Sesión 5: Diseño Avanzado con CSS: Flexbox y Grid Layout

Duración: 3 horas

Objetivo de la clase:

Al finalizar esta sesión, los estudiantes serán capaces de utilizar **Flexbox** y **Grid Layout** para crear diseños avanzados y responsivos en CSS, organizando de manera flexible el contenido de sus páginas web.

---

1. Revisión de la Sesión Anterior (15 minutos)

- Repaso rápido de los conceptos de **CSS**, **selectores de clase e ID**, y la integración de **Bootstrap**.
- Verificación de la tarea: revisión de las páginas creadas con Bootstrap y el uso de clases e IDs personalizados.

¿Alguien tiene alguna duda antes de comenzar con los nuevos temas?

---

2. Introducción a Flexbox (45 minutos)

Flexbox es un modelo de diseño unidimensional que facilita la alineación, distribución y ordenamiento de elementos dentro de un contenedor. Es ideal para crear layouts flexibles y responsivos.

2.1. Conceptos Básicos de Flexbox

- **Contenedor Flex (display: flex)**: Definir un contenedor como flex permite organizar a sus elementos hijos de manera flexible.

```
.contenedor {
  display: flex;
}
```

- **Ejes de Flexbox**:
    - **Eje principal (main axis)**: Es la dirección principal en la que los elementos se organizan (por defecto, horizontal).
    - **Eje transversal (cross axis)**: Es perpendicular al eje principal (por defecto, vertical).

2.2. Propiedades del Contenedor Flex

- **flex-direction**: Define la dirección en la que se organizan los elementos (row, row-reverse, column, column-reverse).

```
.contenedor {
  flex-direction: row; /* Los elementos se colocan en fila */
}
```

- **justify-content**: Alinea los elementos a lo largo del eje principal (flex-start, flex-end, center, space-between, space-around).

```
.contenedor {
  justify-content: center; /* Centra los elementos en el eje principal */
}
```

- **align-items**: Alinea los elementos a lo largo del eje transversal (flex-start, flex-end, center, stretch).

```
.contenedor {
  align-items: center; /* Centra los elementos en el eje transversal */
}
```

2.3. Propiedades de los Elementos Flexibles

- **flex-grow**: Define cómo un elemento puede crecer para llenar el espacio disponible.

```
.elemento {
  flex-grow: 1; /* El elemento crecerá para ocupar el espacio disponible */
}
```

- **flex-shrink**: Define cómo un elemento puede reducir su tamaño cuando no hay suficiente espacio.
- **flex-basis**: Define el tamaño inicial de un elemento antes de distribuir el espacio.

```
.elemento {
  flex-basis: 200px; /* El tamaño inicial del elemento será de 200px */
}
```

3. Introducción a CSS Grid Layout (45 minutos)

Grid Layout es un modelo de diseño bidimensional que permite organizar elementos en filas y columnas, facilitando la creación de layouts complejos y adaptables.

3.1. Conceptos Básicos de CSS Grid

- **Contenedor Grid (display: grid)**: Define un contenedor como una cuadrícula.

```
.grid-contenedor {
  display: grid;
}
```

Definir Filas y Columnas:

- **grid-template-columns**: Especifica el número y tamaño de las columnas.
- **grid-template-rows**: Especifica el número y tamaño de las filas.

```
.grid-contenedor {
  grid-template-columns: 1fr 2fr; /* Primera columna ocupa 1 parte, segunda 2 partes */
  grid-template-rows: 100px 200px; /* Dos filas de 100px y 200px */
}
```

3.2. Posicionamiento de Elementos en CSS Grid

- **grid-column** y **grid-row**: Permiten definir en qué columna o fila se posicionará un elemento.

```
.elemento {
  grid-column: 1 / 3; /* El elemento ocupa desde la columna 1 hasta la 3 */
  grid-row: 2 / 3; /* El elemento ocupa desde la fila 2 hasta la 3 */
}
```

gap: Define el espacio entre filas y columnas de la cuadrícula.

```
.grid-contenedor {
  gap: 10px; /* Espacio de 10px entre los elementos */
}
```

3.3. Propiedades Avanzadas de CSS Grid

- **repeat()**: Facilita la creación de un número de columnas o filas de tamaño repetido.

```
.grid-contenedor {
  grid-template-columns: repeat(3, 1fr); /* Tres columnas de igual tamaño */
}
```

auto-fit y auto-fill: Permiten crear cuadrículas responsivas que se adaptan al tamaño del contenedor.

4. Ejercicio Práctico: Creación de Layouts con Flexbox y Grid (45 minutos)

Ahora vamos a crear un pequeño proyecto donde utilizaremos **Flexbox** y **Grid Layout** para diseñar un sitio web responsivo.

Instrucciones:

1. Crea un archivo HTML llamado layout.html.
2. Aplica **Flexbox** para organizar una barra de navegación.
3. Utiliza **CSS Grid** para crear una sección de contenido principal con una galería de imágenes.

Ejemplo:

Archivo layout.html:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <title>Layout con Flexbox y Grid</title>
    <style>
      .navbar {
        display: flex;
        justify-content: space-between;
        background-color: #333;
        padding: 1rem;
      }
      .navbar a {
        color: white;
        text-decoration: none;
        margin: 0 15px;
      }
```

```
    .grid-contenedor {
      display: grid;
      grid-template-columns: repeat(3, 1fr);
      gap: 10px;
      padding: 20px;
    }
    .grid-contenedor img {
      width: 100%;
      height: auto;
      border-radius: 5px;
    }
  </style>
</head>
<body>
  <div class="navbar">
    <a href="#">Inicio</a>
    <a href="#">Acerca</a>
    <a href="#">Contacto</a>
  </div>

  <div class="grid-contenedor">
    <img src="img1.jpg" alt="Imagen 1">
    <img src="img2.jpg" alt="Imagen 2">
    <img src="img3.jpg" alt="Imagen 3">
    <img src="img4.jpg" alt="Imagen 4">
    <img src="img5.jpg" alt="Imagen 5">
    <img src="img6.jpg" alt="Imagen 6">
```

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

5. Introducción a Flexbox con Bootstrap (45 minutos)

Flexbox es un modelo de diseño unidimensional que organiza elementos en filas o columnas, facilitando la creación de layouts dinámicos y adaptables.

5.1. Flexbox con Bootstrap

Bootstrap incluye utilidades de Flexbox que permiten alinear y distribuir elementos de manera sencilla:

- **d-flex**: Convierte un elemento en un contenedor flex.
- **justify-content-***: Alinea los elementos a lo largo del eje principal (start, end, center, between, around).
- **align-items-***: Alinea los elementos a lo largo del eje transversal (start, center, end, stretch).

Ejemplo de Flexbox con Bootstrap:

```
<div class="d-flex justify-content-center align-items-center"
style="height: 200px;">
  <div class="p-2 bg-primary text-white">Elemento 1</div>
  <div class="p-2 bg-secondary text-white">Elemento 2</div>
  <div class="p-2 bg-success text-white">Elemento 3</div>
</div>
```

En este ejemplo:

- **d-flex** convierte el contenedor en un Flexbox.
- **justify-content-center** centra los elementos horizontalmente.
- **align-items-center** los centra verticalmente.

5.2. Propiedades adicionales de Flexbox

Además de las utilidades de Bootstrap, puedes personalizar la alineación utilizando CSS directo para detalles específicos.

---

6. Creación de Grillas con el Sistema de Grid de Bootstrap (45 minutos)

Bootstrap incluye un **sistema de grillas** que facilita la organización del contenido en filas y columnas, adaptándose automáticamente a diferentes tamaños de pantalla.

6.1. Estructura Básica del Sistema de Grid

- **container**: Define un contenedor central con márgenes automáticos.
- **row**: Define una fila dentro del contenedor.
- **col-***: Define columnas dentro de la fila, utilizando un sistema de 12 columnas.

Ejemplo básico de Grid con Bootstrap:

```
<div class="container">
  <div class="row">
    <div class="col-md-4">Columna 1</div>
    <div class="col-md-4">Columna 2</div>
    <div class="col-md-4">Columna 3</div>
  </div>
</div>
```

En este ejemplo:

- **col-md-4** define tres columnas que ocuparán 4 de las 12 partes disponibles en pantallas medianas.

6.2. Utilidades Responsivas

- **col-**, **col-sm-**, **col-md-**, **col-lg-**, **col-xl-**: Adaptan el número de columnas según el tamaño de la pantalla.
- **g-**: Controla el espaciado entre columnas.

Ejemplo con Espaciado:

```
<div class="container">
  <div class="row g-4">
    <div class="col-6 col-md-3">Columna A</div>
    <div class="col-6 col-md-3">Columna B</div>
    <div class="col-6 col-md-3">Columna C</div>
    <div class="col-6 col-md-3">Columna D</div>
  </div>
</div>
```

En este ejemplo, el espaciado entre columnas se define con **g-4**, que agrega un espacio de 1.5rem entre columnas.

---

7. Ejercicio Práctico: Creación de un Layout con Bootstrap (45 minutos)

Ahora, crearemos una página web simple que combine **Flexbox** y **Grid Layout** utilizando las clases predefinidas de Bootstrap para crear una barra de navegación y una galería de imágenes.

Instrucciones:

1. Crea un archivo HTML llamado layout-bootstrap.html.
2. Usa **Flexbox** para diseñar una barra de navegación.
3. Utiliza el sistema de **Grid** de Bootstrap para organizar una galería de imágenes.

Ejemplo de layout-bootstrap.html:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Layout con Bootstrap</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
  </head>
  <body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
      <div class="container-fluid">
        <a class="navbar-brand" href="#">Mi Sitio</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav">
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
          <ul class="navbar-nav ms-auto">
            <li class="nav-item">
              <a class="nav-link" href="#">Inicio</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#">Acerca</a>
            </li>
            <li class="nav-item">
```

```
              <a class="nav-link" href="#">Contacto</a>
            </li>
          </ul>
        </div>
      </div>
    </nav>

    <div class="container my-4">
      <div class="row g-3">
        <div class="col-md-4">
          <img src="img1.jpg" class="img-fluid rounded"
alt="Imagen 1">
        </div>
        <div class="col-md-4">
          <img src="img2.jpg" class="img-fluid rounded"
alt="Imagen 2">
        </div>
        <div class="col-md-4">
          <img src="img3.jpg" class="img-fluid rounded"
alt="Imagen 3">
        </div>
        <div class="col-md-4">
          <img src="img4.jpg" class="img-fluid rounded"
alt="Imagen 4">
        </div>
        <div class="col-md-4">
          <img src="img5.jpg" class="img-fluid rounded"
alt="Imagen 5">
```

```
        </div>
        <div class="col-md-4">
          <img src="img6.jpg" class="img-fluid rounded"
alt="Imagen 6">
        </div>
      </div>
    </div>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstra
p.bundle.min.js"></script>
  </body>
</html>
```

En este ejemplo:

- **Flexbox** se utiliza en la barra de navegación para organizar enlaces y el menú.
- **Grid** organiza las imágenes de la galería en tres columnas.
- **img-fluid** hace que las imágenes sean responsivas.

---

8. Cierre y Preguntas (15 minutos)

Hoy hemos explorado cómo usar **Flexbox** y **Grid Layout** tanto con CSS puro como con **Bootstrap**. Esto nos permite crear layouts flexibles y adaptables de manera eficiente.