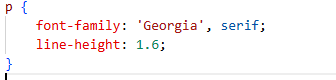
**Lab 0:Foundation & Styling**

**Objective:** Understand different ways to apply CSS.

* **Exercise:**
  + Create an HTML page with h1, p, a (link), and img tags.
  + Apply inline CSS to change the color of one paragraph.
  + Apply internal CSS to style all h1 elements with a specific font and color.
  + Create an external CSS file and link it to the HTML page. Style all p elements with a different font family and line-height.

**Code:**

**Output:**



**Lab 1: CSS Selectors & Basic Styling**  
**Objective:** Learn to target elements via element, class, id, attribute, pseudo‑class, and pseudo‑element selectors.

* **Exercise:**
  1. Create an HTML page with headings, paragraphs, lists, and a <div> with a unique id.
  2. Use element selectors to color all <p> tags.
  3. Use class selectors to style a set of <li> items differently.
  4. Use an attribute selector to style all <a> tags whose href contains “http.”
  5. Demonstrate a pseudo‑class (:hover) and pseudo‑element (::first-letter) rule.
  6. **Use the :nth-child(even) pseudo-class** to style every even <li> in a list with a different background color.
  7. **Use the ::before pseudo-element** on a heading (e.g., <h2>) to insert a decorative symbol or label before the text.

Code:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Advanced CSS Example</title>

    <link rel="stylesheet" href="L1css.css">

</head>

<body>

    <h1>My Web Page</h1>

    <h2>Section Title</h2>

    <p>This is a paragraph that will be styled using an element selector.</p>

    <p>Here's another paragraph to demonstrate styling.</p>

    <div id="uniqueDiv">

        <p>This is inside a uniquely identified div.</p>

    </div>

    <ul>

        <li class="highlight">Item 1</li>

        <li>Item 2</li>

        <li class="highlight">Item 3</li>

        <li>Item 4</li>

        <li>Item 5</li>

    </ul>

    <p>Check out these useful links:</p>

    <a href="http://httpforever.com"> HTTP Link</a><br>

    <a href="https://google.com">Secure HTTPS Link</a><br>

    <a href="/FS Lab.html">Local Link</a>

</body>

</html>

/\* 2. Element selector to color all <p> tags \*/

p {

    color: #333;

    font-size: 16px;

}

/\* 3. Class selector to style certain <li> items \*/

li.highlight {

    font-weight: bold;

    color: darkred;

}

/\* 4. Attribute selector for <a> tags containing "http" in href \*/

a[href\*="http"] {

    color: teal;

    text-decoration: underline;

}

/\* 5a. Pseudo-class :hover for links \*/

a:hover {

    background-color: yellow;

    transition: background-color 0.3s;

}

/\* 5b. Pseudo-element ::first-letter on paragraphs \*/

p::first-letter {

    font-size: 200%;

    color: darkblue;

    font-weight: bold;

}

/\* 6. :nth-child(even) for list items \*/

ul li:nth-child(even) {

    background-color: #b82626;

}

/\* 7. ::before pseudo-element on h2 \*/

h2::before {

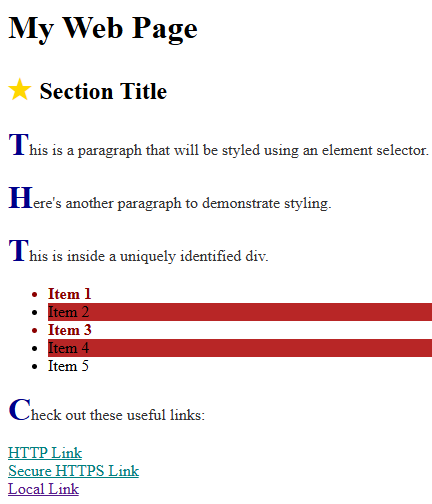
    content: "★ ";

    color: gold;

    font-size: 1.2em;

}

Output:



**Lab 2: The CSS Box Model & Layout Fundamentals**  
**Objective:** Visualize and manipulate content, padding, border, and margin; practice block vs inline.

* **Exercise:**
  1. Create three adjacent <div>s with colored borders.
  2. Adjust padding/margin to control spacing.
  3. Use display: inline-block vs block to observe behavior.
  4. Experiment with border-radius to create rounded corners.
  5. Use box-sizing: border-box and box-sizing: content-box to observe their effects on element sizing.

Code:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Box Model Comparison</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      background: #f9f9f9;

      padding: 20px;

    }

    h2 {

      margin-top: 40px;

    }

    .container {

      background: #eee;

      padding: 10px;

    }

    .box {

      width: 150px;

      height: 100px;

      margin: 10px;

      padding: 20px;

      border: 3px solid;

      border-radius: 15px;

      text-align: center;

      vertical-align: top;

    }

    /\* Display: inline-block \*/

    .inline-block-box {

      display: inline-block;

    }

    /\* Display: block \*/

    .block-box {

      display: block;

    }

    .content-box {

      box-sizing: content-box;

      border-color: crimson;

      background-color: #ffe5e5;

    }

    .border-box {

      box-sizing: border-box;

      border-color: green;

      background-color: #e5ffe5;

    }

    .default-box {

      box-sizing: content-box;

      border-color: blue;

      background-color: #e5eaff;

    }

    /\* Labels for better clarity \*/

    .label {

      font-weight: bold;

      margin-top: 5px;

    }

  </style>

</head>

<body>

  <h1>CSS Box Model & Display Demo</h1>

  <h2>1. Inline-Block Boxes (Appear Side-by-Side)</h2>

  <div class="container">

    <div class="box inline-block-box content-box">

      Content-box<br>(inline-block)

    </div>

    <div class="box inline-block-box border-box">

      Border-box<br>(inline-block)

    </div>

    <div class="box inline-block-box default-box">

      Default Box<br>(inline-block)

    </div>

  </div>

  <h2>2. Block Boxes (Stack Vertically)</h2>

  <div class="container">

    <div class="box block-box content-box">

      Content-box<br>(block)

    </div>

    <div class="box block-box border-box">

      Border-box<br>(block)

    </div>

    <div class="box block-box default-box">

      Default Box<br>(block)

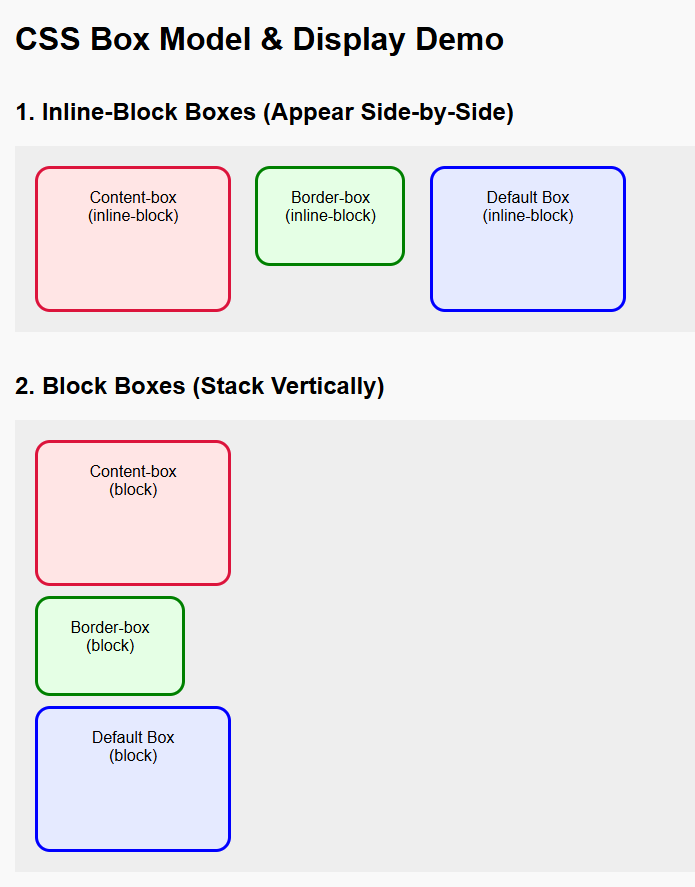
    </div>

  </div>

</body>

</html>

Output:



**Lab 3: Advanced Positioning & Float**  
**Objective:** Use position (static, relative, absolute, fixed) and float to create simple layouts.

* **Exercise:**
  1. Float an image left of a paragraph of text.
  2. Create a two‑column layout: fixed sidebar (absolute) + fluid content area.
  3. Implement a sticky header using position: fixed.

Code:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Layout Demo</title>

  <style>

    /\* General Reset \*/

    \* {

      box-sizing: border-box;

      margin: 0;

      padding: 0;

    }

    body {

      font-family: sans-serif;

      padding-top: 60px; /\* for sticky header space \*/

    }

    /\* 3. Sticky Header \*/

    header {

      position: fixed;

      top: 0;

      left: 0;

      width: 100%;

      background-color: #333;

      color: #fff;

      padding: 15px 20px;

      text-align: center;

      z-index: 999;

      box-shadow: 0 2px 5px rgba(0,0,0,0.2);

    }

    /\* 2. Sidebar (fixed-width, absolute position) \*/

    .sidebar {

      position: absolute;

      top: 60px; /\* below header \*/

      left: 0;

      width: 200px;

      height: 100%;

      background-color: #f2f2f2;

      padding: 20px;

      border-right: 2px solid #ccc;

    }

    /\* 2. Content area (fluid width) \*/

    .content {

      margin-left: 220px; /\* push it to the right of sidebar \*/

      padding: 20px;

    }

    /\* 1. Float image to left of text \*/

    .float-container {

      overflow: auto; /\* clearfix \*/

      margin-top: 20px;

    }

    .float-img {

      float: left;

      height: 150px;

      width: 200px;

      margin-right: 20px;

      border: 3px solid #999;

      border-radius: 10px;

    }

    p {

      line-height: 1.6;

    }

  </style>

</head>

<body>

  <header>

    <h1>Sticky Header</h1>

  </header>

  <div class="sidebar">

    <h2>Sidebar</h2>

    <p>Navigation or sidebar content goes here.</p>

  </div>

  <div class="content">

    <h2>Main Content Area</h2>

    <div class="float-container">

      <img src="city.jpg" alt="City" class="float-img">

      <p>

        This paragraph wraps around an image that is floated to the left. Notice how the text flows neatly to the right of the image. Chicago, on Lake Michigan in Illinois, is among the largest cities in the U.S. Famed for its bold architecture, it has a skyline punctuated by skyscrapers such as the iconic John Hancock Center, 1,451-ft. Willis Tower (formerly the Sears Tower) and the neo-Gothic Tribune Tower. The city is also renowned for its museums, including the Art Institute of Chicago with its noted Impressionist and Post-Impressionist works.

      </p>

    </div> <br>

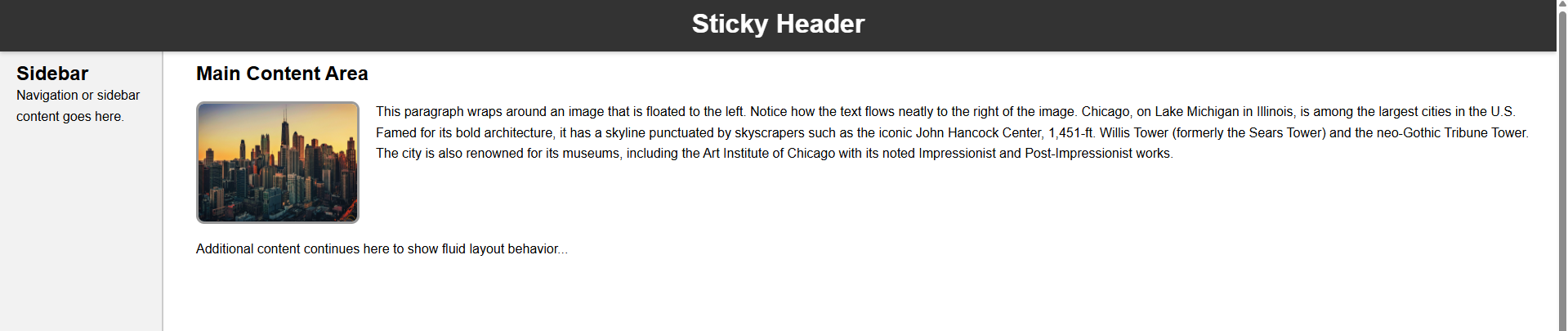
    <p>Additional content continues here to show fluid layout behavior...</p>

  </div>

</body>

</html>

Output:



**Lab 4: Navigation Bar & Responsive Media Queries**  
**Objective:** Build a horizontal navbar, then make it responsive via media queries.

* **Exercise:**
  1. Create a <ul>‑based navbar styled with display: inline-block.
  2. At ≤ 600 px viewport width, switch to a vertical (stacked) menu using @media.

Code:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Layout Demo</title>

  <style>

    \* {

      box-sizing: border-box;

      margin: 0;

      padding: 0;

    }

    body {

      font-family: sans-serif;

      padding-top: 100px; /\* Extra space for larger header with nav \*/

    }

    header {

      position: fixed;

      top: 0;

      left: 0;

      width: 100%;

      background-color: #333;

      color: #fff;

      padding: 10px 20px;

      text-align: center;

      z-index: 999;

      box-shadow: 0 2px 5px rgba(0,0,0,0.2);

    }

    nav ul {

      list-style: none;

      padding: 0;

      margin-top: 10px;

    }

    nav ul li {

      display: inline-block;

      margin: 0 15px;

    }

    nav ul li a {

      color: #fff;

      text-decoration: none;

      font-weight: bold;

      padding: 5px 10px;

    }

    nav ul li a:hover {

      background-color: #555;

      border-radius: 5px;

    }

    .sidebar {

      position: absolute;

      top: 100px;

      left: 0;

      width: 200px;

      height: 100%;

      background-color: #f2f2f2;

      padding: 20px;

      border-right: 2px solid #ccc;

    }

    .content {

      margin-left: 220px;

      padding: 20px;

    }

    .float-container {

      overflow: auto;

      margin-top: 20px;

    }

    .float-img {

      float: left;

      height: 150px;

      width: 200px;

      margin-right: 20px;

      border: 3px solid #999;

      border-radius: 10px;

    }

    p {

      line-height: 1.6;

    }

    /\* 2. Responsive Navbar: stacked on small screens \*/

    @media (max-width: 600px) {

      nav ul {

        text-align: left;

      }

      nav ul li {

        display: block;

        margin: 8px 0;

      }

      .sidebar {

        position: static;

        width: 100%;

        border-right: none;

        border-top: 2px solid #ccc;

      }

      .content {

        margin-left: 0;

      }

    }

  </style>

</head>

<body>

  <header>

    <h1>Chicago</h1>

    <nav>

      <ul>

        <li><a href="#">Home</a></li>

        <li><a href="#">Gallery</a></li>

        <li><a href="#">About</a></li>

        <li><a href="#">Contact</a></li>

      </ul>

    </nav>

  </header>

  <div class="sidebar">

    <h2>Sidebar</h2>

    <p>Navigation or sidebar content goes here.</p>

  </div>

  <div class="content">

    <h2>Main Content Area</h2>

    <div class="float-container">

      <img src="city.jpg" alt="City" class="float-img">

      <p>

        This paragraph wraps around an image that is floated to the left. Notice how the text flows neatly to the right of the image. Chicago, on Lake Michigan in Illinois, is among the largest cities in the U.S. Famed for its bold architecture, it has a skyline punctuated by skyscrapers such as the iconic John Hancock Center, 1,451-ft. Willis Tower (formerly the Sears Tower) and the neo-Gothic Tribune Tower. The city is also renowned for its museums, including the Art Institute of Chicago with its noted Impressionist and Post-Impressionist works.

      </p>

    </div> <br>

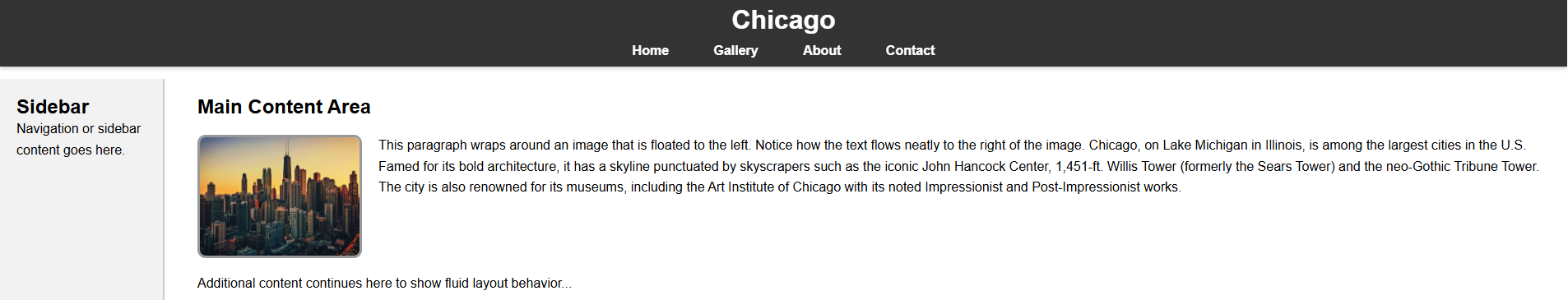
    <p>Additional content continues here to show fluid layout behavior...</p>

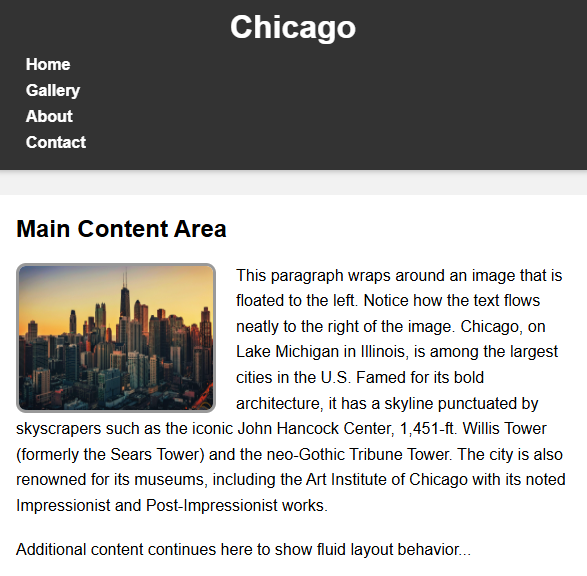
  </div>

</body>

</html>

Output:





**Lab 5: CSS Transitions & Animations**  
**Objective:** Add hover‑transitions and a keyframe animation.

* **Exercise:**
  1. On hover, smoothly change a button’s background over 0.3 s.
  2. Define a @keyframes rule to animate an element’s transform: rotate() indefinitely.

Code:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Hover & Rotate Animation</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      padding: 40px;

      text-align: center;

    }

    .animated-button {

      background-color: #3498db;

      color: white;

      padding: 12px 24px;

      border: none;

      border-radius: 5px;

      cursor: pointer;

      font-size: 16px;

      transition: background-color 0.3s ease;

    }

    .animated-button:hover {

      background-color: #d61616;

    }

    .rotating-box {

      display: inline-flex;

      justify-content: center;

      align-items: center;

      width: 100px;

      height: 100px;

      margin-top: 50px; /\* Space between button and box \*/

      background-color: #f39c12;

      color: white;

      font-weight: bold;

      border-radius: 10px;

      animation: rotateMe 2s linear infinite;

    }

    @keyframes rotateMe {

      from {

        transform: rotate(0deg);

      }

      to {

        transform: rotate(360deg);

      }

    }

  </style>

</head>

<body>

  <h2>Hover Button and Rotating Element</h2>

  <button class="animated-button">Hover Me</button>

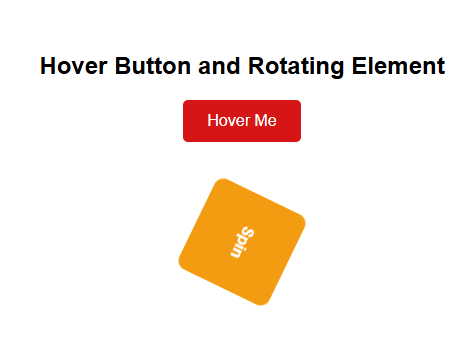
  <br>

  <div class="rotating-box">Spin</div>

</body>

</html>

**Output:**



**Lab 6: Layouts with Floats and Flexbox (or Grid)**

* **Objective:** Implement different page layouts using modern CSS techniques.
* **Exercise:**
  + **Option A (Floats):** Design a two or three-column layout for a simple blog page using float properties. Include a header, navigation, content area, sidebar, and footer. Clear floats appropriately.
  + **Option B (Flexbox):** Design a responsive navigation bar using Flexbox, where menu items are evenly spaced and stack vertically on smaller screens.
  + **Option C (Flexbox & Grid):** Create a simple photo gallery using CSS Grid for the overall layout and Flexbox for individual image captions/details.

Code:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>All CSS Layouts in One Page</title>

  <style>

    \* {

      box-sizing: border-box;

      margin: 0;

      padding: 0;

    }

    body {

      font-family: Arial, sans-serif;

      padding: 20px;

      background-color: #f5f5f5;

    }

    h1 {

      text-align: center;

      margin-bottom: 30px;

    }

    section {

      margin-bottom: 60px;

      background: #fff;

      padding: 20px;

      border-radius: 8px;

    }

    section h2 {

      margin-bottom: 20px;

      color: #2c3e50;

    }

    /\* --- Layout A: Float Blog --- \*/

    .float-layout header, .float-layout footer {

      background-color: #333;

      color: white;

      text-align: center;

      padding: 15px;

    }

    .float-layout nav {

      float: left;

      width: 20%;

      background-color: #f4f4f4;

      padding: 15px;

    }

    .float-layout .content {

      float: left;

      width: 50%;

      padding: 15px;

    }

    .float-layout .sidebar {

      float: left;

      width: 30%;

      background-color: #f9f9f9;

      padding: 15px;

    }

    .float-layout .clearfix::after {

      content: "";

      display: table;

      clear: both;

    }

    .float-layout footer {

      clear: both;

    }

    /\* --- Layout B: Flexbox Navbar --- \*/

    .flex-nav nav {

      background-color: #333;

      padding: 10px;

      margin-bottom: 20px;

    }

    .flex-nav ul {

      display: flex;

      justify-content: space-around;

      list-style: none;

    }

    .flex-nav ul li a {

      color: white;

      text-decoration: none;

      padding: 10px;

      display: block;

    }

    .flex-nav ul li a:hover {

      background: #444;

      border-radius: 4px;

    }

    @media (max-width: 600px) {

      .flex-nav ul {

        flex-direction: column;

        align-items: flex-start;

      }

      .flex-nav ul li {

        width: 100%;

      }

    }

    /\* --- Layout C: Grid Gallery --- \*/

    .gallery {

      display: grid;

      grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));

      gap: 20px;

    }

    .photo-card {

      border: 1px solid #ccc;

      border-radius: 8px;

      overflow: hidden;

      display: flex;

      flex-direction: column;

    }

    .photo-card img {

      width: 100%;

      height: auto;

    }

    .caption {

      background: #f0f0f0;

      padding: 10px;

      display: flex;

      justify-content: space-between;

    }

  </style>

</head>

<body>

  <h1>CSS Layout Demonstrations</h1>

  <!-- Section 1: Float Blog Layout -->

  <section>

    <h2>Float Blog Layout</h2>

    <div class="float-layout">

      <header>

        <h3>Float Blog Header</h3>

      </header>

      <div class="clearfix">

        <nav>

          <h4>Navigation</h4>

          <ul>

            <li>Home</li>

            <li>Blog</li>

            <li>Contact</li>

          </ul>

        </nav>

        <div class="content">

          <h4>Blog Post</h4>

          <p>This is the main content area using float layout.</p>

        </div>

        <aside class="sidebar">

          <h4>Sidebar</h4>

          <p>Additional content or widgets.</p>

        </aside>

      </div>

      <footer>

        <p>Float Footer © 2025</p>

      </footer>

    </div>

  </section>

  <!-- Section 2: Flexbox Navigation -->

  <section>

    <h2>Flexbox Navigation Bar</h2>

    <div class="flex-nav">

      <nav>

        <ul>

          <li><a href="#">Home</a></li>

          <li><a href="#">Gallery</a></li>

          <li><a href="#">Blog</a></li>

          <li><a href="#">Contact</a></li>

        </ul>

      </nav>

    </div>

    <p>Resize the screen to see how the navigation items stack vertically on small screens.</p>

  </section>

  <!-- Section 3: Grid Photo Gallery -->

  <section>

    <h2>CSS Grid Photo Gallery</h2>

    <div class="gallery">

      <div class="photo-card">

        <img src="FS.png" alt="Image 1">

        <div class="caption">

          <span>Photo 1</span>

          <span>📸</span>

        </div>

      </div>

      <div class="photo-card">

        <img src="city.jpg" alt="Image 2">

        <div class="caption">

          <span>Photo 2</span>

          <span>🌇</span>

        </div>

      </div>

      <div class="photo-card">

        <img src="ny.jpg" alt="Image 3">

        <div class="caption">

          <span>Photo 3</span>

          <span>📷</span>

        </div>

      </div>

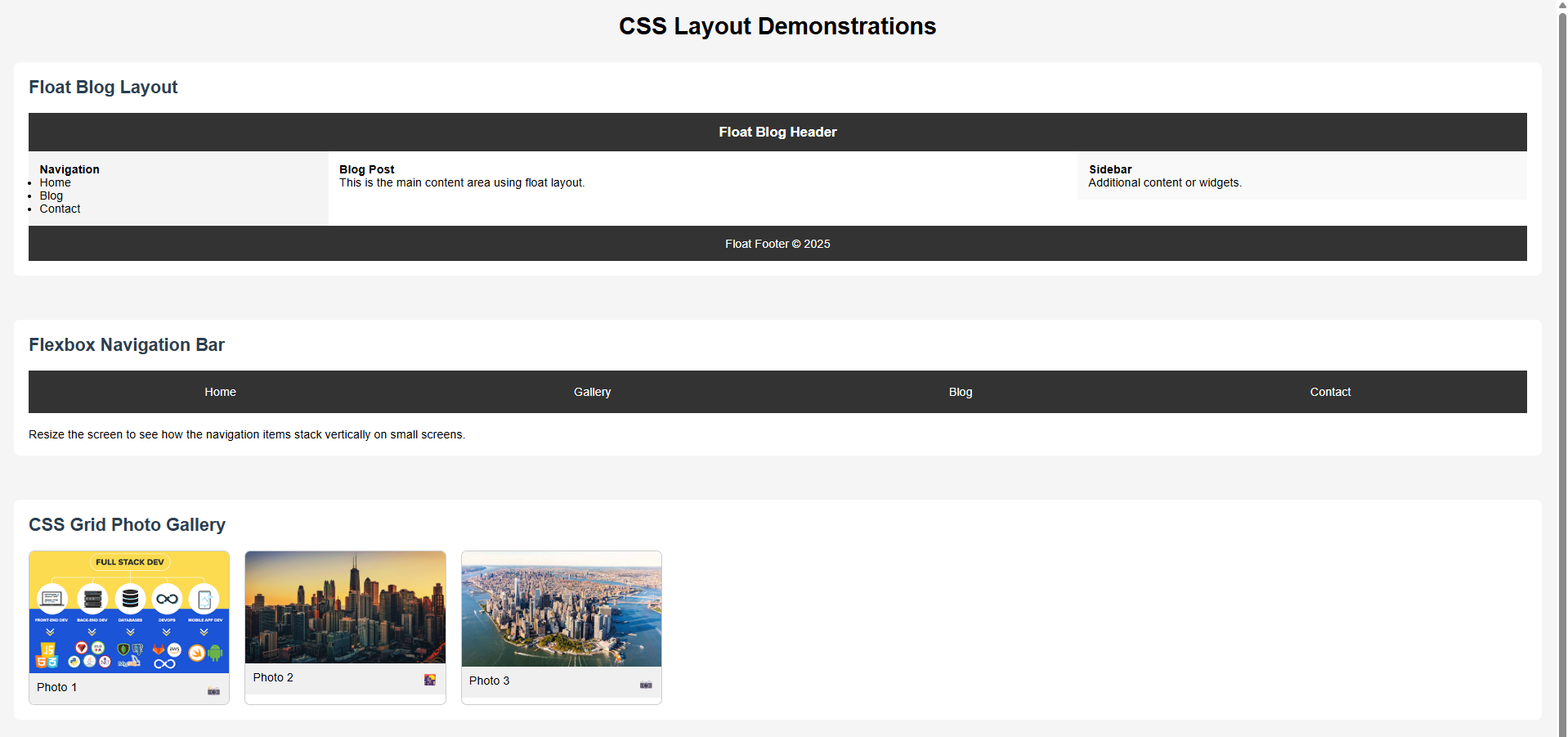
    </div>

  </section>

</body>

</html>

Output:



**Lab 7: Text and Font Properties**

* **Objective:** Control text appearance and typography.
* **Exercise:**
  + Create an HTML page with various headings and paragraphs.
  + Experiment with font-family, font-size, font-weight, font-style.
  + Apply text-align, text-decoration, text-transform.
  + Demonstrate letter-spacing and word-spacing.
  + Use Google Fonts to integrate a custom font into your page.

Code:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Typography and Text Styling</title>

  <!-- Google Font: Poppins -->

  <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;600&display=swap" rel="stylesheet">

  <style>

    body {

      font-family: 'Poppins', sans-serif;

      padding: 30px;

      line-height: 1.6;

      background-color: #f9f9f9;

      color: #333;

    }

    h1 {

      font-size: 36px;

      font-weight: 600;

      text-align: center;

      text-transform: uppercase;

      margin-bottom: 20px;

    }

    h2 {

      font-size: 28px;

      font-style: italic;

      color: #2c3e50;

      text-decoration: underline;

      margin-top: 40px;

    }

    p.normal {

      font-size: 16px;

      text-align: justify;

      font-weight: normal;

    }

    p.bold {

      font-weight: bold;

      font-size: 18px;

      text-align: left;

    }

    p.italic {

      font-style: italic;

      text-align: right;

      color: #000000;

    }

    p.spacing {

      letter-spacing: 2px;

      word-spacing: 10px;

      font-size: 18px;

      text-align: center;

    }

    .decorated {

      text-decoration: line-through;

      color: red;

      font-size: 16px;

    }

    .transformed {

      text-transform: capitalize;

      font-size: 18px;

      color: #444;

    }

    hr {

      margin: 40px 0;

      border: none;

      border-top: 2px dashed #ccc;

    }

  </style>

</head>

<body>

<div>

  <h1>Typography Demo</h1>

  <h2>Font Styles</h2>

  <p class="normal">This is a normal paragraph using the Poppins Google Font.</p>

  <p class="bold">This paragraph is bold and slightly larger.</p>

  <p class="italic">This one is italic and right-aligned.</p>

</div>

  <hr>

<div>

  <h2>Text Appearance</h2>

  <p class="spacing">This paragraph demonstrates letter and word spacing.</p>

  <p class="decorated">This line has a strikethrough decoration.</p>

  <p class="transformed">this text uses capitalization through css.</p>

</div>

</body>

</html>

**Output:**

