¿Qué es CSS Grid?

CSS Grid Layout (o simplemente **Grid**) es un sistema de diseño bidimensional que permite crear **estructuras de página complejas y alineadas** usando filas y columnas.

A diferencia de **Flexbox**, que se enfoca en una sola dirección (fila o columna), **Grid permite trabajar ambas al mismo tiempo**, facilitando diseños más completos y organizados.

Ventajas de CSS Grid

1. **Diseño bidimensional**: puede controlar filas y columnas simultáneamente.
2. **Facilita la alineación**: ubica elementos en el lugar exacto dentro de un contenedor es muy sencillo.
3. **Menos código CSS**: evita usar floats, positions o hacks para alinear elementos.
4. **Responsive design más fácil**: se adapta fácilmente a diferentes tamaños de pantalla.
5. **Control total del espacio**: puede definir tamaños fijos, automáticos o proporcionales con fr (fracción del espacio disponible).

Desventajas de CSS Grid

1. **No todos los navegadores antiguos lo soportan** (aunque hoy en día la mayoría sí).
2. **Curva de aprendizaje**: al inicio puede ser confuso por la cantidad de propiedades y opciones.
3. **No reemplaza totalmente a Flexbox**: Flex sigue siendo mejor para distribuciones lineales o alineaciones dentro de una fila/columna.

Casos de uso comunes

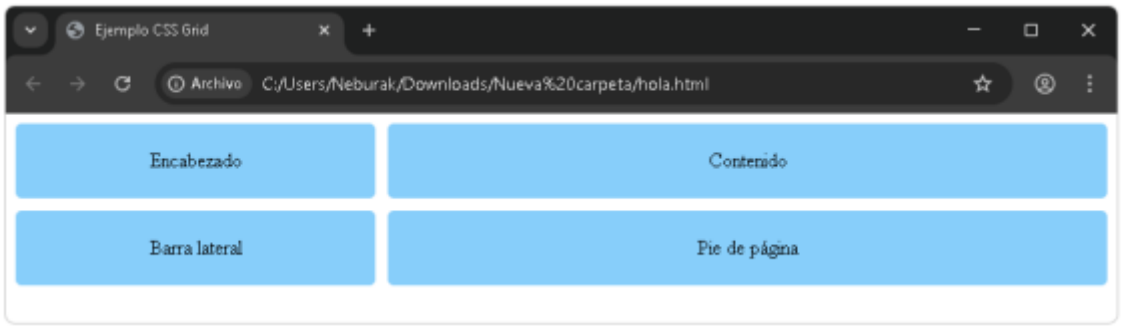
- Crear **layouts de página** completos (cabecera, contenido, barra lateral, pie de página).
- Diseñar **galerías de imágenes** ajustables.
- Organizar **dashboards o paneles** con secciones que cambian de tamaño.

- Distribuir **cartas o productos** de forma ordenada y flexible.

Ejemplo 1: Layout básico con Grid

```
<!DOCTYPE html>
<html lang="es">
<head>
 <meta charset="UTF-8">
 <title>Ejemplo CSS Grid</title>
 <style>
  .contenedor {
   display: grid;
   grid-template-columns: 1fr 2fr; /* Dos columnas: una más ancha */
   grid-template-rows: auto auto;
   gap: 10px; /* Espacio entre elementos */
  }

  .item {
   background-color: #87cefa;
   padding: 20px;
   text-align: center;
   border-radius: 5px;
  }
 </style>
</head>
<body>
```

```
 <div class="contenedor">
   <div class="item">Encabezado</div>
   <div class="item">Contenido</div>
   <div class="item">Barra lateral</div>
   <div class="item">Pie de página</div>
 </div>


</body>
</html>
```



📄 **Explicación:**
Se crea una cuadrícula de **2 columnas** con un espacio (gap) entre los elementos. Las cajas se colocan automáticamente en las filas según el flujo del HTML.

Ejemplo 2: Galería de imágenes responsive

```
<style>
 .galeria {
   display: grid;
   grid-template-columns: repeat(auto-fill, minmax(150px, 1fr));
   gap: 15px;
 }
```

```
  .galeria img {
    width: 100%;
    border-radius: 8px;
  }
</style>

<div class="galeria">
  <img src="https://via.placeholder.com/150" alt="foto 1">
  <img src="https://via.placeholder.com/150" alt="foto 2">
  <img src="https://via.placeholder.com/150" alt="foto 3">
  <img src="https://via.placeholder.com/150" alt="foto 4">
</div>
```

🗒️ **Explicación:**
La galería usa auto-fill y minmax() para crear un diseño **automáticamente ajustable**.

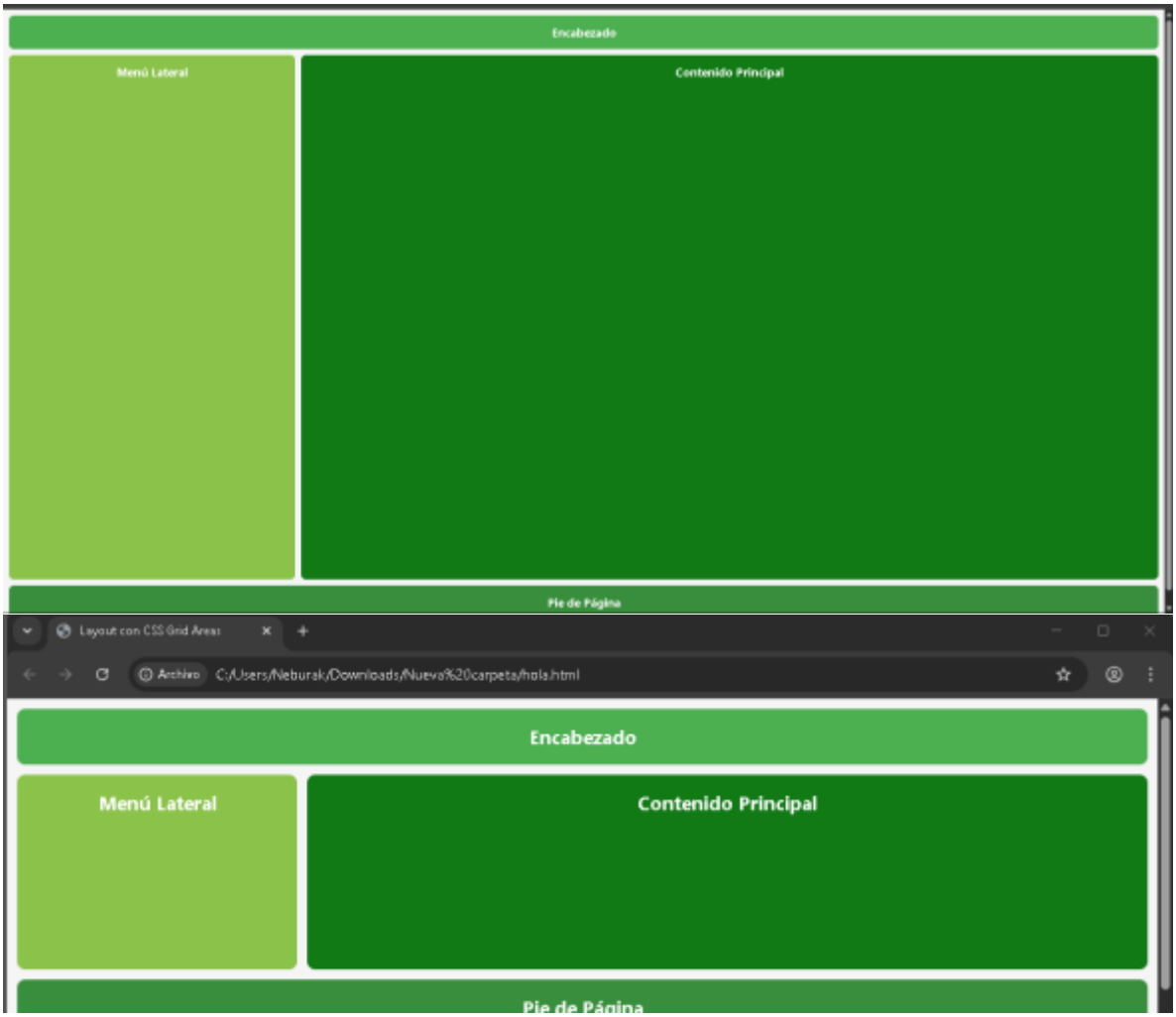
Las imágenes se reacomodan según el tamaño de pantalla, sin necesidad de media queries.(me dio pereza buscar imágenes, pero funciona 👍).

Ejemplo 3: Posicionamiento personalizado

```
<style>
```

```
  .layout {
    display: grid;
    grid-template-areas:
      "header header"
      "menu main"
      "footer footer";
    grid-template-columns: 1fr 3fr;
    gap: 10px;
  }

  header { grid-area: header; background: #4caf50; }
  nav    { grid-area: menu;   background: #8bc34a; }
  main   { grid-area: main;   background: #c8e6c9; }
  footer { grid-area: footer; background: #388e3c; }

  header, nav, main, footer {
    padding: 15px;
    color: white;
    text-align: center;
  }
</style>

<div class="layout">
  <header>Encabezado</header>
  <nav>Menú lateral</nav>
  <main>Contenido principal</main>
```

```
  <footer>Pie de página</footer>

</div>
```



📄 **Explicación:**
Con grid-template-areas, se puede **dibujar visualmente** el diseño y asignar cada elemento a una "zona" del grid, lo que facilita la lectura y mantenimiento del código.

Resumen

Concepto	**Descripción**
Nombre	CSS Grid Layout
Tipo	Sistema de diseño bidimensional (filas y columnas)

Ventajas	Control total, responsive, ordenado, menos código
Desventajas	Curva de aprendizaje, soporte limitado en navegadores antiguos
Casos comunes	Layouts, galerías, dashboards, estructuras web
Compatibilidad	Soportado por todos los navegadores modernos