1.Introduction to CSS

A. Create a simple HTML page and demonstrate the is of all three CSS implementation methods (Inline, internal, and external). Write a brief explanation of when each method is most appropriate.

HTML Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Implementation Methods</title>
    <style>
        /* Internal CSS */
        body {
            font-family: Arial, sans-serif;
            background-color: #f0f0f0;
        }
        h1 {
            color: #333;
        }
        .internal-style {
            color: blue;
        }
    </style>
    <link rel="stylesheet" href="styles.css"> <!-- External CSS -->
</head>
<body>
    <h1>CSS Implementation Methods</h1>
    <p class="internal-style">This paragraph uses internal CSS for its styling.</p>
```

```
    <p style="color: red;">This paragraph uses inline CSS for its styling.</p>
    <p class="external-style">This paragraph uses external CSS for its styling.</p>
</body>
</html>
```

External CSS (styles.css)

```
/* External CSS */
.external-style {
    color: green;
    font-weight: bold;
}
```

B. Take an existing HTML-only webpage and improve its appearance using only CSS. Write a short report on the advantages you observed.

HTML Webpage Before CSS Improvements

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simple Webpage</title>
</head>
<body>
    <h1>Welcome to My Website</h1>
    <p>This is a simple webpage.</p>
    <ul>
        <li>Feature 1</li>
        <li>Feature 2</li>
        <li>Feature 3</li>
    </ul>
```

```
    <p>Contact us at: info@example.com</p>
</body>
</html>
```

Improved Webpage with CSS

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simple Webpage</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 20px;
            color: #333;
        }
        h1 {
            text-align: center;
            color: #4CAF50;
            margin-bottom: 20px;
        }
        p {
            line-height: 1.6;
            margin: 10px 0;
        }
        ul {
            list-style-type: square;
            padding-left: 20px;
        }
```

```
        li {
            margin: 8px 0;
        }
        footer {
            text-align: center;
            margin-top: 20px;
            font-size: 0.9em;
            color: #666;
        }
    </style>
</head>
<body>
    <h1>Welcome to My Website</h1>
    <p>This is a simple webpage.</p>
    <ul>
        <li>Feature 1</li>
        <li>Feature 2</li>
        <li>Feature 3</li>
    </ul>
    <p>Contact us at: info@example.com</p>
    <footer>
        &copy; 2024 My Website
    </footer>
</body>
</html>
```

Report on Observed Advantages

Improved Readability:

The chosen font-family (Arial) and line height enhance readability, making the text easier to digest.

Enhanced Visual Appeal:

A subtle background color (#f4f4f4) provides contrast, making the content stand out against the background.

The heading color (#4CAF50) adds a visually appealing accent, drawing attention to the main title.

Better Structure:

Cantering the heading and applying margin spacing improves the overall layout and visual structure of the page.

Clear spacing between paragraphs and list items creates a less cluttered appearance.

Responsive Design:

Using percentage-based padding and flexible font sizes allows the layout to adjust to different screen sizes, improving usability on mobile devices.

Footer Addition:

Adding a footer provides additional information (like copyright) in a dedicated space, which enhances professionalism.

C. Research and create a timeline of CSS versions, noting key features introduced in each version.

Here's a timeline of CSS versions highlighting key features introduced in each:

CSS Timeline

1. CSS1 (1996)

Release Date: December 1996

Key Features:

Introduction of basic styling capabilities for text, colors, and backgrounds.

Support for simple layout properties (e.g., margins, padding).

Basic selectors (element, class, ID).

The concept of cascading styles and inheritance.

2. CSS2 (1998)

Release Date: May 1998

Key Features:

Added positioning (relative, absolute, fixed).

Media types for different devices (screen, print).

Improved selectors (attribute selectors, pseudo-classes).

Introduction of z-index for controlling stacking order.

Support for tables and advanced layout techniques.

3. CSS2.1 (2011)

Release Date: January 2011

Key Features:

A refinement of CSS2, addressing inconsistencies and clarifying specifications.

Improved support for internationalization and layout.

Minor updates to selectors and properties to enhance compatibility.

Not a new version but a recommendation to fix issues in CSS2.

4. CSS3 (2011 - Ongoing)

Release Date: 2011 (first modules recommended)

Key Features:

Modular approach, allowing independent development of features.

New selectors (nth-child, attribute selectors).

Introduction of media queries for responsive design.

Advanced layout features: Flexbox, Grid Layout.

Visual effects: transitions, animations, and transforms.

Backgrounds and borders enhancements (multiple backgrounds, border-radius).

Custom properties (CSS variables).

New color formats (RGBA, HSLA).

5. CSS4 (Proposed)

Current Status: Still under development (not finalized)

Key Features:

Ongoing discussions around enhancements and new features.

Enhanced selectors and pseudo-classes.

Improved styling capabilities for components (such as custom properties).

Features are being introduced incrementally, with no singular "CSS4" specification as a whole.

Summary

CSS1 laid the groundwork for styling web pages.

CSS2 expanded capabilities for layout and device targeting.

CSS2.1 refined and clarified earlier specifications.

CSS3 introduced significant advancements like media queries, new selectors, and enhanced layout systems, fostering responsive design.

CSS4 is still evolving, focusing on further enhancements and modularity.

This timeline illustrates the progressive nature of CSS development, showcasing how each version has built upon its predecessors to improve web design and development practices.

D. Design a simple webpage and create three different stylesheet for it. demonstrating how CSS can dramatically change the look of a page without altering the html.

HTML Webpage

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Simple Webpage</title>
   <link rel="stylesheet" type="text/css" href="style1.css" id="themeStylesheet">
</head>
<body>
   <header>
      <h1>Welcome to My Website</h1>
   </header>
```

```
    <main>
        <p>This is a simple webpage demonstrating different styles using CSS.</p>
        <ul>
            <li>Feature 1</li>
            <li>Feature 2</li>
            <li>Feature 3</li>
        </ul>
    </main>
    <footer>
        <p>Contact us at: info@example.com</p>
    </footer>

    <button onclick="changeStyle('style1.css')">Light Theme</button>
    <button onclick="changeStyle('style2.css')">Dark Theme</button>
    <button onclick="changeStyle('style3.css')">Colorful Theme</button>

    <script>
        function changeStyle(sheet) {
            document.getElementById('themeStylesheet').setAttribute('href', sheet);
        }
    </script>
</body>
</html>

Stylesheet 1: style1.css (Light Theme)
body {
    font-family: Arial, sans-serif;
    background-color: #ffffff;
    color: #333;
    margin: 0;
    padding: 20px;
}
```

```
header {
    text-align: center;
    background-color: #4CAF50;
    color: white;
    padding: 10px;
}

ul {
    list-style-type: disc;
    padding-left: 20px;
}
```

Stylesheet 2: style2.css (Dark Theme)

```
body {
    font-family: Arial, sans-serif;
    background-color: #333;
    color: #f4f4f4;
    margin: 0;
    padding: 20px;
}

header {
    text-align: center;
    background-color: #555;
    color: #fff;
    padding: 10px;
}
```

```
ul {
    list-style-type: square;
    padding-left: 20px;
}
```

Stylesheet 3: style3.css (Colorful Theme)

```
body {
    font-family: 'Courier New', monospace;
    background-color: #f9e79f;
    color: #2c3e50;
    margin: 0;
    padding: 20px;
}

header {
    text-align: center;
    background-color: #e67e22;
    color: white;
    padding: 15px;
}

ul {
    list-style-type: circle;
    padding-left: 20px;
}
```

2. PROPERTIES

A. Create a webpage that showcases various text and font properties. Include examples of different font families, sizes, weights, and styles.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Text and Font Properties</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
            background-color: #f4f4f4;
            color: #333;
        }
        h1 {
            text-align: center;
            color: #4CAF50;
        }
        .example {
            margin: 20px 0;
            padding: 10px;
            background-color: #fff;
            border: 1px solid #ddd;
            border-radius: 5px;
        }
        .font-family {
            font-family: 'Courier New', monospace;
        }
```

```
        .font-size {
            font-size: 24px; /* Large */
        }
        .font-weight {
            font-weight: bold; /* Bold */
        }
        .font-style {
            font-style: italic; /* Italic */
        }
        .text-decoration {
            text-decoration: underline; /* Underlined */
        }
        .text-transform {
            text-transform: uppercase; /* Uppercase */
        }
    </style>
</head>
<body>

    <h1>Text and Font Properties Showcase</h1>

    <div class="example">
        <h2>Font Family</h2>
        <p class="font-family">This text uses the <code>Courier New</code> font family.</p>
    </div>

    <div class="example">
        <h2>Font Size</h2>
        <p class="font-size">This text is larger (24px).</p>
    </div>

    <div class="example">
```

```
    <h2>Font Weight</h2>
    <p class="font-weight">This text is bold.</p>
  </div>

  <div class="example">
    <h2>Font Style</h2>
    <p class="font-style">This text is italic.</p>
  </div>

  <div class="example">
    <h2>Text Decoration</h2>
    <p class="text-decoration">This text is underlined.</p>
  </div>

  <div class="example">
    <h2>Text Transform</h2>
    <p class="text-transform">This text is uppercase.</p>
  </div>

</body>
</html>
```

B. Design a colorful webpage that demonstrates the use of different color formats, (named colors, hex codes, RGB, and HSL) for both text and background colors.

HTML Structure

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

```
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <h1>Color Formats Showcase</h1>
    </header>
    <main>
        <section class="color-block" style="background-color: lightcoral;">
            <h2>Named Color: Light Coral</h2>
            <p>Color: lightcoral</p>
        </section>
        <section class="color-block" style="background-color: #4682b4;">
            <h2>Hex Color: #4682b4</h2>
            <p>Color: #4682b4</p>
        </section>
        <section class="color-block" style="background-color: rgb(60, 179, 113);">
            <h2>RGB Color: rgb(60, 179, 113)</h2>
            <p>Color: rgb(60, 179, 113)</p>
        </section>
        <section class="color-block" style="background-color: hsl(120, 60%, 50%);">
            <h2>HSL Color: hsl(120, 60%, 50%)</h2>
            <p>Color: hsl(120, 60%, 50%)</p>
        </section>
    </main>
    <footer>
        <p>Learn more about color formats and their usage!</p>
    </footer>
</body>
</html>
```

CSS Styles (styles.css)

```
* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
    font-family: Arial, sans-serif;
}

body {
    background-color: #f0f0f0;
    color: #333;
    line-height: 1.6;
}

header {
    text-align: center;
    padding: 20px;
    background-color: #4a90e2;
    color: white;
}

main {
    display: flex;
    flex-direction: column;
    align-items: center;
    padding: 20px;
}
.color-block {
    width: 80%;
    padding: 20px;
    margin: 15px 0;
    border-radius: 8px;
```

```
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
    color: white;
    text-align: center;
}

footer {
    text-align: center;
    padding: 20px;
    background-color: #4a90e2;
    color: white;
    position: relative;
    bottom: 0;
    width: 100%;
}
```

C. Develop a navigation menu using CSS to style links in their different states (unvisited, visited, hover, active).

HTML Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Navigation Menu</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <nav>
        <ul class="nav-menu">
            <li><a href="#home">Home</a></li>
            <li><a href="#about">About</a></li>
```

```
        <li><a href="#services">Services</a></li>
        <li><a href="#contact">Contact</a></li>
      </ul>
    </nav>
</body>
</html>
```

CSS Code (styles.css)

```
/* Basic styles for body and navigation */
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f0f0f0;
}

nav {
    background-color: #333;
    padding: 10px;
}

/* Navigation menu styles */
.nav-menu {
    list-style-type: none;
    padding: 0;
}


.nav-menu li {
    display: inline;
    margin-right: 20px;
```

```
}

/* Link styles */
.nav-menu a {
    color: #fff; /* Unvisited link */
    text-decoration: none;
    padding: 5px 10px;
}

/* Visited link */
.nav-menu a:visited {
    color: #b0b0b0; /* Gray for visited links */
}

/* Hover state */
.nav-menu a:hover {
    background-color: #575757; /* Darker background on hover */
    color: #fff; /* Maintain text color on hover */
}

/* Active state */
.nav-menu a:active {
    background-color: #444; /* Even darker background when active */
    color: #fff;
}
```

Explanation of the CSS States

Unvisited Links: Styled with a white color (#fff) by default.

Visited Links: Change to a gray color (#b0b0b0) to indicate they have been visited.

Hover State: When the user hovers over a link, it changes the background color to a darker shade (#575757), providing visual feedback.

Active State: When a link is actively being clicked, it shows an even darker background (#444) to indicate the action.

D. Create a webpage with a "card" design that utilize various CSS properties including borders, shadows, and backgrounds.

HTML Code

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Card Design Example</title>
   <link rel="stylesheet" href="styles.css">
</head>
<body>
   <div class="container">
      <div class="card">
         <img src="https://via.placeholder.com/300" alt="Card Image" class="card-image">
         <div class="card-content">
            <h2 class="card-title">Card Title</h2>
            <p class="card-description">This is a description of the card. It contains relevant information and can include additional details.</p>
            <a href="#" class="card-button">Read More</a>
         </div>
      </div>
   </div>
</body>
</html>
```

CSS Code (styles.css)

```
body {
   font-family: Arial, sans-serif;
   background-color: #f7f7f7;
```

```
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
}

.container {
    display: flex;
    justify-content: center;
}

.card {
    background-color: white;
    border-radius: 10px;
    box-shadow: 0 4px 20px rgba(0, 0, 0, 0.1);
    overflow: hidden;
    width: 300px;
    transition: transform 0.2s;
}
.card:hover {
    transform: scale(1.05);
}
.card-image {
    width: 100%;
    height: auto;
}

.card-content {
    padding: 20px;
}
```

```
.card-title {
    font-size: 1.5em;
    margin: 0;
    color: #333;
}

.card-description {
    color: #777;
    margin: 10px 0;
}

.card-button {
    display: inline-block;
    background-color: #007bff;
    color: white;
    padding: 10px 15px;
    text-decoration: none;
    border-radius: 5px;
    transition: background-color 0.3s;
}

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

---

3. CSS SELECTORS

A. Design a webpage that demonstrates the use of at least 7 different types of CSS selectors. Include comments in your CSS explaining each selector.

HTML (index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>CSS Selectors Demo</title>
   <link rel="stylesheet" href="styles.css">
</head>
<body>
   <h1>CSS Selectors Demonstration</h1>
   <p class="intro">This is a demonstration of various CSS selectors.</p>

   <div class="container">
      <p class="example" id="first">First Example</p>
      <p class="example">Second Example</p>
      <p class="example">Third Example</p>
   </div>
   <ul>
      <li class="list-item active">List Item 1</li>
      <li class="list-item">List Item 2</li>
      <li class="list-item">List Item 3</li>
   </ul>
   <a href="#" class="link">Example Link</a>
</body>
```

```
</html>
```

CSS (styles.css)

```
/* 1. Universal Selector */
* {
   box-sizing: border-box; /* Applies box-sizing to all elements */
}

/* 2. Type Selector */
h1 {
   color: darkblue; /* Styles all <h1> elements */
}

/* 3. Class Selector */
.intro {
   font-size: 1.2em; /* Styles all elements with the class "intro" */
}

/* 4. ID Selector */
#first {
   font-weight: bold; /* Styles the element with the ID "first" */
}

/* 5. Descendant Selector */
.container p {
   color: green; /* Styles all <p> elements that are inside an element with the class "container" */
}
/* 6. Attribute Selector */
a[href] {
   text-decoration: none; /* Styles all <a> elements with an href attribute */
```

```
}

/* 7. Pseudo-class Selector */
.list-item.active {
   background-color: yellow; /* Styles the <li> with class "list-item" and also class "active" */
}
```

B. Create an HTML form and use attribute selectors to style different input types uniquely.

HTML (form.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Styled Form</title>
   <link rel="stylesheet" href="styles.css">
</head>
<body>
   <h1>Styled Input Form</h1>
   <form>
      <label for="name">Name:</label>
      <input type="text" id="name" name="name" placeholder="Enter your name">

      <label for="email">Email:</label>
      <input type="email" id="email" name="email" placeholder="Enter your email">

      <label for="password">Password:</label>
      <input type="password" id="password" name="password" placeholder="Enter your password">

      <label for="number">Age:</label>
      <input type="number" id="number" name="age" min="1" max="120">
```

```
        <label for="color">Favorite Color:</label>
        <input type="color" id="color" name="color">

        <label for="date">Birth Date:</label>
        <input type="date" id="date" name="date">

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

CSS (styles.css)

```
/* Style for all input elements */
input {
    padding: 10px;
    margin: 10px 0;
    border-radius: 5px;
    border: 1px solid #ccc;
    width: 300px;
}

/* 1. Text Input */
input[type="text"] {
    border-color: blue; /* Blue border for text input */
}

/* 2. Email Input */
input[type="email"] {
    border-color: green; /* Green border for email input */
}
```

```
/* 3. Password Input */
input[type="password"] {
    border-color: red; /* Red border for password input */
}

/* 4. Number Input */
input[type="number"] {
    border-color: orange; /* Orange border for number input */
}

/* 5. Color Input */
input[type="color"] {
    padding: 0; /* Remove padding for color input */
    width: 50px; /* Set width for color input */
}

/* 6. Date Input */
input[type="date"] {
    border-color: purple; /* Purple border for date input */
}
/* 7. Button */
button {
    padding: 10px 15px;
    background-color: darkgray;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}
button:hover {
    background-color: lightgray; /* Change button color on hover */
}
```

Explanation of the CSS Selectors:

Text Input (input[type="text"]): Styles the text input with a blue border.

Email Input (input[type="email"]): Styles the email input with a green border.

Password Input (input[type="password"]): Styles the password input with a red border.

Number Input (input[type="number"]): Styles the number input with an orange border.

Color Input (input[type="color"]): Adjusts padding and sets a specific width for the color input.

Date Input (input[type="date"]): Styles the date input with a purple border.

Button: Styles the submit button and adds a hover effect.

C. Develop a multi-level navigation menu using descendant and child selectors to style each level differently.

HTML (nav-menu.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Multi-Level Navigation Menu</title>
   <link rel="stylesheet" href="styles.css">
</head>
<body>
   <nav>
      <ul class="menu">
         <li>
            <a href="#">Home</a>
         </li>
         <li>
            <a href="#">About</a>
            <ul class="submenu">
```

```
            <li><a href="#">Team</a></li>
            <li><a href="#">History</a></li>
            <li>
              <a href="#">Mission</a>
              <ul class="subsubmenu">
                <li><a href="#">Vision</a></li>
                <li><a href="#">Values</a></li>
              </ul>
            </li>
          </ul>
        </li>
        <li>
          <a href="#">Services</a>
          <ul class="submenu">
            <li><a href="#">Web Design</a></li>
            <li><a href="#">SEO</a></li>
          </ul>
        </li>
        <li>
          <a href="#">Contact</a>
        </li>
      </ul>
    </nav>
</body>
</html>
```

CSS (styles.css)

```
/* Base styles for the menu */
.menu {
    list-style: none; /* Remove default list styling */
    padding: 0;
    margin: 0;
```

```
    background-color: #333; /* Main menu background color */
}

/* Styles for top-level menu items */
.menu > li {
    display: inline-block; /* Display items in a row */
    position: relative; /* For positioning submenus */
}

/* Styles for top-level links */
.menu > li > a {
    color: white; /* Top-level link text color */
    padding: 15px 20px; /* Padding for top-level links */
    text-decoration: none; /* Remove underline */
    display: block; /* Make links block-level for padding */
}

/* Change background color on hover for top-level links */
.menu > li > a:hover {
    background-color: #444; /* Darker background on hover */
}

/* Styles for first-level submenus */
.submenu {
    list-style: none; /* Remove default list styling */
    padding: 0;
    margin: 0;
    display: none; /* Hide submenus by default */
    position: absolute; /* Position relative to parent */
    background-color: #555; /* Submenu background color */
}
```

```
/* Show submenu on hover of parent item */
.menu > li:hover > .submenu {
    display: block; /* Display submenu on hover */
}

/* Styles for submenu links */
.submenu li a {
    color: white; /* Submenu link text color */
    padding: 10px 15px; /* Padding for submenu links */
    display: block; /* Make submenu links block-level */
}

/* Change background color on hover for submenu links */
.submenu li a:hover {
    background-color: #666; /* Darker background on hover */
}

/* Styles for second-level submenus */
.subsubmenu {
    list-style: none; /* Remove default list styling */
    padding: 0;
    margin: 0;
    display: none; /* Hide subsubmenu by default */
    position: absolute; /* Position relative to parent */
    left: 100%; /* Position to the right of the parent */
    top: 0; /* Align with the top of the parent */
}

/* Show subsubmenu on hover of parent item */
.submenu li:hover > .subsubmenu {
    display: block; /* Display subsubmenu on hover */
}
```

```
/* Styles for subsubmenu links */
.subsubmenu li a {
    color: white; /* Subsubmenu link text color */
    padding: 10px 15px; /* Padding for subsubmenu links */
    display: block; /* Make subsubmenu links block-level */
}

/* Change background color on hover for subsubmenu links */
.subsubmenu li a:hover {
    background-color: #777; /* Darker background on hover */
}
```

D. Design a webpage that makes extensive use of pseudo-classes and pseudo-elements for interactive and decorative effects.

HTML (index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pseudo-Class and Pseudo-Element Demo</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <h1>Interactive Webpage with Pseudo-Classes and Pseudo-Elements</h1>
    </header>

    <main>
        <section>
```

```
        <h2>Interactive Buttons</h2>
        <button class="btn">Hover Me!</button>
        <button class="btn">Click Me!</button>
    </section>

    <section>
        <h2>Styled Links</h2>
        <a href="#">Normal Link</a>
        <a href="#">Another Link</a>
    </section>

    <section>
        <h2>Decorative List</h2>
        <ul class="decorative-list">
            <li>First Item</li>
            <li>Second Item</li>
            <li>Third Item</li>
        </ul>
    </section>
  </main>
</body>
</html>
```

CSS (styles.css)

```
/* Basic styles for the body and header */
body {
  font-family: Arial, sans-serif;
  background-color: #f9f9f9;
  color: #333;
  margin: 0;
  padding: 20px;
```

```
}

header {
    text-align: center;
    margin-bottom: 40px;
}

/* Button styles with pseudo-classes */
.btn {
    padding: 15px 25px;
    border: none;
    border-radius: 5px;
    background-color: #007BFF;
    color: white;
    font-size: 16px;
    cursor: pointer;
    position: relative;
    overflow: hidden;
    transition: background-color 0.3s;
}

/* Hover effect for buttons */
.btn:hover {
    background-color: #0056b3; /* Darker blue on hover */
}

/* Focus effect for buttons */
.btn:focus {
    outline: 2px solid #0056b3; /* Blue outline on focus */
}

/* Active state for buttons */
```

```
.btn:active {
    background-color: #003d7a; /* Even darker blue when clicked */
}

/* Before pseudo-element for buttons */
.btn::before {
    content: '';
    position: absolute;
    top: 50%;
    left: 50%;
    width: 300%;
    height: 300%;
    background: rgba(255, 255, 255, 0.3);
    border-radius: 50%;
    transform: translate(-50%, -50%) scale(0);
    transition: transform 0.5s;
}

/* Animate the before pseudo-element on hover */
.btn:hover::before {
    transform: translate(-50%, -50%) scale(1);
}

/* Link styles */
a {
    color: #007BFF;
    text-decoration: none;
    position: relative;
}

/* Hover effect for links */
a:hover {
```

```
    color: #0056b3; /* Darker blue on hover */
}

/* Underline effect on hover */
a:hover::after {
    content: '';
    display: block;
    width: 100%;
    height: 2px;
    background: #0056b3; /* Underline color */
    position: absolute;
    left: 0;
    bottom: -3px; /* Position it below the text */
    transition: width 0.3s;
    width: 100%; /* Full width on hover */
}

/* List styles */
.decorative-list {
    list-style-type: none; /* Remove default list styling */
    padding: 0;
}

/* List item styles */
.decorative-list li {
    padding: 10px;
    background: #e7f1ff;
    margin: 5px 0;
    border-left: 5px solid #007BFF; /* Decorative left border */
    transition: background 0.3s, transform 0.3s;
}
```

```
/* Hover effect for list items */
.decorative-list li:hover {
    background: #d0e4ff; /* Lighter blue on hover */
    transform: translateX(5px); /* Slight move to the right */
}

/* Before pseudo-element for list items */
.decorative-list li::before {
    content: '✓'; /* Checkmark before each list item */
    color: #007BFF;
    margin-right: 10px; /* Space between checkmark and text */
}
```

Explanation of Pseudo-Classes and Pseudo-Elements:

Pseudo-classes:

:hover: Changes the background color of buttons and links when hovered over.

:focus: Applies an outline to buttons when focused (e.g., via keyboard navigation).

:active: Changes the button color when clicked.

Pseudo-elements:

::before: Adds decorative elements before the buttons and list items. For buttons, it creates a ripple effect; for list items, it adds a checkmark.

::after: Creates a decorative underline effect for links when hovered over.

Transitions: Applied to various elements to create smooth hover effects, enhancing user interaction.

---

4. CSS Box Model

A. Create a series of nested div elements and use the CSS box model properties to create a "bullseye" target design

HTML (index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Bullseye Target Design</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="target">
        <div class="ring outer-ring"></div>
        <div class="ring middle-ring"></div>
        <div class="ring inner-ring"></div>
    </div>
</body>
</html>
```

CSS (styles.css)

```
/* Body styles for centering the target */
body {
    display: flex;
    justify-content: center;
    align-items: center;
```

```
    height: 100vh; /* Full viewport height */
    background-color: #f0f0f0; /* Light gray background */
    margin: 0;
}

/* Styles for the target */
.target {
    position: relative;
    width: 200px; /* Set the width of the target */
    height: 200px; /* Set the height of the target */
}

/* Common styles for all rings */
.ring {
    position: absolute; /* Position them relative to the target */
    border-radius: 50%; /* Make them circular */
}

/* Styles for the outer ring */
.outer-ring {
    width: 100%; /* Full size of the target */
    height: 100%;
    background-color: #d9534f; /* Red */
    box-sizing: border-box; /* Include padding/border in the width/height */
}

/* Styles for the middle ring */
.middle-ring {
    width: 75%; /* 75% of the target size */
    height: 75%;
    background-color: #5bc0de; /* Light blue */
    top: 12.5%; /* Center it within the outer ring */
```

```
    left: 12.5%; /* Center it within the outer ring */
}

/* Styles for the inner ring */
.inner-ring {
    width: 50%; /* 50% of the target size */
    height: 50%;
    background-color: #5cb85c; /* Green */
    top: 25%; /* Center it within the middle ring */
    left: 25%; /* Center it within the middle ring */
}
```

B.Design a product card that demonstrates proper use of margin, border, and padding to create an aesthetically pleasing layout.

HTML (product-card.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Product Card</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="product-card">
        <img src="https://via.placeholder.com/300x200" alt="Product Image" class="product-image">
        <div class="product-info">
            <h2 class="product-title">Product Title</h2>
            <p class="product-description">This is a brief description of the product. It's engaging and informative!</p>
            <span class="product-price">$29.99</span>
```

```
        <button class="add-to-cart">Add to Cart</button>
      </div>
    </div>
</body>
</html>
```

CSS (styles.css)

```
/* Basic styles for the body */
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4; /* Light gray background */
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh; /* Full viewport height */
    margin: 0;
}

/* Styles for the product card */
.product-card {
    background-color: white; /* White background for the card */
    border: 1px solid #ddd; /* Light gray border */
    border-radius: 10px; /* Rounded corners */
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1); /* Subtle shadow */
    width: 300px; /* Fixed width */
    overflow: hidden; /* Prevent content overflow */
    margin: 20px; /* Margin around the card */
}

/* Styles for the product image */
```

```
.product-image {
    width: 100%; /* Full width */
    height: auto; /* Maintain aspect ratio */
}

/* Styles for the product info section */
.product-info {
    padding: 15px; /* Padding inside the info section */
}

/* Styles for the product title */
.product-title {
    font-size: 1.5em; /* Larger font size */
    margin: 0 0 10px 0; /* Bottom margin to separate from description */
}

/* Styles for the product description */
.product-description {
    font-size: 1em; /* Normal font size */
    margin: 0 0 15px 0; /* Bottom margin to separate from price */
}

/* Styles for the product price */
.product-price {
    font-size: 1.2em; /* Slightly larger font size */
    font-weight: bold; /* Bold text */
    display: block; /* Display as block to take full width */
    margin-bottom: 10px; /* Bottom margin before button */
}

/* Styles for the button */
.add-to-cart {
```

```
    background-color: #007BFF; /* Blue background */
    color: white; /* White text */
    border: none; /* Remove border */
    border-radius: 5px; /* Rounded corners */
    padding: 10px; /* Padding inside the button */
    cursor: pointer; /* Pointer cursor on hover */
    transition: background-color 0.3s; /* Smooth background transition */
    width: 100%; /* Full width */
}

/* Button hover effect */
.add-to-cart:hover {
    background-color: #0056b3; /* Darker blue on hover */
}
```

Explanation:

HTML Structure:

The product card is created using a div with the class product-card, containing an image, title, description, price, and an "Add to Cart" button.

CSS Layout:

Margin: The card itself has a margin around it to provide space from surrounding elements.

Border: A light gray border gives definition to the card.

Padding: Inside the card, padding is used to create space between the content and the card edges, ensuring that the text does not touch the border.

Styling Elements:

The card uses a white background with rounded corners and a subtle shadow to give a lift effect.

Each element inside the card has its own styling, with appropriate margins to ensure spacing and a clean layout.

The button has a hover effect to enhance user interactivity.

C. Develop a website header with a logo and navigation menu, using the CSS box model to achieve precise spacing and alignment.

HTML (header.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Simple Header</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <header>
    <div class="logo">MyLogo</div>
    <nav>
      <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">About</a></li>
        <li><a href="#">Services</a></li>
        <li><a href="#">Contact</a></li>
      </ul>
    </nav>
  </header>
</body>
</html>
```

CSS (styles.css)

```
body {
  font-family: Arial, sans-serif;
}
```

```
header {
    display: flex;
    justify-content: space-between;
    background-color: #333;
    color: white;
    padding: 10px 20px;
}

nav ul {
    list-style: none;
    display: flex;
}

nav ul li {
    margin-left: 15px;
}

nav ul li a {
    color: white;
    text-decoration: none;
}
```

D. Create a photo gallery where each image is contained within a box with consistent margin, border, and padding, regardless of the image dimensions.

HTML (gallery.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Photo Gallery</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>Photo Gallery</h1>
    <div class="gallery">
        <div class="photo-box"><img src="https://via.placeholder.com/300x200" alt="Image 1"></div>
        <div class="photo-box"><img src="https://via.placeholder.com/300x250" alt="Image 2"></div>
        <div class="photo-box"><img src="https://via.placeholder.com/300x300" alt="Image 3"></div>
        <div class="photo-box"><img src="https://via.placeholder.com/300x400" alt="Image 4"></div>
        <div class="photo-box"><img src="https://via.placeholder.com/300x350" alt="Image 5"></div>
        <div class="photo-box"><img src="https://via.placeholder.com/300x450" alt="Image 6"></div>
    </div>
</body>
</html>
```

CSS (styles.css)

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

.gallery {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
    gap: 20px;
}
```

```
.photo-box {
    background-color: white;
    border: 2px solid #ddd;
    border-radius: 8px;
    padding: 10px;
}

.photo-box img {
    width: 100%;
    height: auto;
}
```

---

5. Advanced Cascading Style Sheets

A. Design a webpage with a long content area and use CSS to create a fixed header that remains visible when scrolling, demonstrating the use of position and z-index properties.

HTML (index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Fixed Header Example</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header class="header">
        <h1>My Website</h1>
```

```
      <nav>
        <ul class="nav-menu">
          <li><a href="#section1">Section 1</a></li>
          <li><a href="#section2">Section 2</a></li>
          <li><a href="#section3">Section 3</a></li>
          <li><a href="#section4">Section 4</a></li>
        </ul>
      </nav>
    </header>

    <main class="content">
      <section id="section1"><h2>Section 1</h2><p>Content goes here.</p></section>
      <section id="section2"><h2>Section 2</h2><p>Content goes here.</p></section>
      <section id="section3"><h2>Section 3</h2><p>Content goes here.</p></section>
      <section id="section4"><h2>Section 4</h2><p>Content goes here.</p></section>
    </main>
</body>
</html>
```

CSS (styles.css)

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
}

.header {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
```

```
    background-color: #333;
    color: white;
    padding: 15px;
    z-index: 1000;
}

.nav-menu {
    list-style: none;
    display: flex;
    margin: 0;
    padding: 0;
}

.nav-menu li {
    margin-left: 20px;
}

.nav-menu a {
    color: white;
    text-decoration: none;
}

.content {
    padding-top: 70px; /* Space for the fixed header */
}

section {
    padding: 20px;
    border-bottom: 1px solid #ddd;
}
```

B. Create a responsive grid layout using CSS Grid for a portfolio page that adjusts from one to three columns depending on screen size.

HTML (portfolio.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Portfolio</title>
   <link rel="stylesheet" href="styles.css">
</head>
<body>
   <h1>My Portfolio</h1>
   <div class="portfolio">
      <div class="item">Project 1</div>
      <div class="item">Project 2</div>
      <div class="item">Project 3</div>
      <div class="item">Project 4</div>
      <div class="item">Project 5</div>
      <div class="item">Project 6</div>
   </div>
</body>
</html>
```

CSS (styles.css)

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

```
}

h1 {
    text-align: center;
}

.portfolio {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr)); /* Responsive columns */
    gap: 20px; /* Space between items */
}

.item {
    background-color: white;
    border: 1px solid #ddd;
    border-radius: 8px;
    padding: 20px;
    text-align: center;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1); /* Subtle shadow */
}
```

C. Develop a flexible card layout using CSS Flexbox that adjusts the number of cards per row based on container width.

HTML (cards.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Card Layout</title>
    <link rel="stylesheet" href="styles.css">
</head>
```

```
<body>
    <h1>Card Layout</h1>
    <div class="card-container">
        <div class="card">Card 1</div>
        <div class="card">Card 2</div>
        <div class="card">Card 3</div>
        <div class="card">Card 4</div>
        <div class="card">Card 5</div>
        <div class="card">Card 6</div>
    </div>
</body>
</html>
```

CSS (styles.css)

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

h1 {
    text-align: center;
}

.card-container {
    display: flex; /* Use flexbox for layout */
    flex-wrap: wrap; /* Allow wrapping of cards */
    justify-content: space-between; /* Space between cards */
}
```

```
.card {
    background-color: white;
    border: 1px solid #ddd;
    border-radius: 8px;
    padding: 20px;
    margin: 10px; /* Space around each card */
    flex: 1 1 calc(30% - 20px); /* Flex-grow, flex-shrink, and base size */
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    text-align: center;
}

/* Responsive adjustments */
@media (max-width: 600px) {
    .card {
        flex: 1 1 calc(100% - 20px); /* Full width on smaller screens */
    }
}
```

D. Design an image gallery that uses CSS Grid for layout and CSS transitions for smooth hover effects.

HTML (gallery.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Image Gallery</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>Image Gallery</h1>
```

```
    <div class="gallery">
        <div class="gallery-item">
            <img src="https://via.placeholder.com/300" alt="Image 1">
        </div>
        <div class="gallery-item">
            <img src="https://via.placeholder.com/300" alt="Image 2">
        </div>
        <div class="gallery-item">
            <img src="https://via.placeholder.com/300" alt="Image 3">
        </div>
        <div class="gallery-item">
            <img src="https://via.placeholder.com/300" alt="Image 4">
        </div>
        <div class="gallery-item">
            <img src="https://via.placeholder.com/300" alt="Image 5">
        </div>
        <div class="gallery-item">
            <img src="https://via.placeholder.com/300" alt="Image 6">
        </div>
    </div>
</body>
</html>
```

CSS (styles.css)

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

h1 {
```

```
    text-align: center;
    margin-bottom: 20px;
}

.gallery {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(300px, 1fr)); /* Responsive columns */
    gap: 15px; /* Space between items */
}

.gallery-item {
    overflow: hidden; /* Hide overflow for smooth transition */
    border-radius: 8px; /* Rounded corners */
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1); /* Subtle shadow */
    transition: transform 0.3s ease; /* Smooth transition */
}

.gallery-item:hover {
    transform: scale(1.05); /* Slight zoom on hover */
}

.gallery-item img {
    width: 100%; /* Full width */
    height: auto; /* Maintain aspect ratio */
    display: block; /* Remove inline spacing */
}
```

6.Overflow

A. Create a text-heavy webpage with sections that have fixed heights and demonstrate different overflow property values (visible, hidden, scroll, auto).

HTML (text-heavy.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Overflow Property Demo</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="container">
    <h1>Overflow Property Demo</h1>

    <div class="section visible">
      <h2>Overflow: Visible</h2>
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
      <p>This section allows content to overflow without clipping it.</p>
    </div>

    <div class="section hidden">
      <h2>Overflow: Hidden</h2>
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
      <p>This section will clip the content that overflows the fixed height.</p>
    </div>

    <div class="section scroll">
```

```
        <h2>Overflow: Scroll</h2>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
        <p>This section will always show a scrollbar, even if the content fits.</p>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
        <p>More content to trigger the scrollbar.</p>
        <p>Even more content to ensure scrolling is necessary!</p>
      </div>

      <div class="section auto">
        <h2>Overflow: Auto</h2>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
        <p>This section will show a scrollbar only if the content overflows.</p>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
        <p>Additional content to check scrolling behavior.</p>
        <p>Even more content to see if scrolling is necessary!</p>
        <p>One last bit of content to definitely trigger a scrollbar!</p>
      </div>
    </div>
</body>
</html>
```

CSS Code (styles.css)

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

```
.container {
    max-width: 600px;
    margin: 0 auto;
}

h1 {
    text-align: center;
}

.section {
    height: 100px; /* Fixed height */
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #ccc;
    background-color: #fff;
}

.visible {
    overflow: visible;
}

.hidden {
    overflow: hidden;
}

.scroll {
    overflow: scroll;
}

.auto {
    overflow: auto;
}
```

B. Design a horizontal scrolling image gallery using CSS overflow properties

HTML Code

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Horizontal Scrolling Image Gallery</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="gallery-container">
    <h1>Horizontal Scrolling Image Gallery</h1>
    <div class="gallery">
      <img src="https://via.placeholder.com/300x200?text=Image+1" alt="Image 1">
      <img src="https://via.placeholder.com/300x200?text=Image+2" alt="Image 2">
      <img src="https://via.placeholder.com/300x200?text=Image+3" alt="Image 3">
      <img src="https://via.placeholder.com/300x200?text=Image+4" alt="Image 4">
      <img src="https://via.placeholder.com/300x200?text=Image+5" alt="Image 5">
    </div>
  </div>
</body>
</html>
```

CSS Code (styles.css)

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

```
.gallery-container {
    max-width: 100%;
    overflow-x: auto; /* Enable horizontal scrolling */
    white-space: nowrap; /* Prevent line breaks */
    padding: 10px 0;
}

h1 {
    text-align: center;
}

.gallery {
    display: inline-block; /* Align images horizontally */
}

.gallery img {
    height: 200px; /* Fixed height for images */
    margin-right: 10px; /* Space between images */
}
```

C. Develop a "card" component with a fixed height that uses text-overflow to handle varying amounts of text content gracefully.

HTML Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Card Component</title>
    <link rel="stylesheet" href="styles.css">
```

```
</head>
<body>
    <div class="card">
        <h2>Card Title</h2>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus lacinia odio vitae vestibulum. This text might overflow depending on the length of the content.</p>
    </div>
</body>
</html>
```

CSS Code (styles.css)

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

.card {
    width: 300px; /* Fixed width for the card */
    height: 150px; /* Fixed height for the card */
    padding: 15px;
    margin: 20px auto;
    border: 1px solid #ccc;
    border-radius: 8px;
    background-color: #fff;
    overflow: hidden; /* Hide overflow content */
    text-overflow: ellipsis; /* Add ellipsis for overflowing text */
    display: flex;
    flex-direction: column;
}
```

```
.card h2 {
    margin: 0 0 10px;
    font-size: 18px;
}

.card p {
    margin: 0;
    overflow: hidden; /* Hide overflow content */
    text-overflow: ellipsis; /* Add ellipsis for overflowing text */
    display: -webkit-box; /* Required for ellipsis in WebKit */
    -webkit-box-orient: vertical; /* Required for ellipsis in WebKit */
    -webkit-line-clamp: 3; /* Limit to 3 lines */
}
```

D. Create a tabbed interface where the content of each tab may exceed the container height, using overflow to manage the content.

HTML Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Tabbed Interface</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="tabs">
        <a href="#tab1" class="tab-button active">Tab 1</a>
        <a href="#tab2" class="tab-button">Tab 2</a>
        <a href="#tab3" class="tab-button">Tab 3</a>
    </div>
```

```
    <div class="tab-content">
        <div id="tab1" class="tab active">
            <h2>Tab 1 Content</h2>
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent suscipit...</p>
            <p>More content that exceeds the height of the container...</p>
            <p>(Add more content here to test overflow)</p>
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent suscipit.</p>
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent suscipit.</p>
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent suscipit.</p>
        </div>
        <div id="tab2" class="tab">
            <h2>Tab 2 Content</h2>
            <p>More content for Tab 2...</p>
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
            <p>(Add more content here to test overflow)</p>
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent suscipit.</p>
        </div>
        <div id="tab3" class="tab">
            <h2>Tab 3 Content</h2>
            <p>Details about Tab 3...</p>
            <p>(Add more content here to test overflow)</p>
            <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent suscipit.</p>
        </div>
    </div>
</body>
</html>
CSS (styles.css)
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
}
```

```
.tabs {
    display: flex;
    background-color: #f1f1f1;
}

.tab-button {
    padding: 10px 20px;
    cursor: pointer;
    text-decoration: none;
    color: black;
    border: none;
    background-color: inherit;
    transition: background-color 0.3s;
}

.tab-button:hover {
    background-color: #ddd;
}

.tab-button.active {
    background-color: #ccc;
}

.tab-content {
    border: 1px solid #ccc;
    padding: 20px;
    height: 200px; /* Set a fixed height for the container */
    overflow-y: auto; /* Enable vertical scrolling */
}
.tab {
    display: none;
}
```

```
.tab:target {
    display: block;
}

.tab.active {
    display: block;
}

#tab1:target ~ .tab-content .tab,
#tab2:target ~ .tab-content .tab,
#tab3:target ~ .tab-content .tab {
    display: none;
}
```

---

7. Display: Flex

A. Design a responsive navigation menu that switches between horizontal layout on large screens and vertical layout on small screens using Flexbox.

HTML Structure

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Responsive Navigation Menu</title>
    <link rel="stylesheet" href="styles.css">
</head>
```

```
<body>
    <nav class="navbar">
        <ul class="nav-links">
            <li><a href="#home">Home</a></li>
            <li><a href="#about">About</a></li>
            <li><a href="#services">Services</a></li>
            <li><a href="#contact">Contact</a></li>
        </ul>
    </nav>
</body>
</html>
```

CSS Styles (styles.css)

```
* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}

body {
    font-family: Arial, sans-serif;
}

.navbar {
    background-color: #333;
}

.nav-links {
    display: flex;
    justify-content: center;
    list-style: none;
```

```
    padding: 1rem;
}

.nav-links li {
    margin: 0 15px;
}

.nav-links a {
    color: white;
    text-decoration: none;
    padding: 0.5rem 1rem;
}

.nav-links a:hover {
    background-color: #575757;
    border-radius: 4px;
}

/* Responsive Styles */
@media (max-width: 768px) {
    .nav-links {
        flex-direction: column;
        align-items: center;
    }

    .nav-links li {
        margin: 10px 0;
    }
}
```

B. Create a centered card layout that uses Flexbox to distribute space evenly between multiple cards

HTML Structure

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Centered Card Layout</title>
   <link rel="stylesheet" href="styles.css">
</head>
<body>
   <div class="card-container">
      <div class="card">
         <h2>Card 1</h2>
         <p>This is a description for Card 1.</p>
      </div>
      <div class="card">
         <h2>Card 2</h2>
         <p>This is a description for Card 2.</p>
      </div>
      <div class="card">
         <h2>Card 3</h2>
         <p>This is a description for Card 3.</p>
      </div>
      <div class="card">
         <h2>Card 4</h2>
         <p>This is a description for Card 4.</p>
      </div>
   </div>
</body>
</html>
```

CSS Styles (styles.css)

```
* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}

body {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #f0f0f0;
}

.card-container {
    display: flex;
    justify-content: center;
    flex-wrap: wrap;
    gap: 20px; /* Space between cards */
    padding: 20px;
}

.card {
    background-color: white;
    border: 1px solid #ccc;
    border-radius: 8px;
    padding: 20px;
    width: 200px; /* Fixed width for cards */
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
    text-align: center;
}
```

```
.card h2 {
    margin-bottom: 10px;
}

.card p {
    color: #666;
}
```

C. Develop a form layout that uses Flexbox to align labels and inputs, ensuring consistent spacing and alignment

HTML Structure

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Flexbox Form Layout</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="form-container">
        <form>
            <div class="form-group">
                <label for="name">Name:</label>
                <input type="text" id="name" name="name" required>
            </div>
            <div class="form-group">
                <label for="email">Email:</label>
                <input type="email" id="email" name="email" required>
            </div>
            <div class="form-group">
```

```
            <label for="password">Password:</label>
            <input type="password" id="password" name="password" required>
        </div>
        <div class="form-group">
            <label for="message">Message:</label>
            <textarea id="message" name="message" rows="4" required></textarea>
        </div>
        <button type="submit">Submit</button>
    </form>
  </div>
</body>
</html>
```

CSS Styles (styles.css)

```
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}

body {
  font-family: Arial, sans-serif;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  background-color: #f0f0f0;
}
.form-container {
  background-color: white;
  border-radius: 8px;
  padding: 20px;
```

```
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
    width: 300px; /* Fixed width for the form */
}
.form-group {
    display: flex;
    flex-direction: column; /* Stack label and input */
    margin-bottom: 15px; /* Space between form groups */
}
label {
    margin-bottom: 5px; /* Space between label and input */
}

input, textarea {
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 4px;
    font-size: 16px;
}
button {
    padding: 10px;
    background-color: #007bff;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 16px;
}

button:hover {
    background-color: #0056b3; /* Darker shade on hover */
}
```

D. Design a page footer with multiple columns of links that reflow responsively using Flexbox.

HTML Structure

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Responsive Footer</title>
   <link rel="stylesheet" href="styles.css">
</head>
<body>
   <footer class="footer">
      <div class="footer-container">
         <div class="footer-column">
            <h3>About Us</h3>
            <ul>
               <li><a href="#company">Our Company</a></li>
               <li><a href="#team">Our Team</a></li>
               <li><a href="#careers">Careers</a></li>
            </ul>
         </div>
         <div class="footer-column">
            <h3>Services</h3>
            <ul>
               <li><a href="#consulting">Consulting</a></li>
               <li><a href="#development">Development</a></li>
               <li><a href="#design">Design</a></li>
            </ul>
         </div>
         <div class="footer-column">
```

```
            <h3>Support</h3>
            <ul>
                <li><a href="#faq">FAQ</a></li>
                <li><a href="#contact">Contact Us</a></li>
                <li><a href="#terms">Terms of Service</a></li>
            </ul>
        </div>
        <div class="footer-column">
            <h3>Follow Us</h3>
            <ul>
                <li><a href="#facebook">Facebook</a></li>
                <li><a href="#twitter">Twitter</a></li>
                <li><a href="#instagram">Instagram</a></li>
            </ul>
        </div>
    </div>
  </footer>
</body>
</html>
```

CSS Styles (styles.css)

```
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}
body {
  font-family: Arial, sans-serif;
}
```

```
.footer {
    background-color: #333;
    color: white;
    padding: 20px 0;
}

.footer-container {
    display: flex;
    flex-wrap: wrap; /* Allow columns to wrap */
    justify-content: center; /* Center the columns */
    max-width: 1200px;
    margin: 0 auto;
    padding: 0 20px;
}

.footer-column {
    flex: 1 1 200px; /* Grow, shrink, and set base width */
    margin: 10px;
    min-width: 150px; /* Ensure minimum width for responsiveness */
}

.footer-column h3 {
    margin-bottom: 10px;
}

.footer-column ul {
    list-style: none;
}
.footer-column a {
    color: white;
    text-decoration: none;
}
```

```
.footer-column a:hover {
    text-decoration: underline; /* Underline on hover */
}
```

---

8. Display: Grid

A. Create a complex page layout with header, footer, main content area, and sidebars using CSS Grid.

HTML Structure

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Grid Layout</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="grid-container">
        <header class="header">Header</header>
        <aside class="sidebar-left">Left Sidebar</aside>
        <main class="main-content">Main Content Area</main>
        <aside class="sidebar-right">Right Sidebar</aside>
        <footer class="footer">Footer</footer>
    </div>
</body>
</html>
```

CSS (styles.css)

```
body {
    margin: 0;
    font-family: Arial, sans-serif;
}

.grid-container {
    display: grid;
    grid-template-columns: 1fr 3fr 1fr; /* 1 column for each sidebar and 3 for the main content */
    grid-template-rows: auto 1fr auto; /* auto for header and footer, 1fr for content */
    height: 100vh; /* Full height of the viewport */
}

.header {
    grid-column: 1 / -1; /* Header spans all columns */
    background: #4CAF50;
    color: white;
    text-align: center;
    padding: 20px;
}

.sidebar-left {
    background: #f4f4f4;
    padding: 20px;
}
.main-content {
    background: #fff;
    padding: 20px;
}
.sidebar-right {
    background: #f4f4f4;
    padding: 20px;
}
```

```
.footer {
    grid-column: 1 / -1; /* Footer spans all columns */
    background: #4CAF50;
    color: white;
    text-align: center;
    padding: 10px;
}
```

B. Design a responsive image gallery that rearranges images from a 3x3 grid on large screens to a 2x2 grid on medium screens and a single column on small screens. give a simple code

HTML Structure

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Responsive Image Gallery</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="gallery">
        <img src="https://via.placeholder.com/300" alt="Image 1">
        <img src="https://via.placeholder.com/300" alt="Image 2">
        <img src="https://via.placeholder.com/300" alt="Image 3">
        <img src="https://via.placeholder.com/300" alt="Image 4">
        <img src="https://via.placeholder.com/300" alt="Image 5">
        <img src="https://via.placeholder.com/300" alt="Image 6">
        <img src="https://via.placeholder.com/300" alt="Image 7">
        <img src="https://via.placeholder.com/300" alt="Image 8">
        <img src="https://via.placeholder.com/300" alt="Image 9">
```

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

CSS (styles.css)

```
body {
    margin: 0;
    font-family: Arial, sans-serif;
}

.gallery {
    display: grid;
    gap: 10px; /* Space between images */
    padding: 10px;
}

.gallery img {
    width: 100%;
    height: auto; /* Maintain aspect ratio */
    border-radius: 8px; /* Optional: Rounded corners */
}

/* Large screens (3x3 grid) */
@media (min-width: 1024px) {
    .gallery {
        grid-template-columns: repeat(3, 1fr); /* 3 columns */
    }
}

/* Medium screens (2x2 grid) */
@media (min-width: 600px) and (max-width: 1023px) {
    .gallery {
```

```
    grid-template-columns: repeat(2, 1fr); /* 2 columns */
  }
}

/* Small screens (single column) */
@media (max-width: 599px) {
  .gallery {
    grid-template-columns: 1fr; /* 1 column */
  }
}
```

C. Develop a calendar interface for a single month using CSS Grid, with appropriate handling for days that fall outside the month.

HTML Structure

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Simple Calendar</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="calendar">
    <div class="header">April 2024</div>
    <div class="day">Sun</div>
    <div class="day">Mon</div>
    <div class="day">Tue</div>
    <div class="day">Wed</div>
    <div class="day">Thu</div>
    <div class="day">Fri</div>
```

```
<div class="day">Sat</div>

<!-- Days of the month (Assuming April starts on a Monday) -->
<div class="empty"></div> <!-- Empty space for days before the 1st -->
<div class="day">1</div>
<div class="day">2</div>
<div class="day">3</div>
<div class="day">4</div>
<div class="day">5</div>
<div class="day">6</div>
<div class="day">7</div>
<div class="day">8</div>
<div class="day">9</div>
<div class="day">10</div>
<div class="day">11</div>
<div class="day">12</div>
<div class="day">13</div>
<div class="day">14</div>
<div class="day">15</div>
<div class="day">16</div>
<div class="day">17</div>
<div class="day">18</div>
<div class="day">19</div>
<div class="day">20</div>
<div class="day">21</div>
<div class="day">22</div>
<div class="day">23</div>
<div class="day">24</div>
<div class="day">25</div>
<div class="day">26</div>
<div class="day">27</div>
<div class="day">28</div>
```

```
        <div class="day">29</div>
        <div class="day">30</div>
        <div class="day">31</div>
        <div class="empty"></div> <!-- Empty space for days after the last day -->
        <div class="empty"></div>
    </div>
</body>
</html>
```

CSS (styles.css)

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #f0f0f0;
}

.calendar {
    display: grid;
    grid-template-columns: repeat(7, 1fr);
    gap: 5px;
    width: 300px;
    background-color: white;
    border: 1px solid #ccc;
    border-radius: 8px;
    padding: 10px;
}
```

```
.header {
    grid-column: span 7;
    text-align: center;
    font-weight: bold;
    padding: 10px 0;
    background-color: #4CAF50;
    color: white;
    border-radius: 5px;
}

.day {
    background-color: #e7f3fe;
    text-align: center;
    padding: 15px;
    border-radius: 5px;
}

.empty {
    background-color: transparent; /* No background for empty spaces */
}

/* Highlighting today's date (example: 15) */
.day:nth-child(16) {
    background-color: #ffa;
    border: 2px solid #ff9800; /* Optional: Highlight today's date */
}
```

D. Create a dashboard layout with widgets of varying sizes arranged in a grid, demonstrating the use of grid-template-areas for easy rearrangement.

HTML Structure

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Dashboard Layout</title>
   <link rel="stylesheet" href="styles.css">
</head>
<body>
   <div class="dashboard">
      <div class="widget header">Header</div>
      <div class="widget sidebar">Sidebar</div>
      <div class="widget main">Main Content</div>
      <div class="widget widget1">Widget 1</div>
      <div class="widget widget2">Widget 2</div>
      <div class="widget widget3">Widget 3</div>
      <div class="widget footer">Footer</div>
   </div>
</body>
</html>
```

CSS (styles.css)

```
body {
   margin: 0;
   font-family: Arial, sans-serif;
}

.dashboard {
   display: grid;
```

```
    grid-template-areas:
        "header header header"
        "sidebar main main"
        "widget1 widget2 widget3"
        "footer footer footer";
    grid-template-columns: 1fr 3fr;
    grid-template-rows: auto 1fr auto;
    gap: 10px;
    height: 100vh; /* Full height */
    padding: 10px;
}

.widget {
    background-color: #e0f7fa;
    padding: 20px;
    border-radius: 8px;
    text-align: center;
}

.header {
    grid-area: header;
    background-color: #00796b;
    color: white;
}

.sidebar {
    grid-area: sidebar;
    background-color: #4db6ac;
}

.main {
    grid-area: main;
```

```
    background-color: #b2dfdb;
}

.widget1 {
    grid-area: widget1;
    background-color: #80cbc4;
}

.widget2 {
    grid-area: widget2;
    background-color: #4db6ac;
}

.widget3 {
    grid-area: widget3;
    background-color: #00796b;
}

.footer {
    grid-area: footer;
    background-color: #004d40;
    color: white;
}
```

----------The End----------