CS 110

Lab 5: CSS Grid Layout

Overview

In this lab, you will create a responsive web page layout using CSS Grid. You'll learn about grid templates, grid areas, and how to create sticky elements. The layout will feature a navbar, sidebar, main content area, and footer with proper spacing and positioning.

Getting Started

Download these starter files from Canvas and save them to your lab5 folder:

- `grid_layout.html`: Basic HTML structure
- `grid_layout.css`: CSS file for styling
- `article_content.txt`: Content for the main section

Instructions

Step 1: Basic Setup

Add the universal selector to visualize the grid structure:

```
* {
    border: 1px solid rgb(22, 21, 21);
    background: rgba(137, 135, 135, 0.1);
}
```

Preview using Live Server to see each area clearly defined.

Step 2: Initial Grid Layout

First, create a basic two-column layout:

```
body {
    display: grid;
    grid-template-columns: 300px 300px;
}
```

Preview using Live Server to see fixed-width columns.

Next, experiment with fraction units (fr):

```
/* Equal columns */
body {
```

```
    display: grid;
    grid-template-columns: 1fr 1fr;    /* Two equal columns, each taking 50%
*/
}
```

Preview using Live Server.

Try different fraction distributions:

```
/* 1:2 ratio */
grid-template-columns: 1fr 2fr;    /* First column 33%, second 67% */

/* 1:3 ratio */
grid-template-columns: 1fr 3fr;    /* First column 25%, second 75% */

/* Three columns */
grid-template-columns: 1fr 1fr 1fr;    /* Three equal columns */
```

Understanding fr units:

- `fr` = fraction of available space

- `1fr 1fr` = space divided equally (50% each)

- `1fr 2fr` = space divided proportionally (33% / 67%)

- Values scale dynamically with window size - More flexible than fixed pixels or percentages

Return to equal columns for next steps:

```
body {
    display: grid;
    grid-template-columns: 1fr 1fr;
}
```

Preview using Live Server and observe how columns grow and shrink proportionally.

Step 3: Grid Template Areas

Add grid areas to define the layout structure. The grid-template-areas property creates a visual layout where:

- Each set of quotes represents a row

- Area names define where elements will be placed

- Repeating names make elements span multiple cells

```
body {
    display: grid;
    grid-template-columns: 1fr 1fr;
```

```
    grid-template-areas:
        "navbar navbar"   /* Navbar spans full width */
        "sidebar main"    /* Sidebar left, main content right */
        "sidebar main"    /* Repeated for height */
        "footer footer"   /* Footer spans full width */
}
```

Then assign areas to elements:

```
.nav {
    grid-area: navbar;
}

.sidebar {
    grid-area: sidebar;
}

.main {
    grid-area: main;
}

.footer {
    grid-area: footer;
}
```

Preview using Live Server.

Step 4: Adjusting Column Widths

Change the grid template columns:

```
body {
    display: grid;
    grid-template-columns: 200px 1fr;
    grid-template-areas:
        "navbar navbar"
        "sidebar main"
        "sidebar main"
        "footer footer"
}
```

Preview using Live Server to see the fixed sidebar width.

Step 5: Full Height Layout

Add viewport height to ensure full-page coverage:

```
body {
    min-height: 100vh;  /* 100vh = 100% of viewport height */
    display: grid;
    grid-template-columns: 200px 1fr;
```

```
    grid-template-areas:
        "navbar navbar"
        "sidebar main"
        "sidebar main"
        "footer footer"
}
```

Preview using Live Server.

Step 6: Row Heights

Add row template definitions:

```
grid-template-rows: auto 1fr auto;
```

This creates:

- First row (auto): Sizes to navbar content
- Middle row (1fr): Takes remaining space
- Last row (auto): Sizes to footer content

Preview using Live Server.

Step 7: Navigation and Footer Styling

Style the navbar and footer:

```
.nav {
    grid-area: navbar;
    background-color: darkgray;
    color: white;
    padding: 2em;    /* 2em = 2x font size for more spacious header */
}

.footer {
    grid-area: footer;
    background-color: darkgray;
    color: white;
    padding: 1em;    /* 1em = 1x font size for standard footer spacing */
}
```

Preview using Live Server.

Step 8: Adding Content

Replace the “Hello World” heading in the main section with the copyright article about DMCA.

The article content can be found in the `article_content.txt` starter file. Copy the entire content and paste it within the main div, replacing:

```
<h1>Hello World!</h1>
```

With the article content that begins with:

```
<article>
    <h1>9.6 Copyright on the Internet</h1>
    ...
</article>
```

Preview with Live Server to see how the content flows in the grid layout.

Step 9: Sticky Navigation

Currently, when you scroll down, the navigation bar disappears from view. This creates a poor user experience since the navigation should remain accessible at all times. Let's fix this by making the navbar "sticky" so it stays visible while scrolling.

Add these properties to make the navbar stick to the top:

```
.nav {
    top: 0;
    position: sticky;
    grid-area: navbar;
    background-color: darkgray;
    color: white;
    padding: 2em;
}
```

Preview with Live Server and test scrolling behavior. Notice how the navbar now remains visible at the top of the viewport as you scroll through the content.

Step 10: Sticky Sidebar

Similar to the navbar issue, the sidebar currently disappears when scrolling. For better navigation and user experience, we want both the navbar and sidebar to remain visible at all times when scrolling through the main content.

First, try making the sidebar sticky:

```
.sidebar {
    top: 0;
    position: sticky;
    grid-area: sidebar;
}
```

Preview the changes but notice that the sidebar still isn't working correctly when scrolling.

We need to add `align-self: start` to fix the alignment:

```css
.sidebar {
    top: 0;
    position: sticky;
    align-self: start;
    grid-area: sidebar;
}
```

However, now the sidebar overlaps with the navbar. Let's fix this by adjusting the top position based on the navbar's height. Using Chrome Dev Tools, we can see the navbar height is 84 pixels:

```css
.sidebar {
    top: 84px;          /* Navbar height from Chrome Dev Tools */
    position: sticky;
    align-self: start;
    grid-area: sidebar;
}
```

Preview with Live Server and scroll to verify both navbar and sidebar stay in their correct positions.

Step 11: Sidebar Height

Add correct height calculation:

```css
.sidebar {
    height: calc(100vh - 136px);  /* Subtract navbar (84px) and footer (52px)
*/
    top: 84px;
    position: sticky;
    align-self: start;
    grid-area: sidebar;
}
```

Preview with Live Server.

Step 12: Responsive Design

Add media queries for small screens:

```css
@media (max-width: 768px) {
    /* Change from two columns (200px 1fr) to single column */
    body {
        grid-template-columns: 1fr;    /* Single column taking full width */
    }
    .nav {
        display: none;
    }
    .sidebar {
        display: none;
    }
```

```
    .footer {
        display: none;
    }
}
```

Understanding the grid-template-columns change:

- Original layout: `200px 1fr` (fixed sidebar + flexible main content)
- Mobile layout: `1fr` (single column using full width)
- `1fr` means one fraction unit of available space (100% width)
- This creates a mobile-friendly layout where content flows vertically
- Works with `display: none` on nav/sidebar to focus on main content

Preview with Live Server and test these responsive changes by:

1. Opening Chrome Dev Tools (F12)
2. Clicking the device toggle toolbar button
3. Selecting different device sizes
4. Observing how layout changes at 768px breakpoint

Practice Exercise

Modify the CSS to make the footer stay fixed at the bottom of the viewport.

Testing and Validation

1. Test all layout components:
    - Scroll behavior of sticky elements
    - Content flow and spacing
    - Responsive design with different screen sizes
2. Validate code:
    - Use W3C HTML Validator
    - Use W3C CSS Validator
    - Fix any validation errors

Submission

1. Verify all layout elements work correctly
2. Test sticky positioning while scrolling
3. Validate both HTML and CSS files
4. Submit:
    - grid_layout.html and grid_layout.css