Experiment 2

Title

Basics of styling in CSS and layout techniques like Flexbox and Grid

Aim

To apply CSS3 styling techniques and implement Flexbox and Grid layouts for creating visually appealing and well-structured web pages.

Software

Code Editor (VSCode/Notepad++)
Web Browser (Firefox/Chrome)

Theory

CSS is composed of **selectors** and **declarations**.
Basic Structure:

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

- Selector: The HTML element you want to style.
- Property: The aspect of the element you want to change (e.g., color, font-size).
- Value: The value assigned to the property. Example: p { color: red; text-align: center; }

We use different kinds of selectors in different scenarios.

- Type selector: Select all instances of a specific HTML tag

```
p {
   color: blue;
}
```

- Class selector: Select all tags with a particular class
  Html tag: `<p class="intro">Welcome to CSS!</p>`

```
.intro {
   font-weight: bold;
}
```

- Id selector: Select a tag using the id attribute.

  ```
  <p id="main-paragraph">This is the main paragraph.</p>
          #main-paragraph {
              font-size: 20px;
          }
  ```
- Universal selector: Selects all elements

  ```
          * {
              margin: 0px;
              padding: 0px;
          }
  ```
- Grouping selector: Apply the same style to multiple selectors.

  ```
          h1, h2, h3 {
              color: navy;
          }
  ```
- Descendant selector: Selects elements nested within others

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

Before styling, you must understand the three ways to include CSS in a web page.

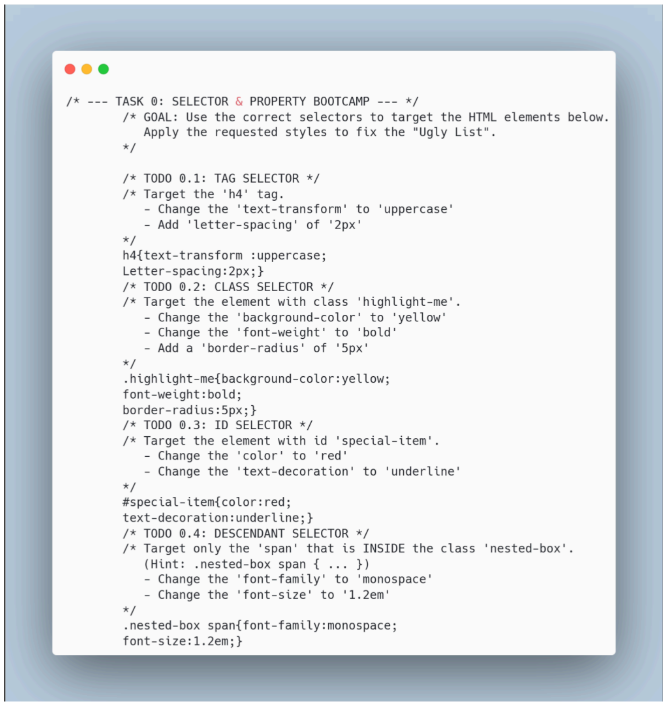
- *Inline CSS:* Apply a `style` attribute directly to an individual HTML element (e.g., `<h2 style="color: blue;">`).
- *Internal CSS:* Use the `<style>` tag within the `<head>` section of the HTML document to define rules for the entire page.
- *External CSS:* Create a separate `.css` file and link it to the HTML document using the `<link>` tag in the `<head>`section.

Conclusion

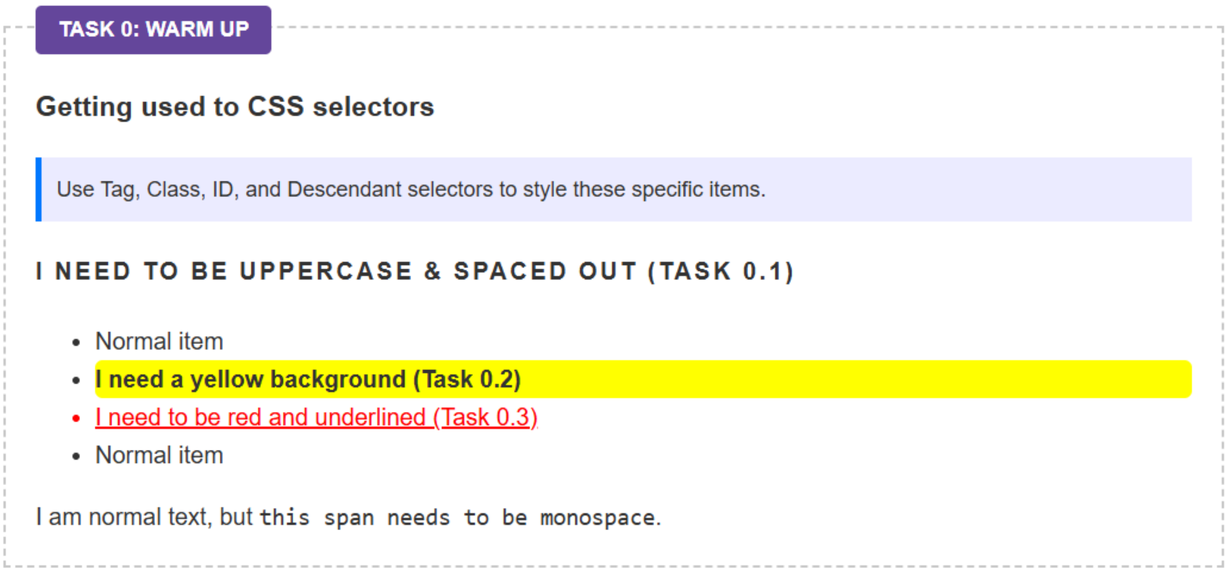
We learnt two types of layouts: Flexbox and Grid. Flexbox layout is suited for the components of an application, and small-scale layouts, while the Grid layout is intended for larger scale layouts.

Tasks

0.



```
/* --- TASK 0: SELECTOR & PROPERTY BOOTCAMP --- */
        /* GOAL: Use the correct selectors to target the HTML elements below.
           Apply the requested styles to fix the "Ugly List".
        */

        /* TODO 0.1: TAG SELECTOR */
        /* Target the 'h4' tag.
           - Change the 'text-transform' to 'uppercase'
           - Add 'letter-spacing' of '2px'
        */
        h4{text-transform :uppercase;
        Letter-spacing:2px;}
        /* TODO 0.2: CLASS SELECTOR */
        /* Target the element with class 'highlight-me'.
           - Change the 'background-color' to 'yellow'
           - Change the 'font-weight' to 'bold'
           - Add a 'border-radius' of '5px'
        */
        .highlight-me{background-color:yellow;
        font-weight:bold;
        border-radius:5px;}
        /* TODO 0.3: ID SELECTOR */
        /* Target the element with id 'special-item'.
           - Change the 'color' to 'red'
           - Change the 'text-decoration' to 'underline'
        */
        #special-item{color:red;
        text-decoration:underline;}
        /* TODO 0.4: DESCENDANT SELECTOR */
        /* Target only the 'span' that is INSIDE the class 'nested-box'.
           (Hint: .nested-box span { ... })
           - Change the 'font-family' to 'monospace'
           - Change the 'font-size' to '1.2em'
        */
        .nested-box span{font-family:monospace;
        font-size:1.2em;}
```



TASK 0: WARM UP

Getting used to CSS selectors

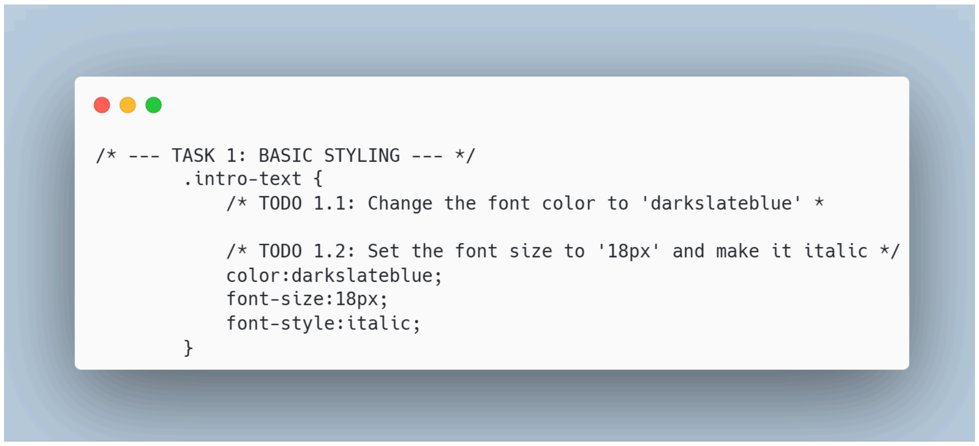
Use Tag, Class, ID, and Descendant selectors to style these specific items.

I NEED TO BE UPPERCASE & SPACED OUT (TASK 0.1)

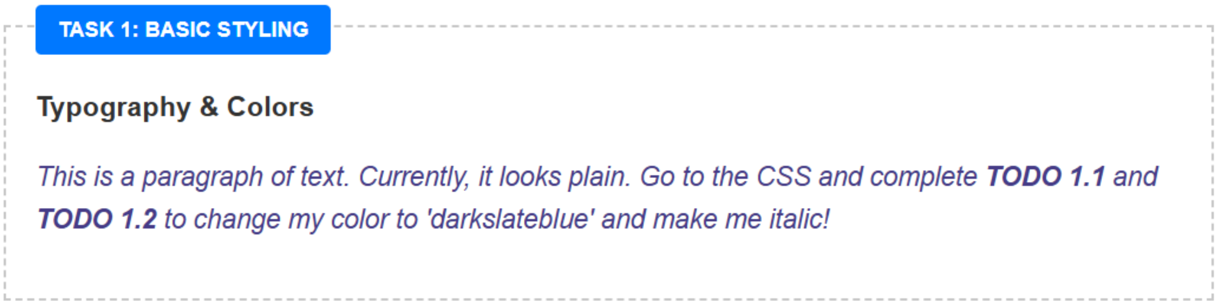
- Normal item
- **I need a yellow background (Task 0.2)**
- I need to be red and underlined (Task 0.3)
- Normal item

I am normal text, but `this span needs to be monospace`.

1.



```
/* --- TASK 1: BASIC STYLING --- */
        .intro-text {
            /* TODO 1.1: Change the font color to 'darkslateblue' *

            /* TODO 1.2: Set the font size to '18px' and make it italic */
            color:darkslateblue;
            font-size:18px;
            font-style:italic;
        }
```

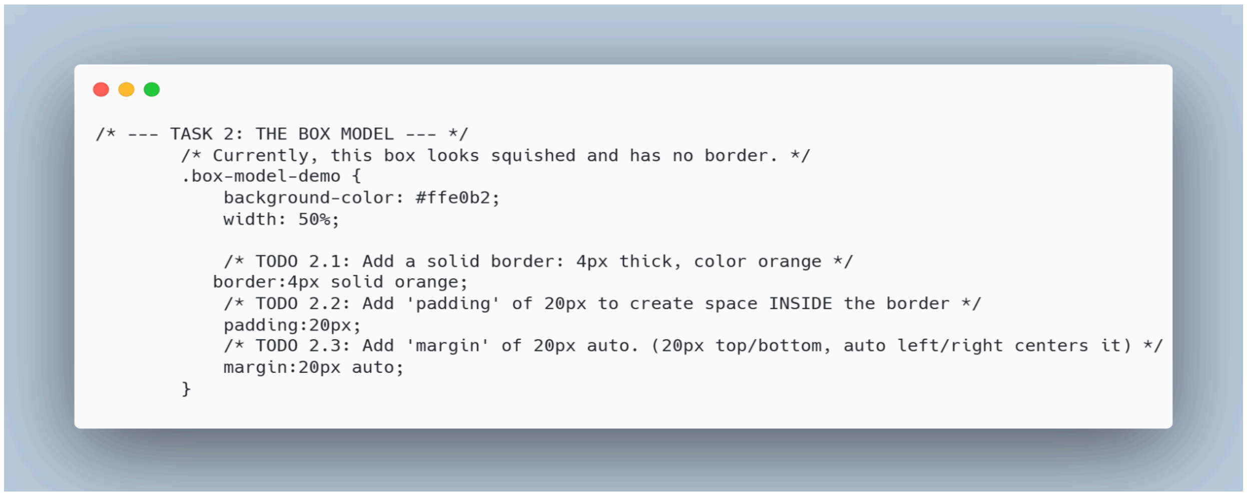


TASK 1: BASIC STYLING

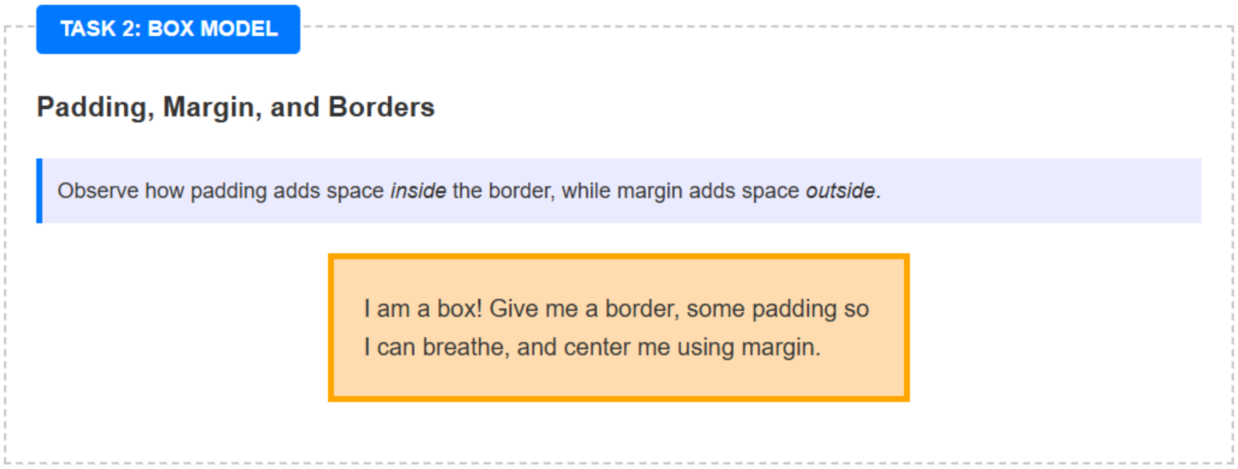
Typography & Colors

This is a paragraph of text. Currently, it looks plain. Go to the CSS and complete ***TODO 1.1*** *and* ***TODO 1.2*** *to change my color to 'darkslateblue' and make me italic!*

2.



```
/* --- TASK 2: THE BOX MODEL --- */
        /* Currently, this box looks squished and has no border. */
        .box-model-demo {
            background-color: #ffe0b2;
            width: 50%;

            /* TODO 2.1: Add a solid border: 4px thick, color orange */
           border:4px solid orange;
            /* TODO 2.2: Add 'padding' of 20px to create space INSIDE the border */
            padding:20px;
            /* TODO 2.3: Add 'margin' of 20px auto. (20px top/bottom, auto left/right centers it) */
            margin:20px auto;
        }
```

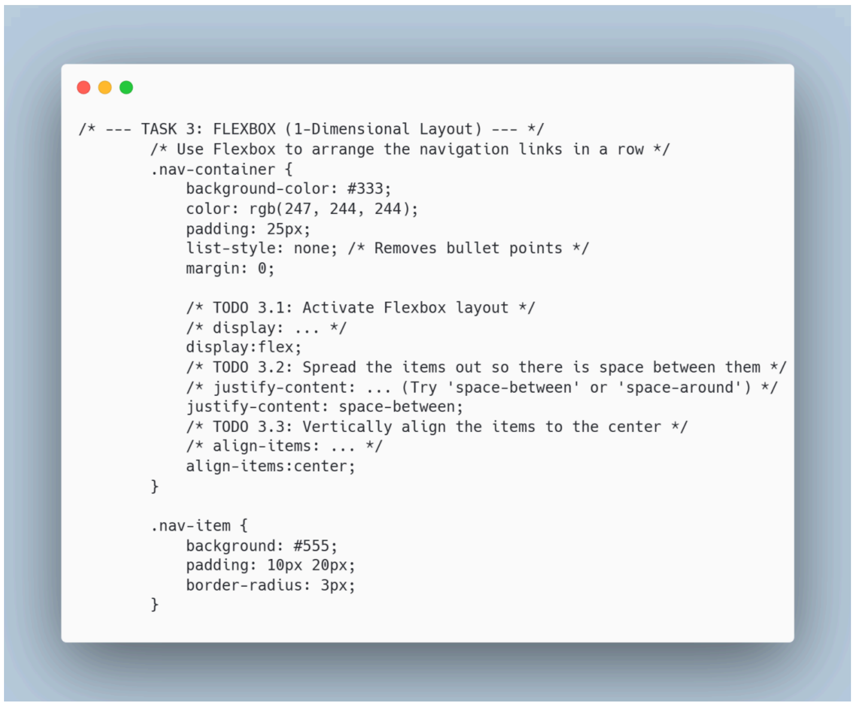

TASK 2: BOX MODEL

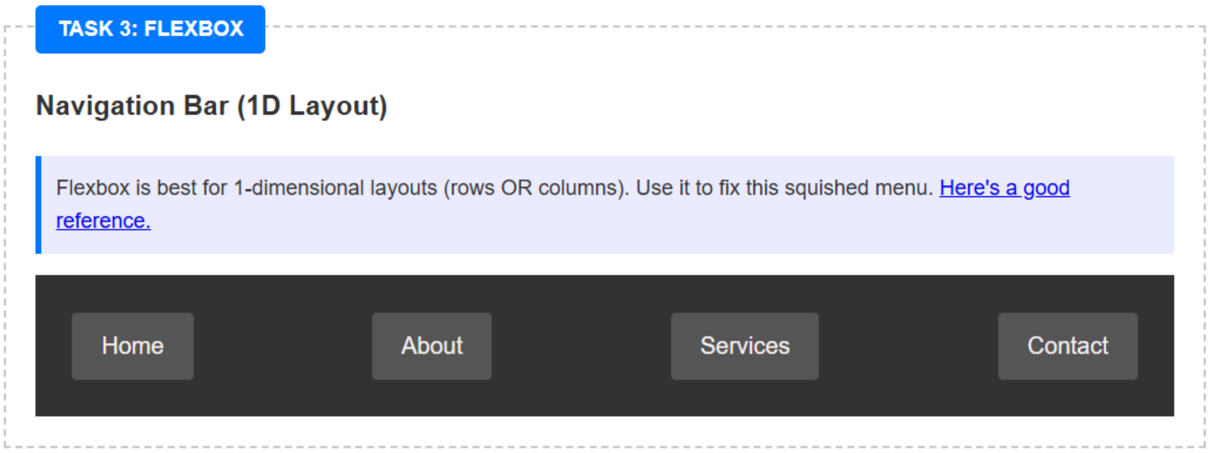
Padding, Margin, and Borders

Observe how padding adds space *inside* the border, while margin adds space *outside*.

I am a box! Give me a border, some padding so I can breathe, and center me using margin.

3.

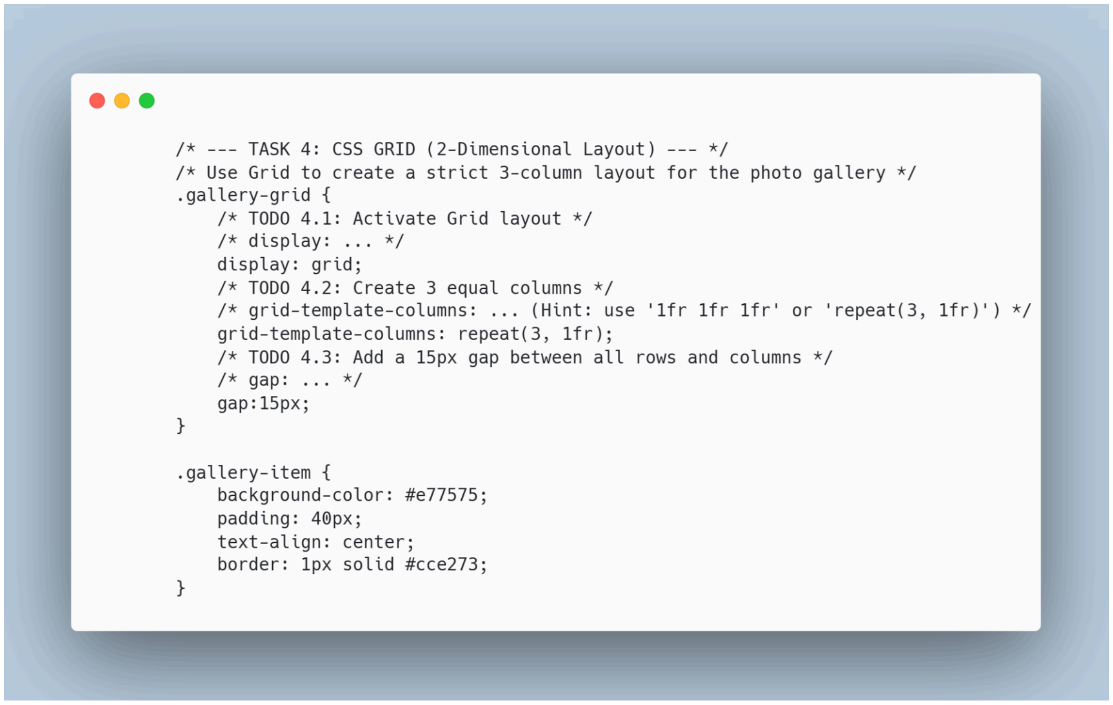

```
/* --- TASK 3: FLEXBOX (1-Dimensional Layout) --- */
        /* Use Flexbox to arrange the navigation links in a row */
        .nav-container {
            background-color: #333;
            color: rgb(247, 244, 244);
            padding: 25px;
            list-style: none; /* Removes bullet points */
            margin: 0;

            /* TODO 3.1: Activate Flexbox layout */
            /* display: ... */
            display:flex;
            /* TODO 3.2: Spread the items out so there is space between them */
            /* justify-content: ... (Try 'space-between' or 'space-around') */
            justify-content: space-between;
            /* TODO 3.3: Vertically align the items to the center */
            /* align-items: ... */
            align-items:center;
        }

        .nav-item {
            background: #555;
            padding: 10px 20px;
            border-radius: 3px;
        }
```


TASK 3: FLEXBOX

Navigation Bar (1D Layout)

Flexbox is best for 1-dimensional layouts (rows OR columns). Use it to fix this squished menu. Here's a good reference.

4.


```
/* --- TASK 4: CSS GRID (2-Dimensional Layout) --- */
/* Use Grid to create a strict 3-column layout for the photo gallery */
.gallery-grid {
    /* TODO 4.1: Activate Grid layout */
    /* display: ... */
    display: grid;
    /* TODO 4.2: Create 3 equal columns */
    /* grid-template-columns: ... (Hint: use '1fr 1fr 1fr' or 'repeat(3, 1fr)') */
    grid-template-columns: repeat(3, 1fr);
    /* TODO 4.3: Add a 15px gap between all rows and columns */
    /* gap: ... */
    gap:15px;
}

.gallery-item {
    background-color: #e77575;
    padding: 40px;
    text-align: center;
    border: 1px solid #cce273;
}
```

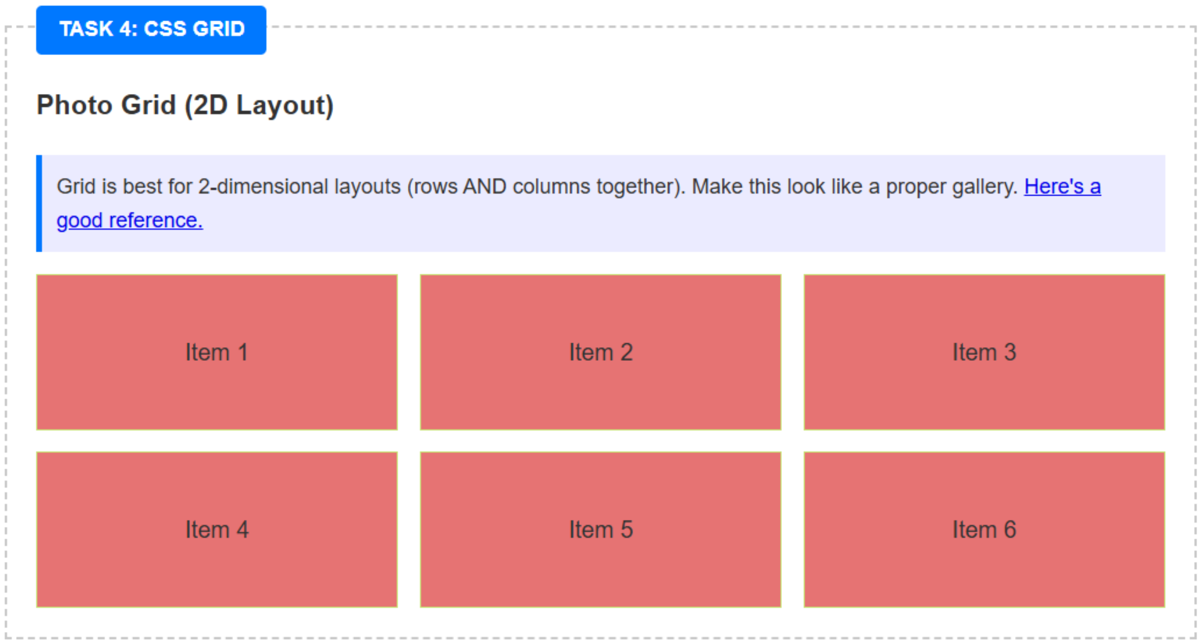
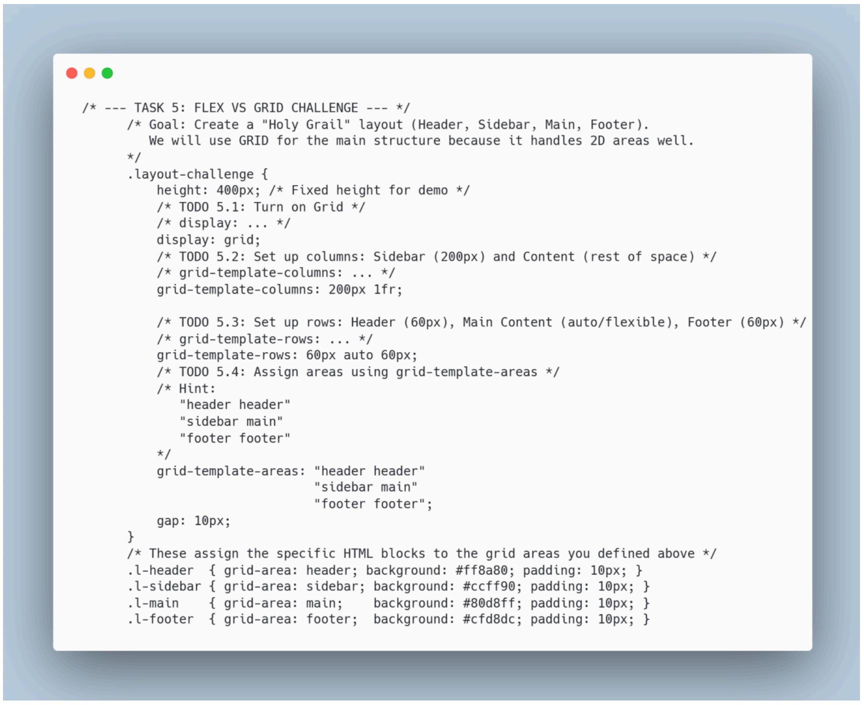

TASK 4: CSS GRID

Photo Grid (2D Layout)

Grid is best for 2-dimensional layouts (rows AND columns together). Make this look like a proper gallery. Here's a good reference.

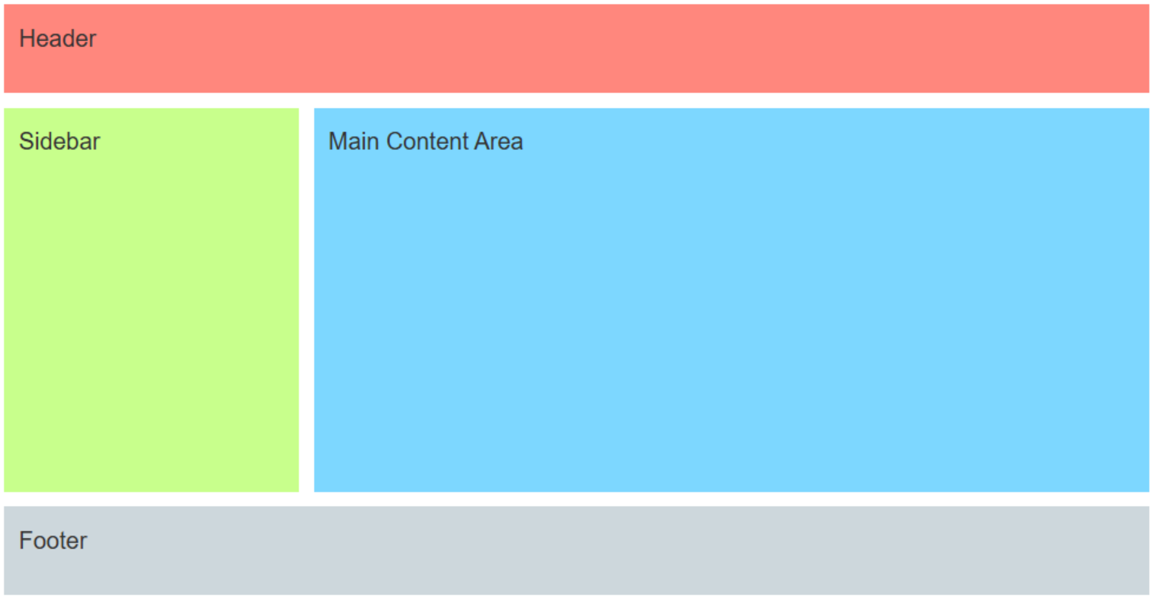
5.


```
/* --- TASK 5: FLEX VS GRID CHALLENGE --- */
      /* Goal: Create a "Holy Grail" layout (Header, Sidebar, Main, Footer).
         We will use GRID for the main structure because it handles 2D areas well.
      */
      .layout-challenge {
          height: 400px; /* Fixed height for demo */
          /* TODO 5.1: Turn on Grid */
          /* display: ... */
          display: grid;
          /* TODO 5.2: Set up columns: Sidebar (200px) and Content (rest of space) */
          /* grid-template-columns: ... */
          grid-template-columns: 200px 1fr;

          /* TODO 5.3: Set up rows: Header (60px), Main Content (auto/flexible), Footer (60px) */
          /* grid-template-rows: ... */
          grid-template-rows: 60px auto 60px;
          /* TODO 5.4: Assign areas using grid-template-areas */
          /* Hint:
             "header header"
             "sidebar main"
             "footer footer"
          */
          grid-template-areas: "header header"
                               "sidebar main"
                               "footer footer";
          gap: 10px;
      }
      /* These assign the specific HTML blocks to the grid areas you defined above */
      .l-header  { grid-area: header; background: #ff8a80; padding: 10px; }
      .l-sidebar { grid-area: sidebar; background: #ccff90; padding: 10px; }
      .l-main    { grid-area: main;    background: #80d8ff; padding: 10px; }
      .l-footer  { grid-area: footer;  background: #cfd8dc; padding: 10px; }
```

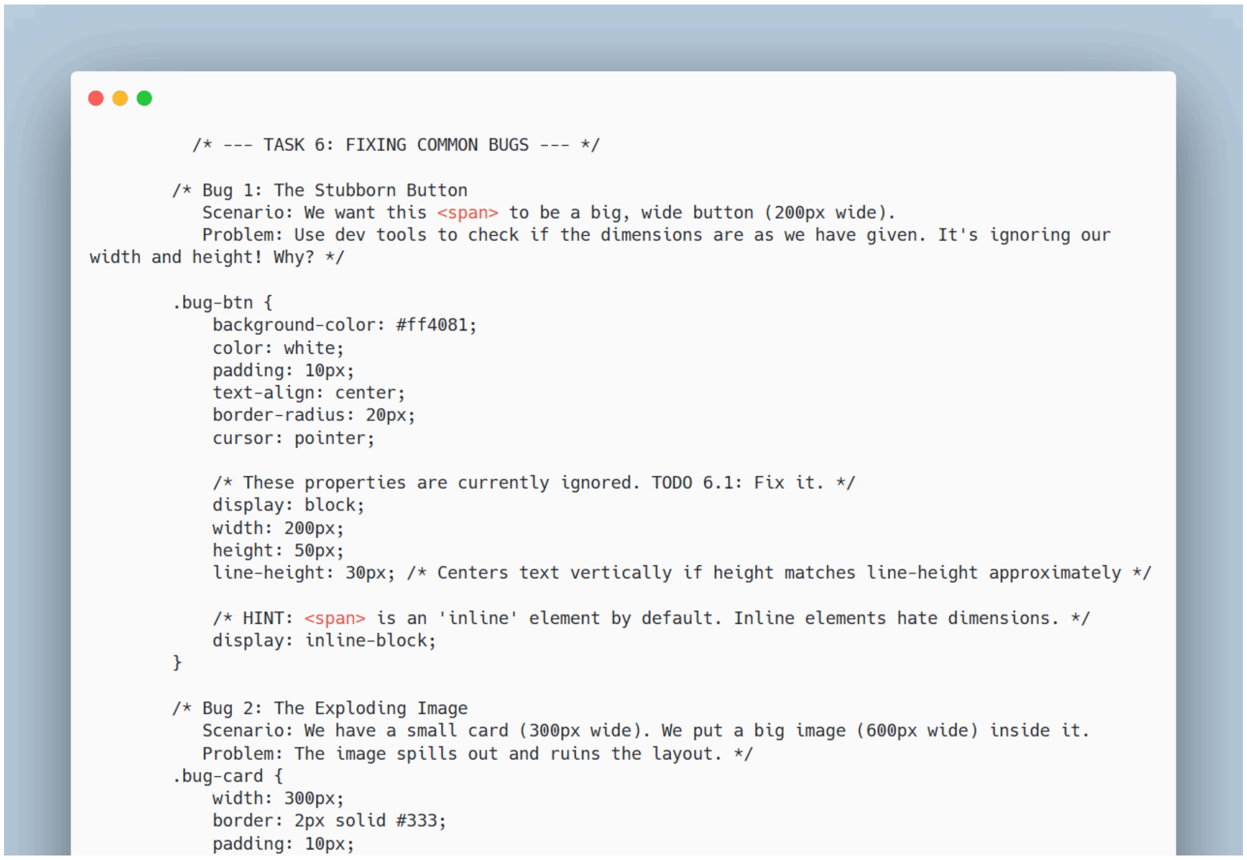
TASK 5: LAYOUT CHALLENGE

The Structure

Advanced: We want a standard website layout. This layout has long been called the 'Holy Grail' in web design because people have been searching for the best solution to achieve it for a while before Flexbox and grid (there were only some clumsy and time-consuming workarounds earlier). The HTML is already tagged with classes (l-header, l-sidebar, etc.). Your job is to define the Grid on the parent container.

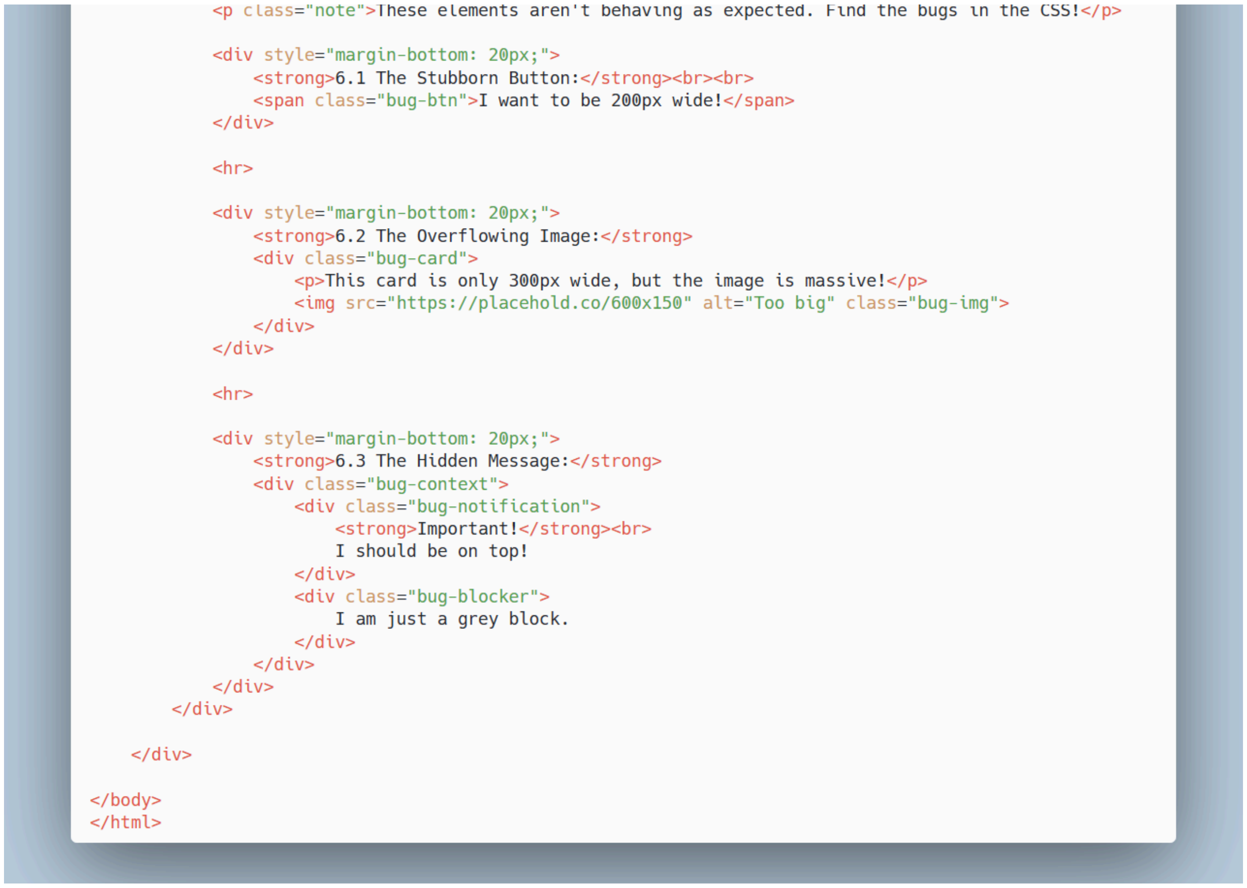


6.



```
          /* --- TASK 6: FIXING COMMON BUGS --- */

        /* Bug 1: The Stubborn Button
           Scenario: We want this <span> to be a big, wide button (200px wide).
           Problem: Use dev tools to check if the dimensions are as we have given. It's ignoring our
width and height! Why? */

        .bug-btn {
            background-color: #ff4081;
            color: white;
            padding: 10px;
            text-align: center;
            border-radius: 20px;
            cursor: pointer;

            /* These properties are currently ignored. TODO 6.1: Fix it. */
            display: block;
            width: 200px;
            height: 50px;
            line-height: 30px; /* Centers text vertically if height matches line-height approximately */

            /* HINT: <span> is an 'inline' element by default. Inline elements hate dimensions. */
            display: inline-block;
        }

        /* Bug 2: The Exploding Image
           Scenario: We have a small card (300px wide). We put a big image (600px wide) inside it.
           Problem: The image spills out and ruins the layout. */
        .bug-card {
            width: 300px;
            border: 2px solid #333;
            padding: 10px;
            background: white;
            margin-top: 20px;
        }

        .bug-img {
            /* TODO 6.2: Ensure the image never exceeds the width of its parent container. */
            max-width: 100%;
            height: auto;
        }

        /* Bug 3: The Hidden Message
           Scenario: We have a 'notification' box that is positioned absolutely.
           Problem: It is currently hiding *behind* the grey box below it. */
        .bug-context {
            position: relative; /* Creates a positioning context */
            height: 150px;
            margin-top: 20px;
            border: 1px solid #ccc;
            background: #eee;
        }

        .bug-notification {
            position: absolute;
            top: 20px;
            left: 20px;
            background: #ffe082;
            padding: 20px;
            border: 1px solid orange;
            box-shadow: 2px 2px 5px rgba(0,0,0,0.2);

            /* TODO 6.3: Bring this element to the front */
            z-index: 10;
        }

        .bug-blocker {
            position: absolute;
            top: 40px;
            left: 40px;
            width: 200px;
            height: 80px;
            background: #90a4ae;
            /* This element is naturally stacking on top because it appears later in the HTML. */
        }

    </style>
</head>
<body>

    <div class="container">
        <header style="text-align: center; margin-bottom: 40px;">
            <h1>Lab 2: CSS3 Styling & Layout</h1>
            <p>Complete the tasks in the <code>&lt;style&gt;</code> section to fix the page.</p>
        </header>

        <div class="task-section">
            <div class="task-badge" style="background: #66479f;">Task 0: Warm Up</div>
            <h3>Getting used to CSS selectors</h3>
            <p class="note">Use Tag, Class, ID, and Descendant selectors to style these specific items.
</p>

            <h4>I need to be Uppercase & Spaced out (Task 0.1)</h4>

            <ul>
                <li>Normal item</li>
                <li class="highlight-me">I need a yellow background (Task 0.2)</li>

                <li id="special-item">I need to be red and underlined (Task 0.3)</li>

                <li>Normal item</li>
            </ul>

            <div class="nested-box">
                I am normal text, but <span>this span needs to be monospace</span>.
            </div>
        </div>
```

```html
        <div class="task-section">
            <div class="task-badge">Task 1: Basic Styling</div>
            <h3>Typography &amp; Colors</h3>
            <p class="intro-text">
                This is a paragraph of text. Currently, it looks plain.
                Go to the CSS and complete <strong>TODO 1.1</strong> and <strong>TODO 1.2</strong>
                to change my color to 'darkslateblue' and make me italic!
            </p>
        </div>

        <div class="task-section">
            <div class="task-badge">Task 2: Box Model</div>
            <h3>Padding, Margin, and Borders</h3>
            <p class="note">Observe how padding adds space <em>inside</em> the border, while margin adds
space <em>outside</em>.</p>
            <div class="box-model-demo">
                I am a box! Give me a border, some padding so I can breathe, and center me using margin.
            </div>
        </div>

        <div class="task-section">
            <div class="task-badge">Task 3: Flexbox</div>
            <h3>Navigation Bar (1D Layout)</h3>
            <p class="note">Flexbox is best for 1-dimensional layouts (rows OR columns). Use it to fix
this squished menu. <a href="https://css-tricks.com/snippets/css/a-guide-to-flexbox/"
target="_blank">Here's a good reference.</a></p>

            <ul class="nav-container">
                <li class="nav-item">Home</li>
                <li class="nav-item">About</li>
                <li class="nav-item">Services</li>
                <li class="nav-item">Contact</li>
            </ul>
        </div>

        <div class="task-section">
            <div class="task-badge">Task 4: CSS Grid</div>
            <h3>Photo Grid (2D Layout)</h3>
            <p class="note">Grid is best for 2-dimensional layouts (rows AND columns together). Make this
look like a proper gallery. <a href="https://css-tricks.com/css-grid-layout-guide/"
target="_blank">Here's a good reference.</a></p>

            <div class="gallery-grid">
                <div class="gallery-item">Item 1</div>
                <div class="gallery-item">Item 2</div>
                <div class="gallery-item">Item 3</div>
                <div class="gallery-item">Item 4</div>
                <div class="gallery-item">Item 5</div>
                <div class="gallery-item">Item 6</div>
            </div>
        </div>

        <div class="task-section">
            <div class="task-badge">Task 5: Layout Challenge</div>
            <h3>The Structure</h3>
            <p class="note">
                <strong>Advanced:</strong> We want a standard website layout.
                This layout has long been called the <a
href="https://en.wikipedia.org/wiki/Holy_grail_(web_design)" target="_blank">'Holy Grail'</a>
                in web design because people have been searching for the best solution to achieve it for
a while before Flexbox and grid (there were only some clumsy and time-consuming workarounds earlier).
                The HTML is already tagged with classes (l-header, l-sidebar, etc.).
                Your job is to define the Grid on the parent container.
            </p>

            <div class="layout-challenge">
                <div class="l-header">Header</div>
                <div class="l-sidebar">Sidebar</div>
                <div class="l-main">Main Content Area</div>
                <div class="l-footer">Footer</div>
            </div>
        </div>

        <div class="task-section">
            <div class="task-badge" style="background: #dc3545;">Task 6: Debugging Zone</div>
            <h3>Fix the Broken Styles</h3>
```



```html
            <p class="note">These elements aren't behaving as expected. Find the bugs in the CSS!</p>

            <div style="margin-bottom: 20px;">
                <strong>6.1 The Stubborn Button:</strong><br><br>
                <span class="bug-btn">I want to be 200px wide!</span>
            </div>

            <hr>

            <div style="margin-bottom: 20px;">
                <strong>6.2 The Overflowing Image:</strong>
                <div class="bug-card">
                    <p>This card is only 300px wide, but the image is massive!</p>
                    <img src="https://placehold.co/600x150" alt="Too big" class="bug-img">
                </div>
            </div>

            <hr>

            <div style="margin-bottom: 20px;">
                <strong>6.3 The Hidden Message:</strong>
                <div class="bug-context">
                    <div class="bug-notification">
                        <strong>Important!</strong><br>
                        I should be on top!
                    </div>
                    <div class="bug-blocker">
                        I am just a grey block.
                    </div>
                </div>
            </div>
        </div>

    </div>

</body>
</html>
```

TASK 6: DEBUGGING ZONE

Fix the Broken Styles

These elements aren't behaving as expected. Find the bugs in the CSS!

6.1 The Stubborn Button:

I want to be 200px wide!

6.2 The Overflowing Image:

This card is only 300px wide, but the image is massive!

6.3 The Hidden Message:

Important!
I should be on top!