection Sort Function**: Implement the selection sort algorithm to sort students by their semester (`sem`).
4. **HTML and CSS**: Display the sorted student records in an HTML table, styled with basic CSS.

## 3. Running the Script

- Make sure your MySQL server is running, and PHP is properly configured.
- Place the `sort_students.php` file in your web server's root directory.
- Access the script via your browser, for example, `http://localhost/sort_students.php`.

This script will display student records sorted by the `sem` field using the selection sort algorithm. Adjust database credentials and table schema as needed.

**Program 9**

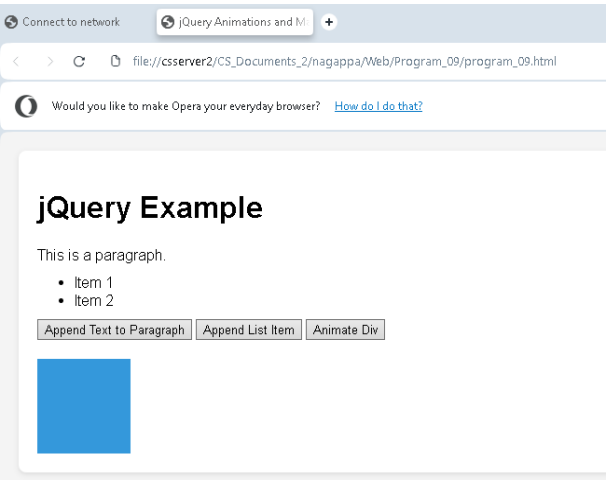
9. Develop jQuery script (with HTML/CSS) for:
   a. Appends the content at the end of the existing paragraph and list.
   b. Change the state of the element with CSS style using animate() method
   c. Change the color of any div that is animated.

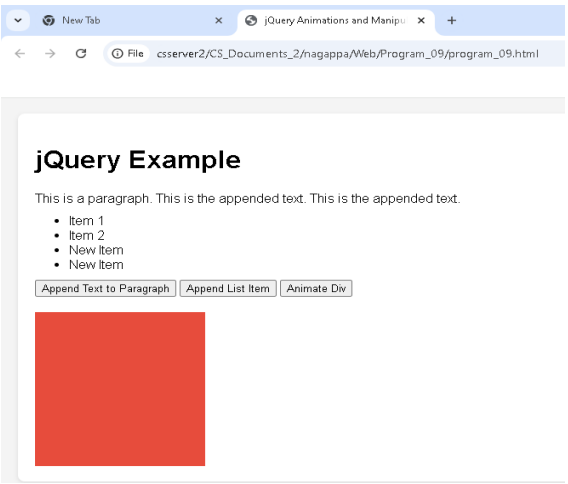
**Program**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>jQuery Animations and Manipulations</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 20px;
      background-color: #f4f4f4;
    }
    .container {
      background-color: #fff;
      padding: 20px;
      border-radius: 8px;
      box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    }
    p, ul {
      margin: 10px 0;
    }
    .animated-div {
      width: 100px;
      height: 100px;
      background-color: #3498db;
      margin-top: 20px;
      transition: background-color 0.5s;
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>jQuery Example</h1>

    <!-- Paragraph and List to append content to -->
    <p id="paragraph">This is a paragraph.</p>
    <ul id="list">
```

```
        <li>Item 1</li>
        <li>Item 2</li>
      </ul>

      <!-- Buttons for jQuery actions -->
      <button id="appendText">Append Text to Paragraph</button>
      <button id="appendListItem">Append List Item</button>
      <button id="animateDiv">Animate Div</button>

      <!-- Div to animate -->
      <div id="animatedDiv" class="animated-div"></div>
    </div>

    <!-- jQuery Library -->
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script>
      $(document).ready(function() {
        // Append content to paragraph and list
        $('#appendText').click(function() {
          $('#paragraph').append(' This is the appended text.');
        });

        $('#appendListItem').click(function() {
          $('#list').append('<li>New Item</li>');
        });

        // Animate div
        $('#animateDiv').click(function() {
          $('#animatedDiv').animate({
            width: '200px',
            height: '200px'
          }, 1000, function() {
            // Change color after animation completes
            $(this).css('background-color', '#e74c3c');
          });
        });
      });
    </script>
  </body>
  </html>
```

## Output




## Explanation:

1. **HTML**:
   - Contains a paragraph and a list to which content can be appended.
   - Includes buttons for appending text, appending list items, and animating a `div`.
   - The `div` with the `id="animatedDiv"` is styled and animated.

2. **CSS**:
   - Basic styling for the body, container, paragraph, and list.
   - The `.animated-div` class provides initial styling and a transition effect for background color changes.

3. **jQuery**:
   - **Append Content**:
     - `$('#appendText').click()`: Appends text to the existing paragraph when the button is clicked.
     - `$('#appendListItem').click()`: Appends a new list item to the existing list when the button is clicked.

   - **Animate Div**:
     - `$('#animateDiv').click()`: Animates the size of the `div` when the button is clicked. After the animation completes, the background color of the `div` changes to red.

This script demonstrates basic jQuery functionalities for manipulating and animating HTML elements. You can save the above code into an `index.html` file and open it in a web browser to see it in action.

**Program 10**

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.

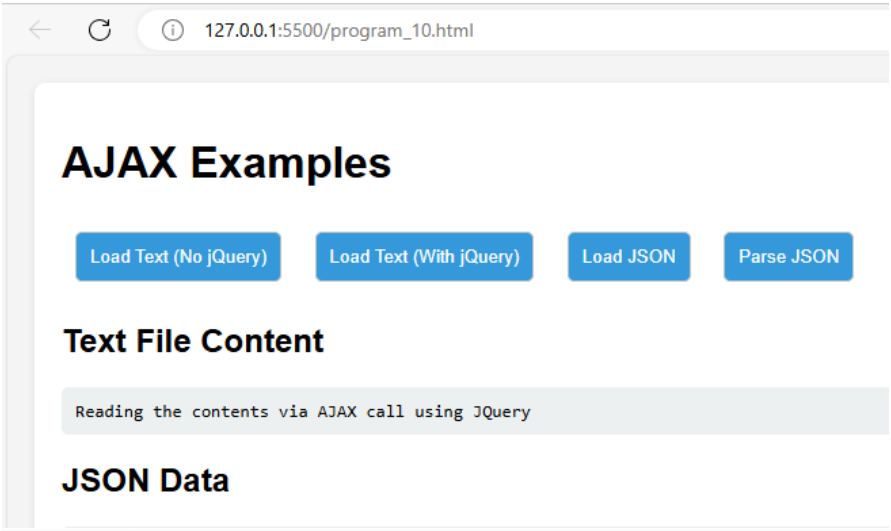
**Program**

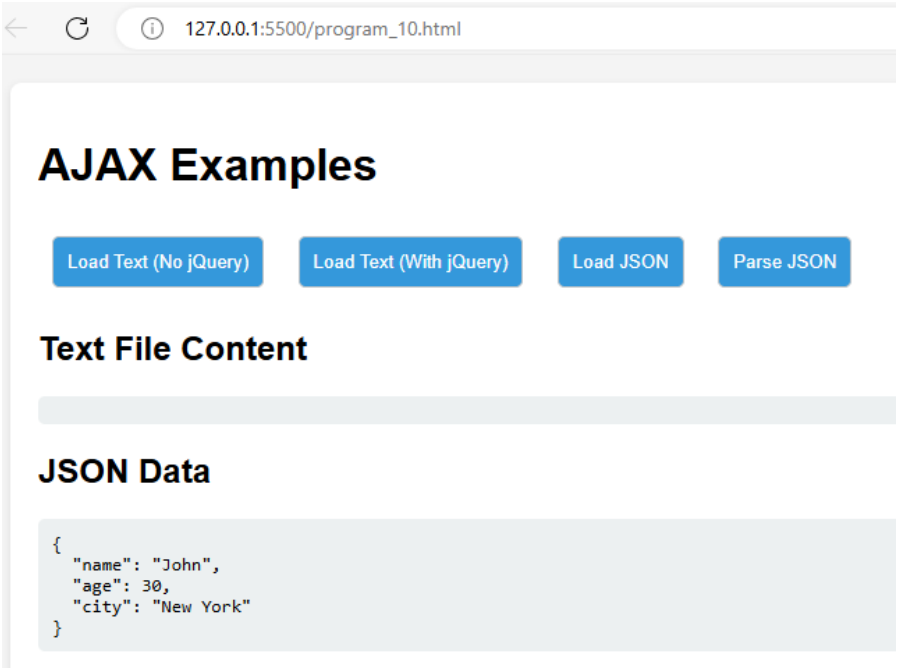
```
<!DOCTYPE html>
<html lang="en">
        <head>
          <meta charset="UTF-8">
          <meta name="viewport" content="width=device-width, initial-scale=1.0">
          <title>AJAX with and without jQuery</title>
          <style>
            body {
              font-family: Arial, sans-serif;
              margin: 20px;
              background-color: #f4f4f4;
            }
            .container {
              background-color: #fff;
              padding: 20px;
              border-radius: 8px;
              box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
            }
            button {
              margin: 10px;
              padding: 10px;
              border: 1px solid #bdc3c7;
              border-radius: 5px;
              background-color: #3498db;
              color: #fff;
              cursor: pointer;
            }
            button:hover {
              background-color: #2980b9;
            }
            pre {
              background-color: #ecf0f1;
              padding: 10px;
              border-radius: 5px;
              white-space: pre-wrap;
```

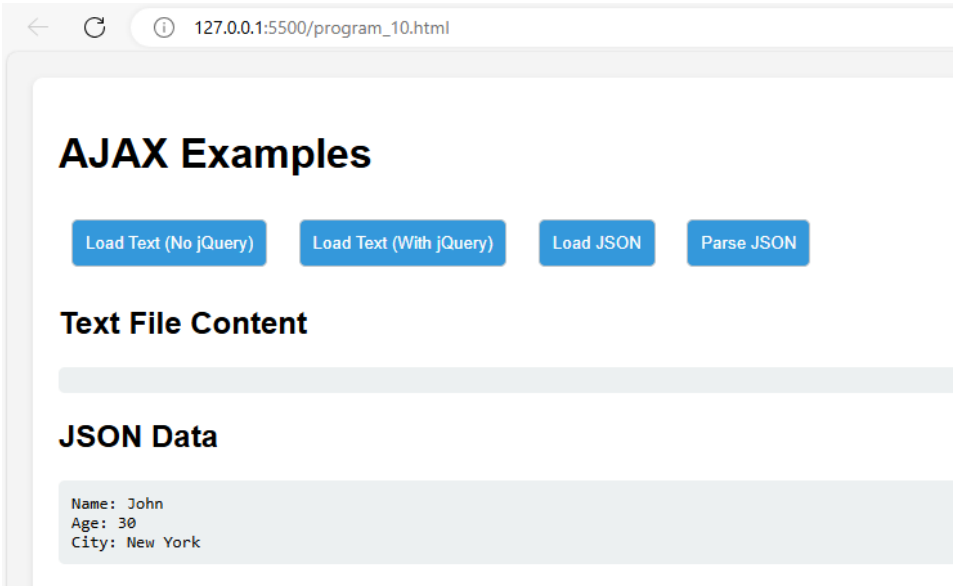
```
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>AJAX Examples</h1>

        <!-- Buttons for AJAX requests -->
        <button id="loadTextWithoutJQuery">Load Text (No jQuery)</button>
        <button id="loadTextWithJQuery">Load Text (With jQuery)</button>
        <button id="loadJson">Load JSON</button>
        <button id="parseJson">Parse JSON</button>

        <!-- Output areas -->
        <h2>Text File Content</h2>
        <pre id="textOutput"></pre>

        <h2>JSON Data</h2>
        <pre id="jsonOutput"></pre>
    </div>

    <!-- jQuery Library -->
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

    <!-- JavaScript -->
    <script>
        // Function to load text content using vanilla JavaScript
        document.getElementById('loadTextWithoutJQuery').addEventListener('click',
function() {
            var xhr = new XMLHttpRequest();
            xhr.open('GET', 'sample.txt', true);
            xhr.onreadystatechange = function() {
                if (xhr.readyState === 4 && xhr.status === 200) {
                    document.getElementById('textOutput').textContent = xhr.responseText;
                }
            };
            xhr.send();
        });

        // Function to load text content using jQuery
        $('#loadTextWithJQuery').click(function() {
            $.ajax({
                url: 'sample.txt',
                method: 'GET',
                success: function(data) {
                    $('#textOutput').text(data);
                },
                error: function() {
```

```
                $('#textOutput').text('Error loading text file.');
            }
        });
    });

    // Function to load JSON data using jQuery's getJSON method
    $('#loadJson').click(function() {
        $.getJSON('data.json', function(data) {
            $('#jsonOutput').text(JSON.stringify(data, null, 2));
        }).fail(function() {
            $('#jsonOutput').text('Error loading JSON file.');
        });
    });

    // Function to parse JSON data using jQuery's parseJSON method
    $('#parseJson').click(function() {
        var jsonString = '{"name": "John", "age": 30, "city": "New York"}';
        var jsonObject = $.parseJSON(jsonString);
        $('#jsonOutput').text('Name: ' + jsonObject.name + '\nAge: ' + jsonObject.age +
'\nCity: ' + jsonObject.city);
    });
  </script>
</body>
</html>
```

**Output**


127.0.0.1:5500/program_10.html
AJAX Examples
Load Text (No jQuery)
Load Text (With jQuery)
Load JSON
Parse JSON
Text File Content
Reading the contents via AJAX call using JQuery
JSON Data


127.0.0.1:5500/program_10.html
AJAX Examples
Load Text (No jQuery)
Load Text (With jQuery)
Load JSON
Parse JSON
Text File Content
JSON Data
{
"name": "John",
"age": 30,
"city": "New York"
}


127.0.0.1:5500/program_10.html
AJAX Examples
Load Text (No jQuery)
Load Text (With jQuery)
Load JSON
Parse JSON
Text File Content
JSON Data
Name: John
Age: 30
City: New York

**Explanation:**

1. **HTML**:
   - The HTML structure includes buttons for different AJAX operations and output areas to display results.
   - There are buttons for loading text and JSON data using both vanilla JavaScript and jQuery, as well as for parsing JSON.

2. **CSS**:
   - Basic styling is applied to the body, buttons, and output areas to make the page visually appealing.

3. **JavaScript**:
   - **Without jQuery**:
     - The `loadTextWithoutJQuery` button uses vanilla JavaScript's `XMLHttpRequest` to load and display text from a file (`sample.txt`).
   - **With jQuery**:
     - The `loadTextWithJQuery` button uses jQuery's `$.ajax()` method to load and display text from a file.
     - The `loadJson` button uses jQuery's `$.getJSON()` method to fetch and display JSON data from `data.json`.
     - The `parseJson` button demonstrates `$.parseJSON()` by parsing a JSON string and displaying its contents.

**Files Needed:**

1. **`sample.txt`**: A text file containing sample text. For example:
   **Content of sample.txt file**
   ```
   This is some sample text loaded from a text file.
   ```

2. **`data.json`**: A JSON file containing sample JSON data. For example:
   **Content of data.json file**
   ```
   {
       "name": "John",
       "age": 30,
       "city": "New York"
   }
   ```

You can save the provided HTML code into an `index.html` file and ensure that `sample.txt` and `data.json` are in the same directory as your HTML file. Open `index.html` in a web browser to see the examples in action.
Remember that AJAX requests need to be served from a web server or local server environment to work correctly (i.e., they might not work if just opened directly from the file system due to browser security restrictions). You can use tools like Live Server in VS Code or a simple local server like `http-server` in Node.js for testing.

# VIVA QUESTIONS

## HTML

1. **What is HTML, and what does it stand for?**
   - HTML stands for HyperText Markup Language. It is used to structure content on the web.

2. **What is the basic structure of an HTML document?**
   - The basic structure includes <!DOCTYPE html>, <html>, <head>, and <body> tags.

3. **What is the purpose of the <head> section in an HTML document?**
   - It contains meta-information, links to stylesheets, scripts, and the title of the document.

4. **What is the purpose of the <title> tag?**
   - The <title> tag sets the title of the webpage, which appears in the browser tab.

5. **What is an HTML element?**
   - An HTML element is a component of an HTML document, consisting of an opening tag, content, and a closing tag.

6. **What is an attribute in HTML?**
   - An attribute provides additional information about an HTML element and is included in the opening tag. It usually consists of a name and a value (e.g., src="image.jpg").

7. **What is the difference between block-level and inline elements?**
   - Block-level elements (e.g., , <h1>) take up the full width available and start on a new line. Inline elements (e.g., , <a>) only take up as much width as necessary and do not start on a new line.

8. **How do you create a hyperlink in HTML?**
   - Use the <a> tag with the href attribute to specify the URL. Example: <a href="https://example.com">Visit Example</a>.

9. **What does the target="_blank" attribute do in a hyperlink?**
   - It opens the linked document in a new tab or window.

10. **How do you include an image in an HTML document?**
    - Use the <img> tag with the src attribute to specify the image source and the alt attribute for alternative text. Example: <img src="image.jpg" alt="Description">.

11. **What is the purpose of the alt attribute in the <img> tag?**
    - The alt attribute provides alternative text for the image if it cannot be displayed. It also improves accessibility.

**12. What is an unordered list, and how is it created?**

- An unordered list is a list where items are not numbered. It is created using the <ul> tag, with each item marked by <li> tags. Example:
```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
</ul>
```

**13. How do you create an ordered list in HTML?**

- Use the <ol> tag to create an ordered list, with each item marked by <li> tags. Example:
```
<ol>
  <li>First item</li>
  <li>Second item</li>
</ol>
```

**14. What is the purpose of the <form> tag in HTML?**

- The <form> tag is used to create a form for collecting user input.

**15. How do you create a text input field in a form?**

- Use the <input> tag with the type="text" attribute. Example: <input type="text" name="username">.

**16. What is the <textarea> element used for?**

- The <textarea> element is used to create a multi-line text input field.

**17. How do you create a dropdown menu in HTML?**

- Use the <select> tag with <option> tags for each dropdown item. Example:
```
<select name="options">
  <option value="1">Option 1</option>
  <option value="2">Option 2</option>
</select>
```

**18. What is the <button> element used for?**

- The <button> element is used to create a clickable button. It can be used within forms or as standalone buttons.

**19. How do you create a table in HTML?**

- Use the <table> tag with <tr> for rows, <th> for header cells, and <td> for data cells. Example:
```
<table>
  <tr>
    <th>Header 1</th>
    <th>Header 2</th>
  </tr>
  <tr>
    <td>Data 1</td>
    <td>Data 2</td>
  </tr>
</table>
```

**20. What is the <caption> tag used for in a table?**

- The <caption> tag provides a title or description for the table.

**21. What is the purpose of the <meta> tag?**

- The <meta> tag provides metadata about the HTML document, such as character encoding and viewport settings.

**22. How do you specify the character encoding of an HTML document?**

- Use the <meta charset="UTF-8"> tag in the <head> section.

**23. What is the <link> tag used for?**

- The <link> tag is used to link external resources, such as stylesheets, to the HTML document.

**24. How do you include an external CSS file in an HTML document?**

- Use the <link> tag with rel="stylesheet" and href attributes. Example: <link rel="stylesheet" href="styles.css">.

**25. What is the <script> tag used for in HTML?**

- The <script> tag is used to include JavaScript code or link to external JavaScript files.

**26. How do you add comments in HTML?**

- Use the <!-- comment --> syntax to add comments.

**27. What is the  element used for?**

- The  element is used as a container to group and style sections of a webpage.

**28. What is the  element used for?**

- The  element is used to group and style inline portions of text or other elements.

**29. What are semantic HTML elements?**

- Semantic HTML elements clearly describe their content and purpose, such as <header>, <footer>, <article>, and <section>.

**30. How do you create a line break in HTML?**

- Use the <br> tag to insert a line break.

**31. What is the difference between the  and <b> tags?**

- The  tag indicates that text is of strong importance, while <b> simply applies bold styling without implying importance.

**32. What does the class attribute do in HTML?**

- The class attribute assigns one or more class names to an element, which can be used for styling with CSS or targeting with JavaScript.

**33. What does the id attribute do in HTML?**

- The id attribute assigns a unique identifier to an element, which can be used for styling, JavaScript targeting, or linking.

**34. How do you create a checkbox in HTML?**

- Use the <input> tag with the type="checkbox" attribute. Example: <input type="checkbox" name="accept" value="yes">.

**35. How do you create a radio button in HTML?**

- Use the <input> tag with the type="radio" attribute. Example: <input type="radio" name="gender" value="male"> Male.

**36. What is the <iframe> tag used for?**

- The <iframe> tag is used to embed another HTML document within the current document.

**37. How do you make a form submit to a specific URL?**

- Use the action attribute of the <form> tag to specify the URL where the form data will be sent. Example: <form action="submit.php">.

**38. What does the method attribute do in a form?**

- The method attribute specifies how the form data should be sent to the server. Common values are GET and POST.

**39. How do you make text italic in HTML?**

- Use the <i> or  tags to make text italic.  is semantically preferred as it indicates emphasis.

**40. What is the data-* attribute used for in HTML?**

- The data-* attribute is used to store custom data that can be accessed using JavaScript. It allows embedding of custom information into HTML elements.

## Stylesheets – CSS

1. **What is CSS, and what does it stand for?**
   - CSS stands for Cascading Style Sheets. It is used to control the presentation, layout, and design of HTML elements on a webpage.

2. **How can CSS be included in an HTML document?**
   - CSS can be included in three ways:
     - **Inline CSS**: Using the style attribute within HTML elements (e.g., <p style="color: red;">).
     - **Internal CSS**: Using a <style> tag within the <head> section of the HTML document.
     - **External CSS**: Linking to a separate CSS file using a <link> tag in the <head> section.

3. **What is the syntax for a CSS rule?**
   - A CSS rule consists of a selector and a declaration block. Example:
     ```
     selector {
       property: value;
     }
     ```
     For example:
     ```
     p {
       color: blue;
     }
     ```

4. **What is the purpose of the class selector in CSS?**
   - The class selector is used to apply styles to elements with a specific class attribute. It is defined with a dot before the class name (e.g., .my-class).

5. **How does the id selector differ from the class selector in CSS?**
   - The id selector targets a single unique element with a specific id attribute, while the class selector can target multiple elements with the same class. The id selector is defined with a hash symbol (e.g., #my-id), and the class selector is defined with a dot (e.g., .my-class).

6. **What is specificity in CSS?**
   - Specificity determines which CSS rule is applied when multiple rules match the same element. It is calculated based on the types of selectors used (e.g., inline styles, IDs, classes, and elements).

7. **What is the !important declaration in CSS?**
   - The !important declaration is used to give a CSS rule higher priority and override other rules. It should be used sparingly. Example: color: red !important;.

8. **What are pseudo-classes in CSS?**
   - Pseudo-classes are keywords added to selectors that specify a special state of the selected elements. Examples include :hover, :focus, and :nth-child.

9. **What are pseudo-elements in CSS?**
   - Pseudo-elements are used to style specific parts of an element. Examples include ::before, ::after, and ::first-line.

10. **What is the box model in CSS?**
    - The box model describes the rectangular boxes generated for elements in the document tree. It includes content, padding, border, and margin areas.

11. **How do you center a block-level element horizontally using CSS?**
    - Set the margin property to auto and specify a width for the element. Example:
      ```
      .centered {
        width: 50%;
        margin: 0 auto;
      }
      ```

**12. What is the difference between padding and margin in CSS?**

- Padding is the space between the content of an element and its border. Margin is the space outside the border, separating the element from other elements.

**13. How do you apply a style to multiple elements using CSS?**

- List the elements separated by commas in the selector. Example:

```
h1, h2, h3 {
  color: green;
}
```

**14. What is the display property in CSS, and what are its common values?**

- The display property specifies how an element is displayed on the page. Common values include block, inline, inline-block, none, and flex.

**15. How do you create a responsive design using CSS?**

- Use media queries to apply different styles for different screen sizes and devices. Example:

```
@media (max-width: 600px) {
  body {
    background-color: lightblue;
  }
}
```

**16. What is Flexbox in CSS?**

- Flexbox is a layout model that allows you to design complex layouts with flexible and responsive arrangements. It uses properties like display: flex, justify-content, align-items, and flex-direction.

**17. What is CSS Grid Layout?**

- CSS Grid Layout is a two-dimensional layout system for the web that allows you to create complex grid-based designs with rows and columns. It uses properties like display: grid, grid-template-rows, and grid-template-columns.

**18. What does the float property do in CSS?**

- The float property is used to position an element to the left or right within its containing block, allowing other content to wrap around it.

**19. What is the position property in CSS, and what are its values?**

- The position property specifies how an element is positioned in the document. Common values include static, relative, absolute, fixed, and sticky.

**20. How do you use the z-index property in CSS?**

- The z-index property specifies the stack order of elements. Elements with a higher z-index value are displayed in front of elements with a lower value. It only works on positioned elements (position set to relative, absolute, fixed, or sticky).

**21. What is the opacity property in CSS?**

- The opacity property sets the transparency level of an element, with values ranging from 0 (completely transparent) to 1 (completely opaque).

**22. How do you change the font size of an element using CSS?**

- Use the font-size property. Example: font-size: 16px;.

**23. What is the background shorthand property in CSS?**

- The background shorthand property allows you to set multiple background properties in one declaration, such as background-color, background-image, background-repeat, and background-position.

**24. What is the purpose of the border-radius property in CSS?**

- The border-radius property is used to create rounded corners on elements.

**25. How do you apply a gradient background using CSS?**

- Use the background-image property with a linear-gradient or radial-gradient function. Example:
  background-image: linear-gradient(to right, red, yellow);

**26. What are media queries in CSS?**

- Media queries are used to apply different styles based on the characteristics of the device, such as screen width, height, or orientation.

**27. How do you set the width of an element to 100% of its parent container?**

- Use width: 100%; in the CSS rule for the element.

**28. What is the text-align property used for?**

- The text-align property is used to set the horizontal alignment of text within an element. Common values include left, right, center, and justify.

**29. What is the line-height property in CSS?**

- The line-height property specifies the amount of space between lines of text.

**30. How do you remove the default underline from links using CSS?**

- Use the text-decoration property with the value none. Example: a { text-decoration: none; }.

**31. What is the box-shadow property used for?**

- The box-shadow property adds shadow effects around an element's frame. Example:
  box-shadow: 2px 2px 5px grey;

**32. How do you change the font family of an element using CSS?**

- Use the font-family property. Example: font-family: Arial, sans-serif;.

**33. What is the text-transform property in CSS?**

- The text-transform property is used to control the capitalization of text. Values include uppercase, lowercase, and capitalize.

**34. How do you use the :hover pseudo-class in CSS?**

- The :hover pseudo-class applies styles when the user hovers over an element with their mouse. Example:

```
a:hover {
  color: red;
}
```

**35. What is the cursor property in CSS?**

- The cursor property specifies the type of cursor to be displayed when hovering over an element. Common values include pointer, default, and wait.

**36. How do you apply different styles to even and odd table rows?**

- Use the :nth-child pseudo-class with the even or odd keyword. Example:

```
tr:nth-child(even) {
  background-color: #f2f2f2;
}
```

**37. What is the visibility property in CSS, and how does it differ from display?**

- The visibility property controls whether an element is visible or not, with values visible and hidden. Unlike display: none, visibility: hidden keeps the element in the layout but hides it

## PHP

1. **What is PHP?**
   - PHP (Hypertext Preprocessor) is a widely-used, open-source server-side scripting language designed primarily for web development. It can be embedded into HTML and interacts with databases to generate dynamic content.

2. **How does PHP handle form data?**
   - PHP handles form data through the $_GET and $_POST superglobals. $_GET is used for retrieving data sent via HTTP GET method, while $_POST is used for data sent via HTTP POST method.

3. **What are the different types of variables in PHP?**
   - PHP supports several types of variables including:
     - **Integers**: Whole numbers.
     - **Floats**: Decimal numbers.
     - **Strings**: Textual data.
     - **Booleans**: true or false.
     - **Arrays**: Collections of values.
     - **Objects**: Instances of classes.
     - **NULL**: Represents a variable with no value.

4. **What is the difference between == and === in PHP?**
   - == is the equality operator that checks if two values are equal after type juggling (type conversion). === is the identity operator that checks if two values are equal and of the same type.

5. **How do you define a function in PHP?**
   - A function in PHP is defined using the function keyword.
     ```
     function myFunction($param) {
        // function body
     }
     ```

6. **What is the purpose of the include and require statements in PHP?**
   - Both include and require are used to include the content of one PHP file into another. The main difference is that require will produce a fatal error and stop script execution if the file is not found, whereas include will only produce a warning and continue execution.

7. **What are PHP superglobals?**
   - Superglobals are built-in global arrays in PHP that are always accessible, regardless of scope. Examples include:
     - $_GET, $_POST, $_REQUEST, $_SESSION, $_COOKIE, $_FILES, $_SERVER, $_ENV, $_GLOBALS.

8. **How do you connect to a MySQL database in PHP?**
   - You can use the mysqli or PDO extension to connect to a MySQL database. Example using mysqli:
     ```
     $conn = new mysqli("hostname", "username", "password", "database");
     if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
     }
     ```

9. **How do you execute a SQL query in PHP?**
   - To execute a SQL query, you use the query method of the mysqli object or exec method of the PDO object.
     ```
     $sql = "SELECT * FROM table";
     $result = $conn->query($sql);

     if ($result->num_rows > 0) {
        while($row = $result->fetch_assoc()) {
           // Process each row
        }
     }
     ```

10. **How do you handle errors in PHP?**
    - PHP provides various methods for error handling, including:
      - **Error reporting functions**: error_reporting(), set_error_handler().
      - **Exception handling**: Using try, catch, and finally blocks to handle exceptions.
      ```
      try {
         // Code that may throw an exception
      } catch (Exception $e) {
         echo 'Caught exception: ',  $e->getMessage(), "\n";
      }
      ```

**11. What is SQL injection and how can you prevent it?**

- SQL injection is a security vulnerability where an attacker can manipulate SQL queries by injecting malicious input. To prevent it, use prepared statements and parameterized queries:

```
$stmt = $conn->prepare("SELECT * FROM users WHERE username = ?");
$stmt->bind_param("s", $username);
$stmt->execute();
```

**12. How do you manage sessions in PHP?**

- Sessions are managed using the $_SESSION superglobal. Start a session with session_start() and use $_SESSION to store session data.

```
session_start();
$_SESSION['username'] = 'JohnDoe';
```

**13. How do you upload a file in PHP?**

- File uploads are handled through the $_FILES superglobal. You use the move_uploaded_file() function to move the uploaded file from the temporary directory to a desired location.

```
if (isset($_FILES['file'])) {
   $file = $_FILES['file'];
   $destination = 'uploads/' . $file['name'];
   move_uploaded_file($file['tmp_name'], $destination);
}
```

**14. How do you read and write files in PHP?**

- Use functions like fopen(), fread(), fwrite(), and fclose() to handle file operations.

```
$file = fopen("example.txt", "r");
$content = fread($file, filesize("example.txt"));
fclose($file);

$file = fopen("example.txt", "w");
fwrite($file, "New content");
fclose($file);
```

## JavaScript

1. **What is JavaScript and what is it used for?**
   - JavaScript is a high-level, interpreted programming language primarily used to create dynamic and interactive content on websites. It allows for client-side scripting to enhance user experiences.

2. **Explain the difference between `var`, `let`, and `const`.**
   - var is function-scoped and can be redeclared. let is block-scoped and cannot be redeclared within the same block. const is also block-scoped but is used for variables that should not be reassigned after initialization.

3. **What are data types in JavaScript?**
   - JavaScript has primitive data types like number, string, boolean, undefined, null, symbol, and bigint. There are also non-primitive types like object and array.

4. **What is the difference between == and ===?**
   - == (loose equality) compares values after performing type coercion if necessary, while === (strict equality) compares both value and type without type coercion.

5. **How does prototypal inheritance work in JavaScript?**
   - JavaScript uses prototypes for inheritance. Each object has a prototype object from which it can inherit properties and methods. This is achieved through the prototype chain.

6. **What is the this keyword in JavaScript?**
   - this refers to the context in which a function is executed. It can vary depending on how the function is called: it refers to the global object in non-strict mode, the object that is calling the function in method calls, or it can be explicitly bound using call, apply, or bind.

7. **What are JavaScript prototypes and how do you use them?**
   - Prototypes are a mechanism by which JavaScript objects inherit features from one another. You can set the prototype of an object using Object.create() or by setting the prototype property of a constructor function.

8. **What are callbacks in JavaScript?**
   - Callbacks are functions passed as arguments to other functions and are executed after the completion of the other function. They are used to handle asynchronous operations.

9. **Explain promises and how they are used.**
   - Promises represent the eventual completion (or failure) of an asynchronous operation and its resulting value. They provide a cleaner way to handle asynchronous code compared to callbacks, using methods like then(), catch(), and finally().

**10. What is async/await and how does it improve asynchronous code?**

- async and await are syntactic sugar built on top of promises, making asynchronous code look and behave more like synchronous code. async functions return a promise, and await pauses the function execution until the promise is resolved.

**11. What are JavaScript modules and why are they used?**

- JavaScript modules are a way to organize code by exporting and importing functions, objects, or primitives from one file to another. This improves code maintainability and encapsulation. Modules are typically managed using the import and export keywords.

**12. Explain event delegation in JavaScript.**

- Event delegation is a technique where a single event listener is added to a parent element to manage events for its child elements. This leverages event bubbling and can be more efficient than adding multiple listeners to individual elements.

**13. What are the different ways to handle errors in JavaScript?**

- Errors can be handled using try, catch, finally, and throw. try contains the code that might throw an error, catch handles the error, and finally executes code regardless of the result. throw is used to manually trigger an error.

## AJAX

1. **What is AJAX and what are its primary components?**
   - AJAX stands for Asynchronous JavaScript and XML. It allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. Its primary components include JavaScript, the XMLHttpRequest object, and often XML or JSON for data interchange.

2. **How does AJAX improve user experience on web pages?**
   - AJAX improves user experience by enabling web pages to update content dynamically without needing a full page reload. This results in a smoother, faster, and more responsive user experience.

3. **Can you explain the process of making an AJAX request?**
   - Making an AJAX request typically involves creating an instance of the XMLHttpRequest object, setting up the request (specifying the HTTP method, URL, and whether the request should be asynchronous), sending the request, and handling the server's response in a callback function.

4. **What are the key methods and properties of the XMLHttpRequest object?**
   - Key methods include `open()`, `send()`, `setRequestHeader()`, and `abort()`. Key properties include `readyState`, `status`, and `responseText` or `responseXML`.

5. **What is the purpose of the `readyState` property in XMLHttpRequest?**
   - The `readyState` property indicates the current state of the XMLHttpRequest object. It can have values from 0 (uninitialized) to 4 (complete), which correspond to different stages of the request lifecycle.

6. **How can you handle errors in an AJAX request?**
   - Errors can be handled by checking the `status` property of the XMLHttpRequest object in the `onreadystatechange` callback function. You can also use the `onerror` event handler to catch network-level errors.

7. **What is JSON, and how is it used with AJAX?**
   - JSON (JavaScript Object Notation) is a lightweight data interchange format that is easy for humans to read and write and easy for machines to parse and generate. In AJAX, JSON is often used as the format for sending and receiving data between the client and server due to its simplicity and ease of use with JavaScript.

8. **Explain the difference between synchronous and asynchronous requests in AJAX.**
   - In synchronous requests, the browser waits for the server response before continuing with the execution of the script, which can lead to a frozen user interface. In asynchronous requests, the browser can continue to process other tasks while waiting for the server response, providing a more fluid user experience.

9. **What are some common uses of AJAX in web development?**
   - Common uses include form submission without reloading the page, loading new content dynamically (e.g., infinite scroll), updating specific parts of a page (e.g., live search suggestions), and interacting with APIs to fetch or send data.

10. **How does AJAX differ from traditional full-page form submission?**
    - Traditional form submission involves sending the entire form data to the server and refreshing the page with the server's response. AJAX allows for sending and receiving data asynchronously without reloading the page, which enables partial updates and a more seamless user experience.

## JQUERY

1. **What is jQuery, and why is it used in web development?**
   - jQuery is a fast, lightweight, and feature-rich JavaScript library designed to simplify HTML document traversal, event handling, animation, and Ajax interactions. It is used to streamline and simplify tasks that would otherwise require more verbose JavaScript code.

2. **How do you include jQuery in a web page?**
   - jQuery can be included in a web page by linking to a jQuery CDN (Content Delivery Network) or by downloading and hosting the jQuery file locally. The

typical inclusion in HTML is done with a `<script>` tag, either referencing the jQuery library from a CDN or from a local file:

```
<script      src="https://code.jquery.com/jquery-3.6.0.min.js">
</script>
```

or

```
<script src="path/to/your/local/jquery.min.js"></script>
```

3. **What is the purpose of the `$` function in jQuery?**
   - The `$` function is a shorthand for the `jQuery` function. It is used to select elements, create new jQuery objects, and call jQuery methods.
     For example, `$('#myElement')` selects an element with the ID `myElement`.

4. **How do you handle events in jQuery?**
   - Events in jQuery are handled using the `.on()` method or its shorthand `.click()`, `.hover()`, etc.
     For example:

```
$('#myButton').on('click', function() {
  alert('Button clicked!');
});
```

or

```
$('#myButton').click(function() {
  alert('Button clicked!');
});
```

5. **What is the difference between `.hide()` and `.fadeOut()` in jQuery?**
   - `.hide()` immediately hides the selected elements by setting their `display` property to `none`. `.fadeOut()` gradually hides the elements by animating the opacity from visible to hidden, creating a fading effect.

6. **How can you make an AJAX request using jQuery?**
   - jQuery provides several methods for making AJAX requests, such as `.ajax()`, `.get()`, and `.post()`.
     For example:

```
$.ajax({
  url: 'example.com/data',
  method: 'GET',
  success: function(response) {
    console.log(response);
  },
  error: function(error) {
    console.log('Error:', error);
  }
});
```

7. **What is chaining in jQuery, and how does it work?**
   - Chaining in jQuery allows you to perform multiple operations on the same jQuery object in a single line of code. It works by returning the jQuery object itself from a method, enabling subsequent methods to be called. For example:

```
$('#myElement').css('color',
'red').slideUp(2000).slideDown(2000);
```

8. **How do you traverse the DOM with jQuery?**
   - jQuery provides various methods for traversing the DOM, such as `.parent()`, `.children()`, `.find()`, `.siblings()`, and `.prev()`.

For example:

```
$('#myElement').parent(); // Selects the parent element
$('#myElement').find('.child'); // Selects child elements with
class 'child'
```

**9. What is the purpose of the .data() method in jQuery?**

- The `.data()` method is used to store and retrieve data associated with DOM elements. It provides a way to attach arbitrary data to elements that can be accessed later.
  For example:

```
$('#myElement').data('key', 'value'); // Store data
var value = $('#myElement').data('key'); // Retrieve data
```

**10. How do you handle form submission with jQuery?**

- You can handle form submission using the `.submit()` event handler. You can prevent the default form submission behavior and perform actions like AJAX requests:

```
$('#myForm').on('submit', function(event) {
  event.preventDefault(); // Prevent default form submission
  // Perform AJAX request or other actions
});
```