**Week 4: CSS Layouts**

**Objective:**

By the end of this class, students should understand how to use **Flexbox** and **CSS Grid** to create responsive layouts. They will also build a **responsive card layout** as a hands-on project.

**1. Introduction to CSS Layouts (10 mins)**

Start by explaining:

* Why CSS layouts are important.
* The difference between **Flexbox** and **CSS Grid**.
* When to use **Flexbox** vs **CSS Grid**.

**Key Differences:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Flexbox** | **Grid** |
| **Use Case** | One-dimensional layout (row or column) | Two-dimensional layout (rows + columns) |
| **Main Property** | display: flex; | display: grid; |
| **Best For** | Aligning items in a row or column | Structuring entire page layouts |

**2. Flexbox: Basics and Use Cases (30 mins)**

**Introduction to Flexbox**

* Flexbox is a layout system that helps position and align items easily.
* It works on a **parent-child relationship**, where the parent (display: flex;) controls its children.

**Core Flexbox Properties**

1. **Parent (Container) Properties**
   * display: flex; → Enables Flexbox.
   * flex-direction: row | column; → Controls item direction.
   * justify-content: flex-start | center | space-between; → Aligns items horizontally.
   * align-items: flex-start | center | stretch; → Aligns items vertically.
   * gap: 10px; → Adds space between flex items.
2. **Child (Item) Properties**
   * flex-grow: 1; → Allows items to grow and take up space.
   * flex-shrink: 0; → Prevents items from shrinking.
   * align-self: flex-start | center | stretch; → Controls individual item alignment.

**Hands-on: Creating a Flexbox Navigation Bar**  
Guide students through building a simple **nav bar** using Flexbox.

<header>

<nav class="navbar">

<ul>

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

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

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

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

</ul>

</nav>

</header>

.navbar {

display: flex;

justify-content: space-between;

background: #333;

padding: 10px;

}

.navbar ul {

display: flex;

list-style: none;

gap: 15px;

}

.navbar a {

color: white;

text-decoration: none;

}

**3. CSS Grid: Basics and Creating Simple Grids (30 mins)**

**What is CSS Grid?**

* CSS Grid is a two-dimensional layout system.
* It allows control over both **rows** and **columns**.

**Key CSS Grid Properties**

1. **Parent (Grid Container) Properties**
   * display: grid; → Enables Grid.
   * grid-template-columns: repeat(3, 1fr); → Creates 3 equal columns.
   * grid-template-rows: auto; → Defines row size.
   * gap: 10px; → Adds spacing between grid items.
2. **Child (Grid Item) Properties**
   * grid-column: span 2; → Makes an item take up 2 columns.
   * grid-row: span 2; → Makes an item take up 2 rows.

**Hands-on: Creating a Simple Grid Layout**  
Guide students in building a **basic grid layout**.

<section class="grid-container">

<div class="box">1</div>

<div class="box">2</div>

<div class="box">3</div>

<div class="box">4</div>

</section>

.grid-container {

display: grid;

grid-template-columns: repeat(2, 1fr);

grid-template-rows: repeat(2, 100px);

gap: 10px;

}

.box {

background: lightblue;

display: flex;

align-items: center;

justify-content: center;

font-size: 20px;

font-weight: bold;

}

**4. Hands-on: Building a Responsive Card Layout (30 mins)**

Now, combine **Flexbox and Grid** to build a **responsive card layout**.

**Final Project Structure**

<section class="card-container">

<div class="card">

<img src="image1.jpg" alt="Card Image">

<h3>Card Title</h3>

<p>Card description goes here.</p>

</div>

<div class="card">

<img src="image2.jpg" alt="Card Image">

<h3>Card Title</h3>

<p>Card description goes here.</p>

</div>

<div class="card">

<img src="image3.jpg" alt="Card Image">

<h3>Card Title</h3>

<p>Card description goes here.</p>

</div>

</section>

.card-container {

display: grid;

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

gap: 20px;

}

.card {

background: white;

padding: 15px;

border-radius: 5px;

box-shadow: 2px 2px 10px rgba(0, 0, 0, 0.1);

text-align: center;

}

.card img {

width: 100%;

height: auto;

border-radius: 5px;

}

**5. Q&A and Recap (10 mins)**

**Ask students:**

* When should you use **Flexbox** instead of **Grid**?
* What’s the difference between **justify-content** and **align-items**?
* How does **grid-template-columns: repeat(3, 1fr);** work?

**Final Recap:**

✅ **Flexbox** = One-directional layout, great for navigation bars.  
✅ **Grid** = Two-dimensional layout, great for page structures.  
✅ **Use both** to create modern, responsive web designs!

**Homework Assignment**

💡 **Create a Responsive Portfolio Layout**  
Ask students to build a **portfolio webpage** using **Flexbox and Grid**.

* **Header with a nav bar** (Flexbox)
* **Grid-based layout** for projects
* **Footer** with contact links

Encourage them to submit their work before the next class! 🚀