INTERNSHIP REPORT

WEEK 2 DAY 5

Submitted to:	**Ali Hyder**	Submission Date:	**4TH July, 2025**
Internship Domain:	**Front Development**	Internship Name:	**ProSensia**
Student Name:	**Yasal Qamar**	Roll No.	**S25031**

CSS Display and Positioning

Objective of the Day

To understand and apply the different display and position properties in CSS to control how elements appear and are placed within a webpage layout.

Introduction to CSS

Today's session focused on two fundamental layout concepts in CSS: **Display** and **Positioning**. These properties allow developers to control the visual structure and placement of elements on a webpage.

- The display property defines how an element is rendered in the document flow (e.g., block, inline, flex, grid).
- The position property controls how elements are positioned within their parent or relative to the viewport (e.g., static, relative, absolute, fixed).

We built real-world examples like a **Bookshelf (using Position)** and a **Photo Gallery (using Display)** to solidify these concepts.

Key Concepts Learned

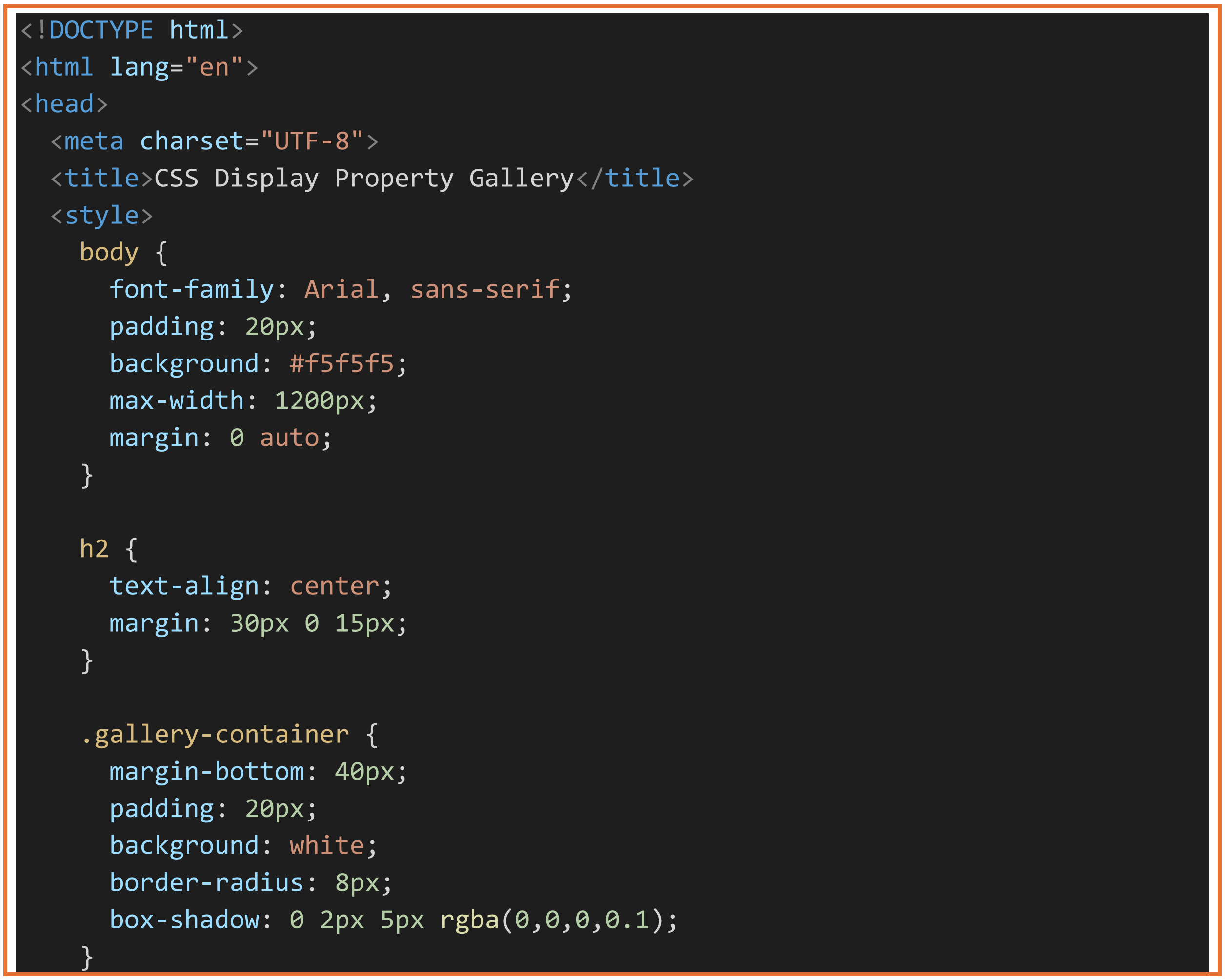
- **CSS Display Property:**
  - **block:** Makes the element start on a new line and take full width.
  - **inline:** Displays elements on the same line without breaking the flow.
  - **inline-block:** Like inline, but allows width and height to be set.

- **none:** Hides the element completely.
- **flex:** Makes the container a flex container for dynamic layouts.

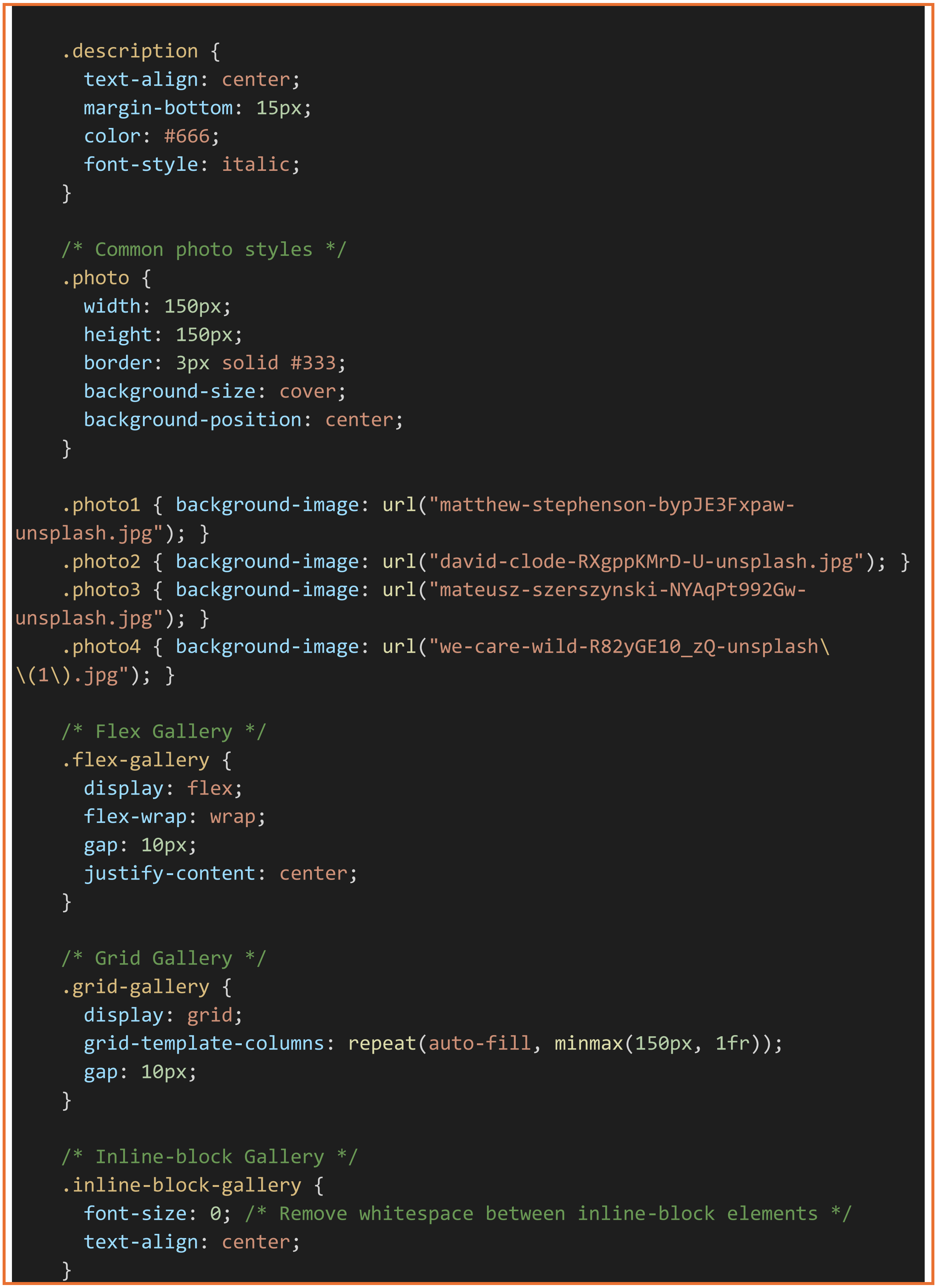
- **CSS Position Property:**

Property	Description
static	Default position. Elements follow the normal flow of the page.
relative	Shifts element relative to its original position.
absolute	Positions the element relative to its nearest positioned ancestor.
fixed	Fixes the element to the viewport. It stays in place even when scrolling.

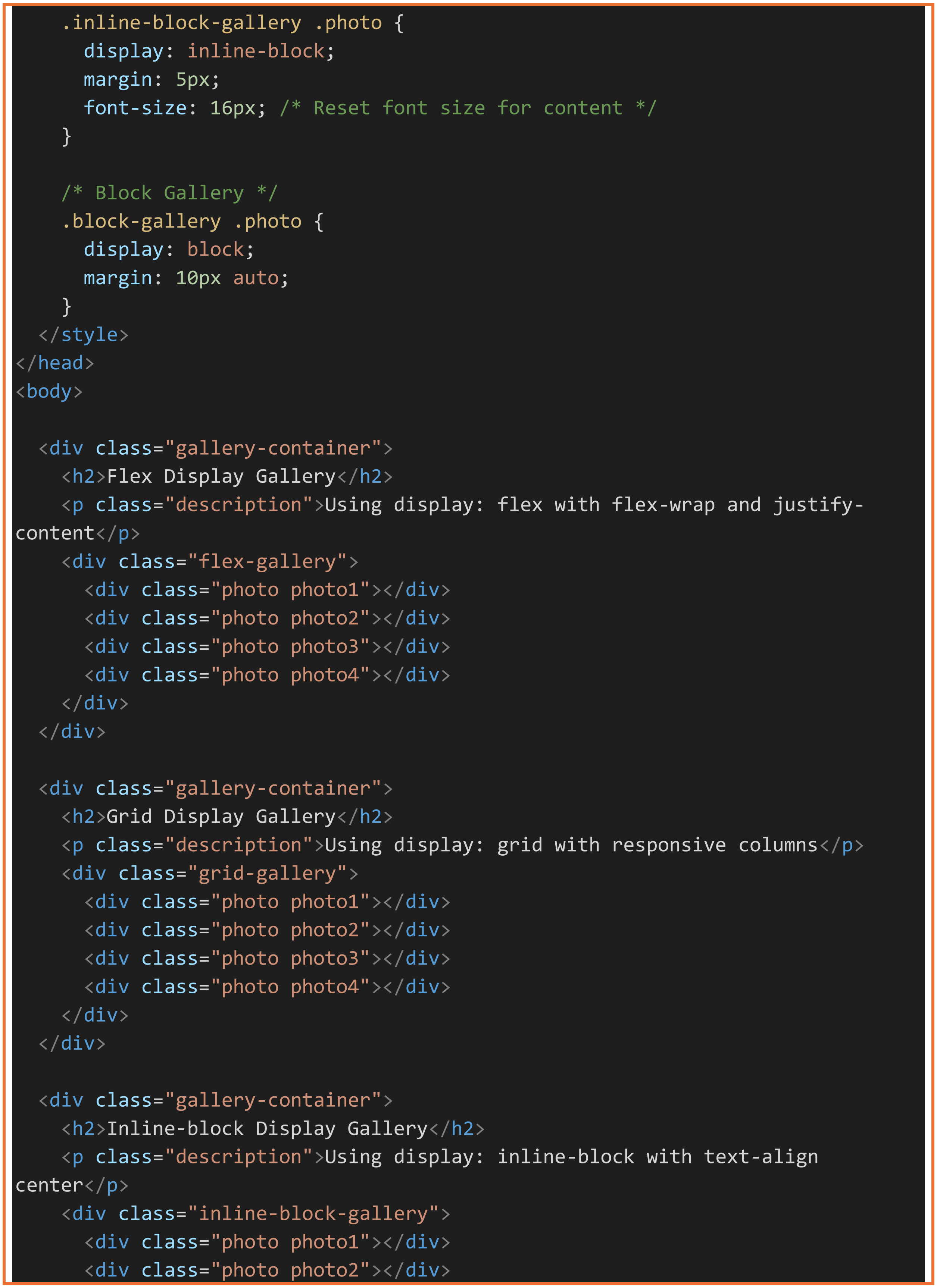
1. Example of CSS Display:



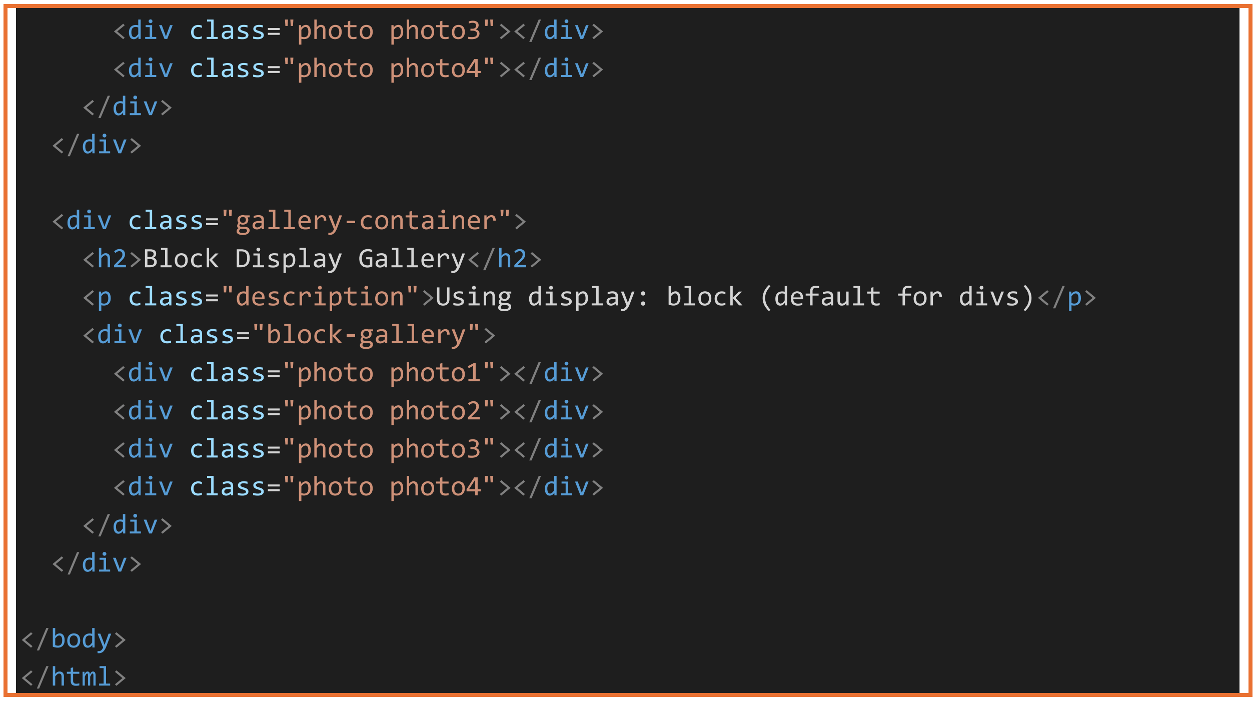
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>CSS Display Property Gallery</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      padding: 20px;
      background: #f5f5f5;
      max-width: 1200px;
      margin: 0 auto;
    }

    h2 {
      text-align: center;
      margin: 30px 0 15px;
    }

    .gallery-container {
      margin-bottom: 40px;
      padding: 20px;
      background: white;
      border-radius: 8px;
      box-shadow: 0 2px 5px rgba(0,0,0,0.1);
    }
```



```
    .description {
      text-align: center;
      margin-bottom: 15px;
      color: #666;
      font-style: italic;
    }

    /* Common photo styles */
    .photo {
      width: 150px;
      height: 150px;
      border: 3px solid #333;
      background-size: cover;
      background-position: center;
    }

    .photo1 { background-image: url("matthew-stephenson-bypJE3Fxpaw-
unsplash.jpg"); }
    .photo2 { background-image: url("david-clode-RXgppKMrD-U-unsplash.jpg"); }
    .photo3 { background-image: url("mateusz-szerszynski-NYAqPt992Gw-
unsplash.jpg"); }
    .photo4 { background-image: url("we-care-wild-R82yGE10_zQ-unsplash\
\(1\).jpg"); }

    /* Flex Gallery */
    .flex-gallery {
      display: flex;
      flex-wrap: wrap;
      gap: 10px;
      justify-content: center;
    }

    /* Grid Gallery */
    .grid-gallery {
      display: grid;
      grid-template-columns: repeat(auto-fill, minmax(150px, 1fr));
      gap: 10px;
    }

    /* Inline-block Gallery */
    .inline-block-gallery {
      font-size: 0; /* Remove whitespace between inline-block elements */
      text-align: center;
    }
```

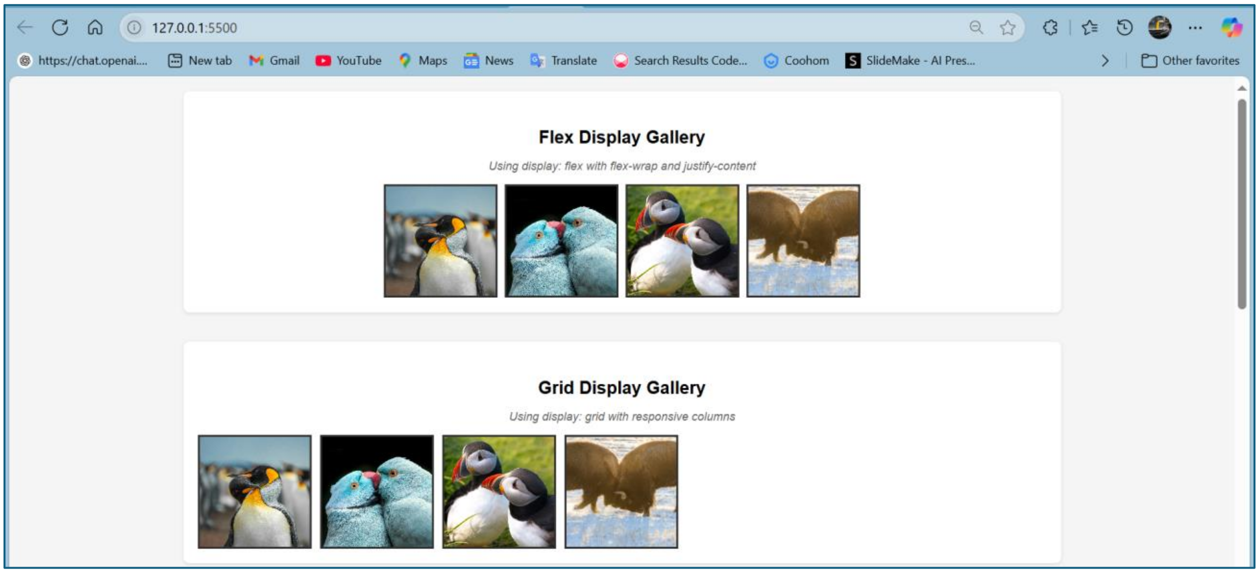


```
    .inline-block-gallery .photo {
      display: inline-block;
      margin: 5px;
      font-size: 16px; /* Reset font size for content */
    }

    /* Block Gallery */
    .block-gallery .photo {
      display: block;
      margin: 10px auto;
    }
  </style>
</head>
<body>

  <div class="gallery-container">
    <h2>Flex Display Gallery</h2>
    <p class="description">Using display: flex with flex-wrap and justify-content</p>
    <div class="flex-gallery">
      <div class="photo photo1"></div>
      <div class="photo photo2"></div>
      <div class="photo photo3"></div>
      <div class="photo photo4"></div>
    </div>
  </div>

  <div class="gallery-container">
    <h2>Grid Display Gallery</h2>
    <p class="description">Using display: grid with responsive columns</p>
    <div class="grid-gallery">
      <div class="photo photo1"></div>
      <div class="photo photo2"></div>
      <div class="photo photo3"></div>
      <div class="photo photo4"></div>
    </div>
  </div>

  <div class="gallery-container">
    <h2>Inline-block Display Gallery</h2>
    <p class="description">Using display: inline-block with text-align center</p>
    <div class="inline-block-gallery">
      <div class="photo photo1"></div>
      <div class="photo photo2"></div>
```



```
      <div class="photo photo3"></div>
      <div class="photo photo4"></div>
    </div>
  </div>

  <div class="gallery-container">
    <h2>Block Display Gallery</h2>
    <p class="description">Using display: block (default for divs)</p>
    <div class="block-gallery">
      <div class="photo photo1"></div>
      <div class="photo photo2"></div>
      <div class="photo photo3"></div>
      <div class="photo photo4"></div>
    </div>
  </div>

</body>
</html>
```

OUTPUT:

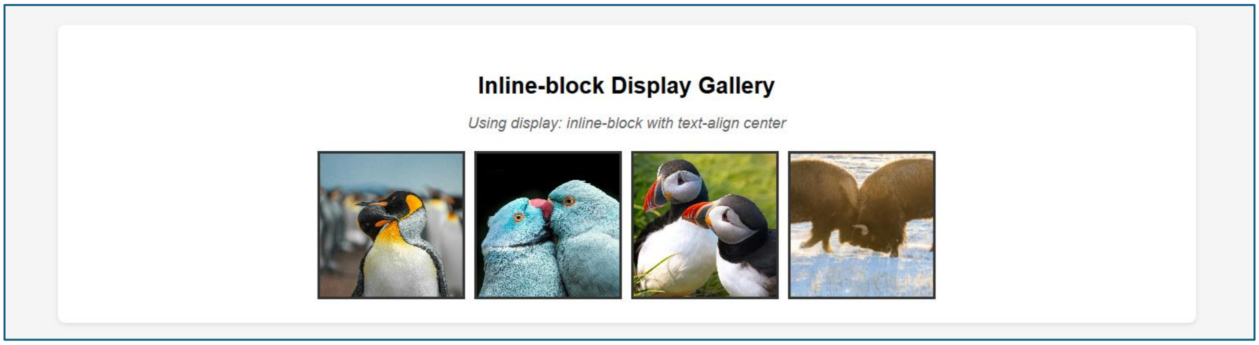


Flex Display Gallery

Using display: flex with flex-wrap and justify-content

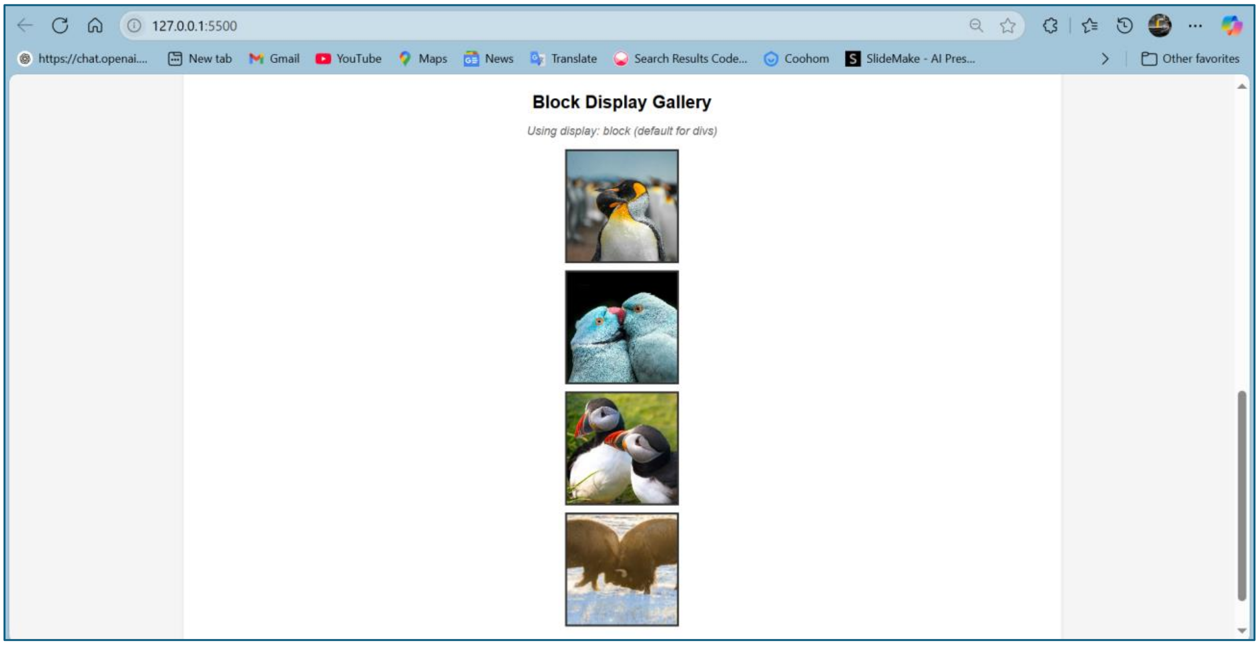
Grid Display Gallery

Using display: grid with responsive columns



Inline-block Display Gallery

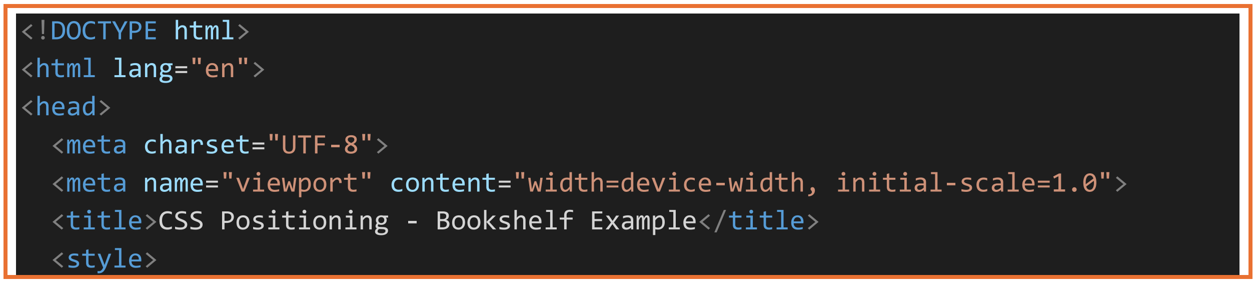
Using display: inline-block with text-align center



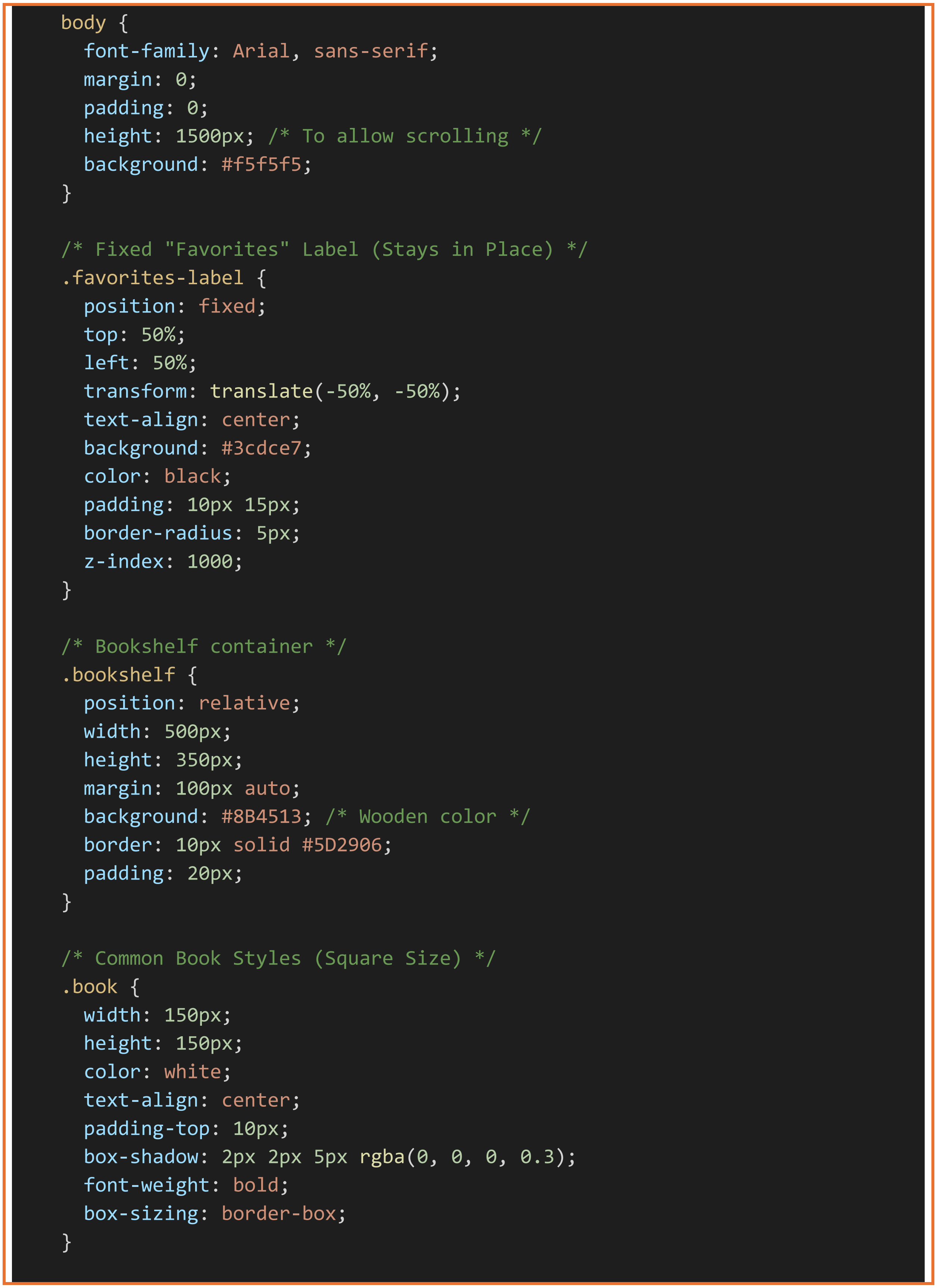
Block Display Gallery

Using display: block (default for divs)

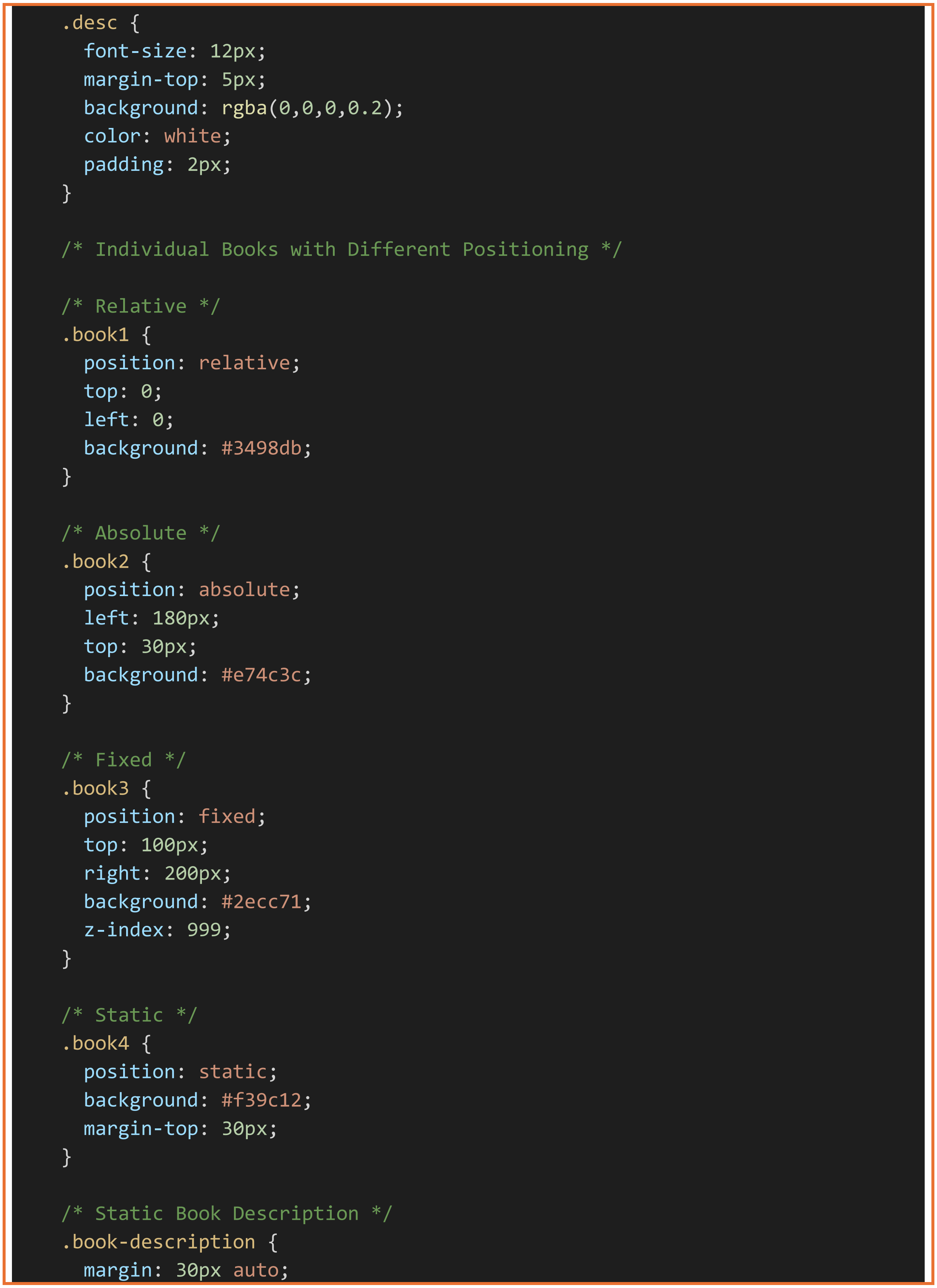
2. Example of CSS Positioning:



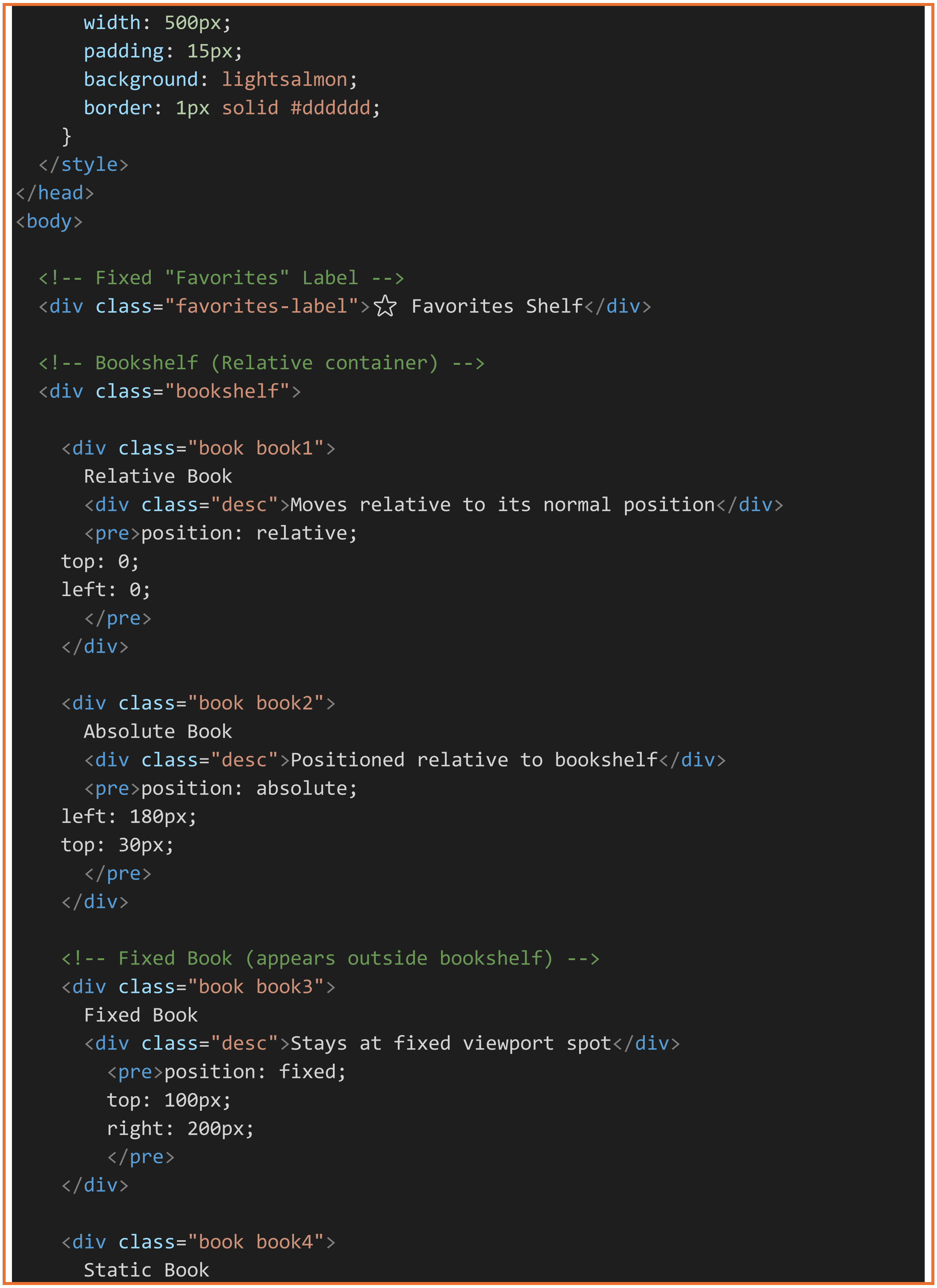
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CSS Positioning - Bookshelf Example</title>
  <style>
```



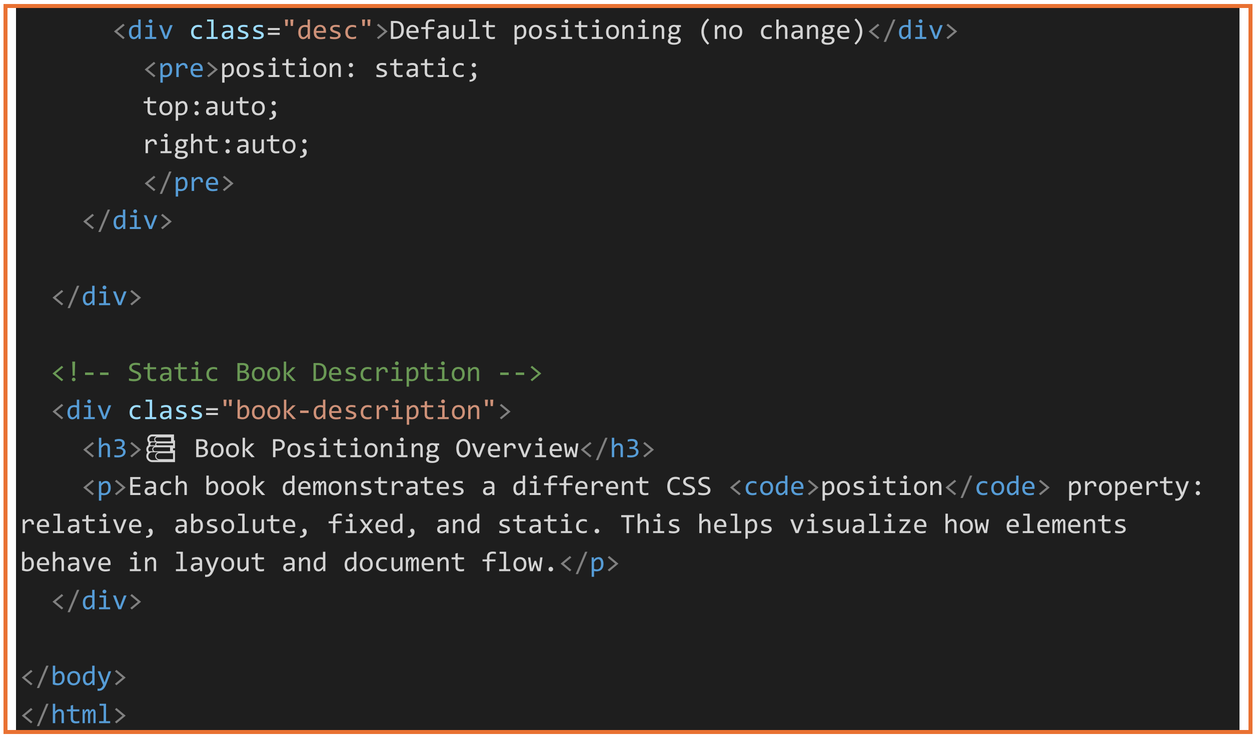
```css
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  height: 1500px; /* To allow scrolling */
  background: #f5f5f5;
}

/* Fixed "Favorites" Label (Stays in Place) */
.favorites-label {
  position: fixed;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  text-align: center;
  background: #3cdce7;
  color: black;
  padding: 10px 15px;
  border-radius: 5px;
  z-index: 1000;
}

/* Bookshelf container */
.bookshelf {
  position: relative;
  width: 500px;
  height: 350px;
  margin: 100px auto;
  background: #8B4513; /* Wooden color */
  border: 10px solid #5D2906;
  padding: 20px;
}

/* Common Book Styles (Square Size) */
.book {
  width: 150px;
  height: 150px;
  color: white;
  text-align: center;
  padding-top: 10px;
  box-shadow: 2px 2px 5px rgba(0, 0, 0, 0.3);
  font-weight: bold;
  box-sizing: border-box;
}
```



```css
.desc {
  font-size: 12px;
  margin-top: 5px;
  background: rgba(0,0,0,0.2);
  color: white;
  padding: 2px;
}

/* Individual Books with Different Positioning */

/* Relative */
.book1 {
  position: relative;
  top: 0;
  left: 0;
  background: #3498db;
}

/* Absolute */
.book2 {
  position: absolute;
  left: 180px;
  top: 30px;
  background: #e74c3c;
}

/* Fixed */
.book3 {
  position: fixed;
  top: 100px;
  right: 200px;
  background: #2ecc71;
  z-index: 999;
}

/* Static */
.book4 {
  position: static;
  background: #f39c12;
  margin-top: 30px;
}

/* Static Book Description */
.book-description {
  margin: 30px auto;
```

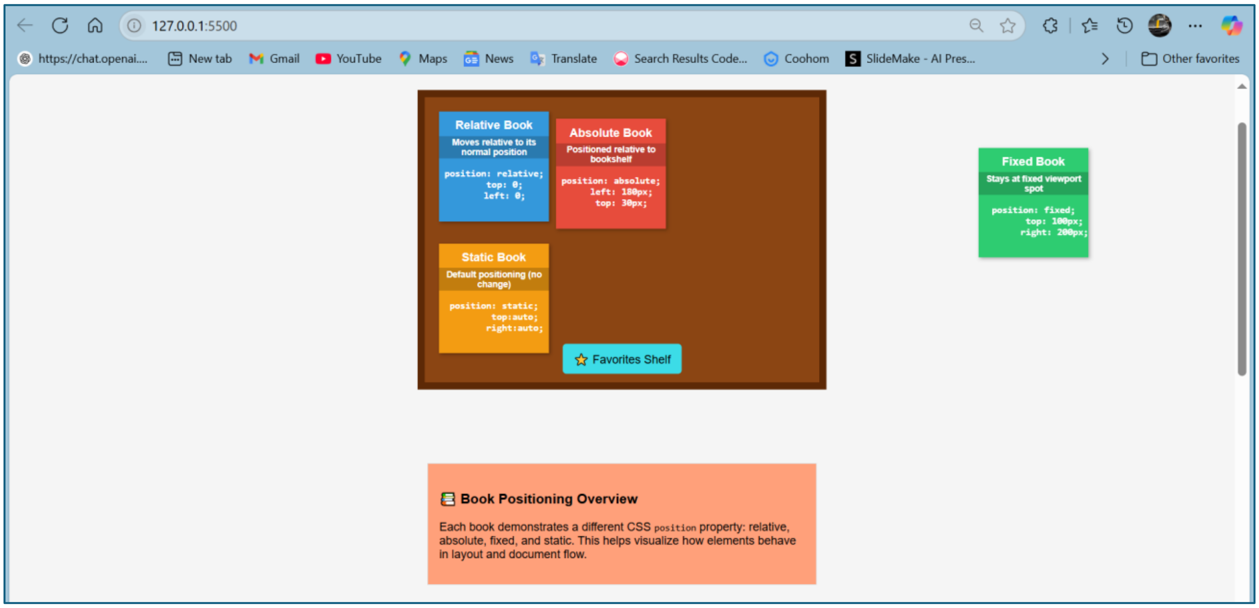


```
      width: 500px;
      padding: 15px;
      background: lightsalmon;
      border: 1px solid #dddddd;
    }
  </style>
</head>
<body>

  <!-- Fixed "Favorites" Label -->
  <div class="favorites-label">☆ Favorites Shelf</div>

  <!-- Bookshelf (Relative container) -->
  <div class="bookshelf">

    <div class="book book1">
      Relative Book
      <div class="desc">Moves relative to its normal position</div>
      <pre>position: relative;
    top: 0;
    left: 0;
      </pre>
    </div>

    <div class="book book2">
      Absolute Book
      <div class="desc">Positioned relative to bookshelf</div>
      <pre>position: absolute;
    left: 180px;
    top: 30px;
      </pre>
    </div>

    <!-- Fixed Book (appears outside bookshelf) -->
    <div class="book book3">
      Fixed Book
      <div class="desc">Stays at fixed viewport spot</div>
        <pre>position: fixed;
        top: 100px;
        right: 200px;
        </pre>
    </div>

    <div class="book book4">
      Static Book
```



```
      <div class="desc">Default positioning (no change)</div>
        <pre>position: static;
        top:auto;
        right:auto;
        </pre>
    </div>

  </div>

  <!-- Static Book Description -->
  <div class="book-description">
    <h3>📚 Book Positioning Overview</h3>
    <p>Each book demonstrates a different CSS <code>position</code> property:
relative, absolute, fixed, and static. This helps visualize how elements
behave in layout and document flow.</p>
  </div>

</body>
</html>
```

OUTPUT:



Relative Book

Moves relative to its normal position

```
position: relative;
      top: 0;
      left: 0;
```

Absolute Book

Positioned relative to bookshelf

```
position: absolute;
      left: 180px;
       top: 30px;
```

Static Book

Default positioning (no change)

```
position: static;
        top:auto;
      right:auto;
```

⭐ Favorites Shelf

Fixed Book

Stays at fixed viewport spot

```
position: fixed;
      top: 100px;
    right: 200px;
```

📚 Book Positioning Overview

Each book demonstrates a different CSS `position` property: relative, absolute, fixed, and static. This helps visualize how elements behave in layout and document flow.

Conclusion:

In summary, I have learned how to effectively use layout-controlling properties such as display and position in CSS. These tools are essential for organizing content, aligning elements, and building responsive web interfaces. By experimenting with real-world examples like a bookshelf and image gallery in VS Code, I understood how different display types (block, inline, flex, grid) and positioning methods (static, relative, absolute, fixed) affect page structure and element behavior. This hands-on experience improved my ability to combine design and layout logic, which is key to creating structured, user-friendly front-end applications.