Building Layouts Before Flexbox

Learning Objectives

By the end of this lab, students will be able to:

- Create multi-column layouts using traditional CSS (floats and inline-block).
- Understand layout limitations before Flexbox.
- Compare fixed, inline-block, and float techniques.
- Recognize alignment and spacing challenges.

Lab Setup

Please edit and preview the following **starter HTML file**.
You should try to make sections behave like a simple web page: header, navigation, content area, sidebar, and footer — all without flex which we will discus in another session.

Your Task

1. **Task 1:** Use display: inline-block to align the navigation links horizontally.
2. **Task 2:** Float the main article and sidebar to create a two-column layout.
3. **Task 3:** Add a footer that stays below the floated elements.
4. **Task 4 (Challenge):** Add a media query to stack the article and sidebar on small screens.
5. **Task 5 (Reflection):**
    - What's hard about aligning and spacing these elements?
    - How does the layout behave when content size changes?
    - What happens if you remove overflow: auto from main?

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pre-Flexbox Layout Lab</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
        }

        /* 1️⃣ Header */
        header {
            background: #2c3e50;
            color: white;
            padding: 15px;
            text-align: center;
        }

        /* 2️⃣ Navigation using inline-block */
        nav {
            background: #34495e;
            padding: 10px;
            text-align: center;
        }

            nav a {
                display: inline-block;
                color: white;
                text-decoration: none;
                padding: 10px 20px;
                background: #3b5998;
                margin: 5px;
            }

                nav a:hover {
                    background: #2b4162;
                }

        /* 3️⃣ Main content area using floats */
        main {
            overflow: auto; /* clears floated elements */
            padding: 10px;
            background: #ecf0f1;
        }

        article {
            float: left;
            width: 70%;
            background: white;
            padding: 20px;
            box-sizing: border-box;
        }

        aside {
            float: right;
            width: 25%;
            background: #bdc3c7;
            padding: 20px;
            box-sizing: border-box;
        }

        /* 4️⃣ Footer using block layout */
        footer {
            clear: both;
            background: #2c3e50;
            color: white;
            text-align: center;
            padding: 15px;
            margin-top: 10px;
        }

        /* 5️⃣ Challenge: Responsive behavior (optional) */
        @media (max-width: 700px) {
            article, aside {
                float: none;
                width: 100%;
            }
        }
    </style>
</head>
<body>

    <header>
        <h1>Pre-Flexbox Layout</h1>
    </header>

    <nav>
        <a href="#">Home</a>
        <a href="#">About</a>
        <a href="#">Services</a>
        <a href="#">Contact</a>
    </nav>

    <main>
        <article>
            <h2>Main Content</h2>
            <p>This is the main content area. Notice that we use <strong>float:
left</strong> to position it next to the sidebar. Without Flexbox, alignment and equal
height are difficult to achieve.</p>
        </article>

        <aside>
            <h3>Sidebar</h3>
            <p>This sidebar uses <strong>float: right</strong>. Try resizing the
window — notice how it behaves before the media query kicks in.</p>
        </aside>
    </main>

    <footer>
        <p>Footer Area — aligned using traditional block layout.</p>
    </footer>
```

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