

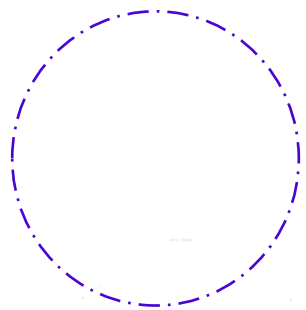


dev Senior JavaScript

MEAN Mastery de cero a líder Full Stack

www.devseniorcode.com






dev Senior

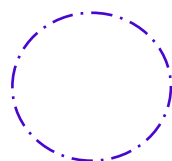


Módulo 1: Fundamentos del Desarrollo Web y JavaScript (Meses 1 y 2)

Objetivo: Dominar los fundamentos de JavaScript, HTML y CSS, y sentar las bases del desarrollo web.

Clase 6: CSS avanzado: Flexbox y Grid

 **Objetivo de la sesión:** Dominar el uso de Flexbox y Grid en CSS para construir interfaces web modernas y adaptables, integrando interacciones básicas con JavaScript.



www.devseniorcode.com

🧠 1. Introducción

- Breve repaso a layout tradicional con display: block, inline, float.
- Problemas que resuelven Flexbox y Grid.
- Casos de uso de cada uno.





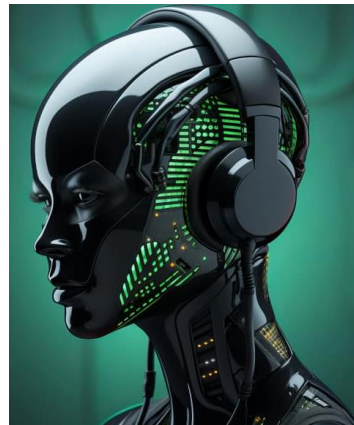
2. Flexbox

Contenedor: display: flex

Ejes principal y cruzado

Propiedades clave:

- Contenedor: justify-content, align-items, flex-direction, flex-wrap, gap
- Ítems: flex-grow, flex-shrink, flex-basis, order, align-self





Ejercicio 1: Galería con Flexbox

Objetivo: Crear una galería de imágenes con disposición flexible.

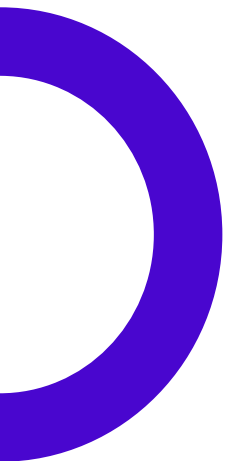
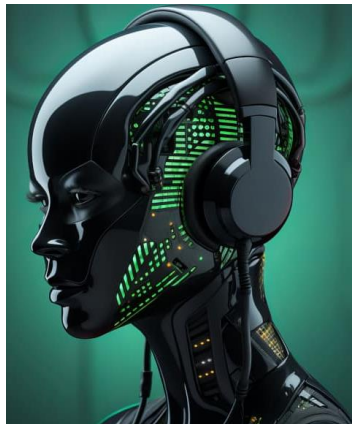
```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="style.css">
  <title>Galeria Flexbox</title>
</head>

<body>
  <div class="gallery">
    
    
    
    
    
    
    
    
  </div>
</body>

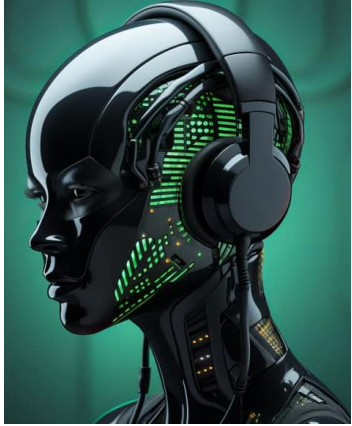
</html>
```

```
.gallery {
  display: flex;
  flex-wrap: wrap;
  gap: 10px;
}
.gallery img {
  width: 30%;
}
```





3. Grid Layout



- Contenedor: `display: grid`
- Definir columnas y filas: `grid-template-columns`, `grid-template-rows`
- Posicionamiento: `grid-area`, `grid-column`, `grid-row`
Áreas
- nombradas





Ejercicio 2: Tarjetas de productos en rejilla

Objetivo: Diseñar un catálogo de productos con varias tarjetas acomodadas automáticamente en filas y columnas, usando display: grid.

```
<!DOCTYPE html>
<html lang="en">

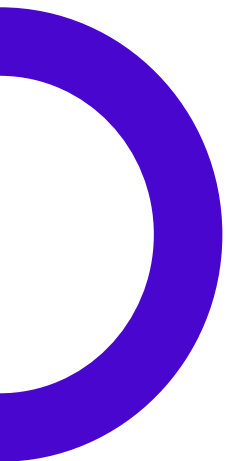
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Grid</title>
  <link rel="stylesheet" href="style2.css">
</head>

<body>
  <div class="product-grid">
    <div class="card">Producto 1</div>
    <div class="card">Producto 2</div>
    <div class="card">Producto 3</div>
    <div class="card">Producto 4</div>
    <div class="card">Producto 5</div>
    <div class="card">Producto 6</div>
  </div>
</body>

</html>
```

```
.product-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
  gap: 20px;
  padding: 20px;
}

.card {
  background-color: #f0f0f0;
  padding: 30px;
  border-radius: 10px;
  text-align: center;
  font-weight: bold;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}
```





Ejercicio 3: Cambiar diseño dinámicamente

Caso: Alternar entre un layout con Flexbox y otro con Grid.

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <link rel="stylesheet" href="style3.css">
</head>

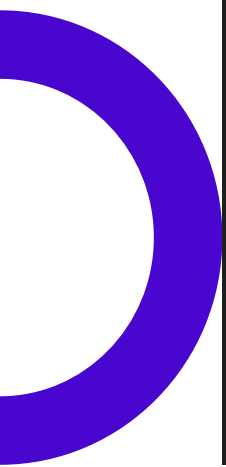
<body>
  <button onclick="toggleLayout()">Cambiar diseño</button>
  <div id="container" class="flex-layout">
    <div>Item 1</div>
    <div>Item 2</div>
    <div>Item 3</div>
  </div>
</body>
<script src="diseño.js"> </script>

</html>
```

```
.flex-layout {
  display: flex;
  gap: 10px;
}

.grid-layout {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 10px;
}
```

```
function toggleLayout() {
  const container = document.getElementById('container');
  container.classList.toggle('flex-layout');
  container.classList.toggle('grid-layout');
}
```



6. Conclusiones



- Flexbox y Grid son herramientas fundamentales para construir interfaces modernas: Permiten diseñar layouts responsivos, adaptables y organizados sin depender de técnicas obsoletas como float o position: absolute.
- Flexbox se adapta mejor a diseños lineales y componentes pequeños, mientras que Grid es ideal para estructuras de página completas: Conocer cuándo usar cada uno permite mayor eficiencia y claridad en el desarrollo.
- La integración de CSS avanzado con JavaScript potencia la interactividad del sitio web: Usar JavaScript para alternar diseños o responder a eventos del usuario abre la puerta a experiencias dinámicas y personalizables.





¿ Dudas e Inquietudes ?



Gracias