«Talento Tech»

Front-End JS

Clase 07

Buenos Aires Ciudad

Clase N° 07: CSS 4 - Grid, Media Queries

Temario:

1. **Grid**
    - ¿Qué es Grid?
    - Implementación de Grid
    - Maquetado con Flex y Grid
    - Conceptos Básicos de Grid
    - Propiedades Importantes en Grid
2. **Media Queries**
    - ¿Qué son Media Queries?
    - Implementación de Media Queries
    - Breakpoints en Media Queries
    - Orientaciones

Objetivo de la clase:

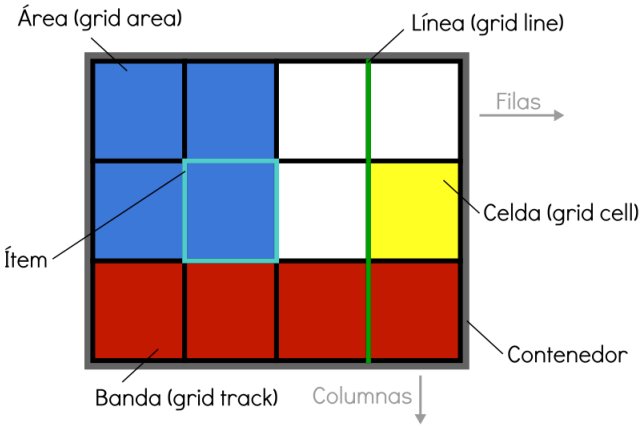
En esta clase vamos a profundizar en **CSS Grid**, aprendiendo qué es, cómo se implementa y cómo usarlo junto a **Flexbox** para maquetar sitios de manera eficiente. Vamos a repasar sus conceptos básicos y las propiedades más importantes, incluyendo `grid-area`, `gap` y la definición de filas y columnas.

También vamos a trabajar el diseño **responsive** con **Media Queries**, entendiendo qué son, cómo se usan y cuáles son los *breakpoints* más comunes, incluyendo la adaptación según la orientación de pantalla.

Con estos conceptos, vas a poder construir interfaces más adaptables, limpias y visualmente atractivas.

Grid

CSS Grid Layout es una técnica poderosa que te permite organizar el contenido de una página web en una cuadrícula (grid) de filas y columnas. A diferencia de Flexbox, que solo te permite trabajar con una sola dimensión (filas o columnas), Grid es bidimensional, lo que significa que podés organizar el contenido en ambas direcciones al mismo tiempo.



Ventajas de CSS Grid:

- **Control Completo:** te permite organizar y posicionar los elementos de forma precisa en la cuadrícula, tanto horizontal como verticalmente.
- **Diseños Complejos:** es perfecto para crear diseños de páginas complejas con múltiples secciones, como cabeceras, barras laterales, contenido principal y pies de página.
- **Responsividad:** Grid hace que sea fácil ajustar tu diseño para diferentes tamaños de pantalla, haciendo que tu sitio se vea bien en cualquier dispositivo.

Implementación de Grid

Para empezar a usar CSS Grid, todo lo que tenés que hacer es definir un contenedor como un grid utilizando `display: grid`

A continuación, definís cómo se van a distribuir las columnas y las filas dentro de este grid.

Ejemplo básico de Grid:

```
<div class="grid-container">
   <div class="header">Cabecera</div>
   <div class="menu">Menú</div>
   <div class="main">Contenido Principal</div>
   <div class="footer">Pie de página</div>
</div>
```

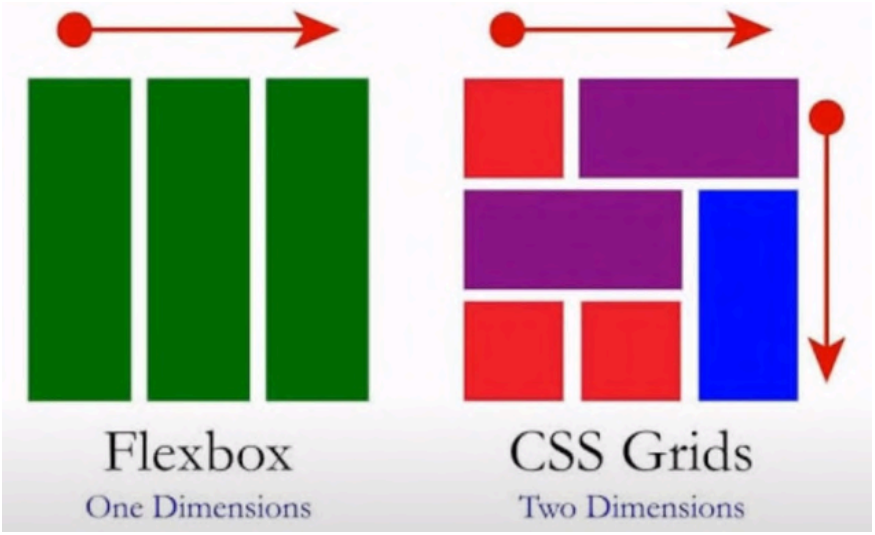
CSS básico para Grid:

```
.grid-container {
display: grid;
grid-template-columns: 200px 1fr; /* Una columna de 200px y otra que ocupa el espacio restante */
grid-template-rows: auto 1fr auto; /* Tres filas, la del medio es flexible */
gap: 10px; /* Espacio entre los elementos */
}

.header {
grid-column: 1 / 3; /* La cabecera ocupa las dos columnas */
}

.menu {
grid-column: 1 / 2; /* El menú ocupa la primera columna */
}

.main {
grid-column: 2 / 3; /* El contenido principal ocupa la segunda columna */ }

.footer {
grid-column: 1 / 3; /* El pie de página también ocupa las dos columnas */
}
```

Explicación:

- **grid-template-columns:** define cuántas columnas tiene el layout y el tamaño de cada una. En el ejemplo, la primera columna tiene un tamaño fijo de `200px`, y la segunda columna ocupa el espacio restante (`1fr`).
- **grid-template-rows:** define el número de filas y su tamaño; la primera y la última fila (`auto`) se ajustan automáticamente al contenido, mientras que la fila del medio (`1fr`) ocupa el espacio restante.
- **grid-column:** controla en qué columnas se sitúa un elemento; por ejemplo, la cabecera ocupa las dos columnas, mientras que el contenido principal ocupa solo una.

Maquetado con Flex y Grid



En proyectos de diseño web, a veces es útil combinar Grid y Flexbox, ya que cada uno tiene sus propias ventajas:

- **Grid:** Es ideal para la estructura principal de la página, donde necesitas definir áreas específicas para cada sección (cabecera, contenido, pie, etc.). Grid es perfecto cuando trabajás con una disposición de columnas y filas simultáneamente.
- **Flexbox:** Funciona muy bien dentro de esas áreas específicas, ya que te permite distribuir y alinear elementos a lo largo de una sola dimensión (filas o columnas).

Ejemplo de combinación:

```
<div class="grid-container">
     <header class="header">Cabecera</header>
     <nav class="menu">
       <ul class="flex-menu">
         <li><a href="#">Inicio</a></li>
         <li><a href="#">Servicios</a></li>
         <li><a href="#">Contacto</a></li>
       </ul>
     </nav>
     <section class="main">Contenido principal</section>
     <footer class="footer">Pie de página</footer>
   </div>
```

CSS:

```css
.grid-container {
   display: grid;
   grid-template-columns: 200px 1fr;
   grid-template-rows: auto 1fr auto;
   gap: 10px;
}

.flex-menu {
   display: flex;
   flex-direction: column;
   gap: 5px;}

.header, .footer {
   grid-column: 1 / 3;
}

.main {
   grid-column: 2 / 3;
}
```

Explicación:

- Usamos Grid para organizar el layout principal de la página (cabecera, menú, contenido, pie de página).
- Dentro del menú (`.flex-menu`), utilizamos Flexbox para distribuir los elementos de la lista en una columna, con un espacio (`gap`) entre cada uno de ellos.

Conceptos Básicos de Grid

- **Grid Container:** es el contenedor que define la cuadrícula, configurado con `display: grid`.
- **Grid Item:** son los elementos dentro del grid container; se alinean en filas y columnas automáticamente.
- **Grid Track:** el espacio entre dos líneas adyacentes, ya sea una fila o una columna.
- **Grid Cell:** la intersección entre una fila y una columna, donde se coloca un grid item.
- **Grid Area:** un área rectangular dentro de la cuadrícula que puede estar ocupada por uno o varios grid items.

Propiedades Importantes en Grid

grid-template-columns: define cuántas columnas tendrá tu grid y el tamaño de cada una.

```
.grid-container {
    grid-template-columns: 1fr 2fr; /* Primera columna más pequeña, segunda más grande */
}
```

grid-template-rows: define el número de filas y su tamaño.

```
.grid-container {
    grid-template-rows: auto 1fr; /* Primera fila automática, segunda ocupa el resto */
}
```

grid-column / grid-row: permite controlar cuántas filas o columnas ocupa un grid item.

```
.header {
   grid-column: 1 / 3; /* Ocupa desde la columna 1 hasta la 3 */
}
```

gap: establece el espacio entre las filas y columnas del grid.

```
.grid-container {
   gap: 10px; /* Espacio uniforme entre filas y columnas */
}
```

`grid-area:` Permite asignar un área específica del grid a un elemento. Es especialmente útil cuando usás **`grid-template-areas`** para nombrar las secciones del layout.

```
/* Definición del layout con nombres de áreas */
.grid-container {
   display: grid;
   grid-template-areas:
      "header header"
      "sidebar main";
}
/* Asignación de cada elemento a un área */
.header {
   grid-area: header;
}
.sidebar {
   grid-area: sidebar;
}
.main {
   grid-area: main;
}
```

Media Queries

Las **Media Queries** son una funcionalidad de CSS que permite aplicar estilos dependiendo de las características del dispositivo, como el ancho o la orientación de la pantalla. Son clave para implementar un diseño **responsivo**, asegurando que tu sitio web se vea y funcione bien en todos los dispositivos, desde celulares hasta computadoras de escritorio.

Implementación de Media Queries

Para implementar Media Queries, usamos la regla `@media` en CSS. Esta regla permite establecer condiciones bajo las cuales ciertos estilos se aplicarán solo si esas condiciones se cumplen.

Breakpoints en Media Queries

Un **breakpoint** es el punto en el que se aplican estilos diferentes según el **ancho de pantalla** del dispositivo.

Esto permite que el diseño sea **responsivo** y se adapte bien a celulares, tablets, laptops y monitores grandes.

Rango ancho pantalla	**Tipo de dispositivo aproximado**
Hasta 600px	Celulares pequeños
600px – 768px	Tablets en modo vertical (portrait)
768px – 992px	Tablets en modo horizontal y laptops pequeñas
992px – 1200px	Laptops y monitores estándar
Más de 1200px	Monitores grandes

Veamos un ejemplo Práctico:

HTML (ESTRUCTURA)

```
<section class="galeria">
  <div class="card">Card 1</div>
  <div class="card">Card 2</div>
  <div class="card">Card 3</div>
  <div class="card">Card 4</div>
</section>
```

CSS (ESTILOS DE LA GALERÍA Y LAS TARJETAS)

```
.galeria {
  display: grid;
  grid-template-columns: 1fr; /* 1 columna por defecto */
  gap: 20px;
  padding: 20px;
}
.card {
  background-color: #90caf9;
  padding: 30px;
  text-align: center;
  border-radius: 10px;
  font-weight: bold;
}
```

RESPONSIVE (@MEDIA)

```
/* Tablets*/
@media (min-width: 600px) {
  .galeria {
    grid-template-columns: 1fr 1fr;
  }
}
/* Laptops y monitores medianos */
@media (min-width: 992px) {
  .galeria {
    grid-template-columns: 1fr 1fr 1fr;
  }
}
/* Pantallas grandes */
@media (min-width: 1200px) {
  .galeria {
    grid-template-columns: repeat(4, 1fr);
  }
}
```

💭**Storytelling**

Experiencia del Usuario y Adaptabilidad

Tomás (Desarrollador Senior)

¡Buen trabajo hasta acá! Ya tenemos una buena base estructural y visual. Ahora es momento de pensar en el feedback y la experiencia real del usuario en distintos dispositivos.
Para eso, vamos a trabajar en una sección muy especial: **Reseñas**.

Ejercicio práctico #1:

Sección “Reseñas”

Lucía (Product Owner)

Para esta sección necesitamos mostrar los testimonios de usuarios que ya probaron nuestros servicios. Queremos que se vean bien ordenados, que transmitan confianza y que se adapten a diferentes pantallas.

1. **Crear la estructura HTML**

- Dentro de tu etiqueta `<main>`, agregá una nueva `<section>`.
- Esta `<section>` debe tener una clase llamada `reseñas`.
- Dentro de ella, incluí **tres bloques** para los testimonios. Cada uno debe ser un elemento `<section>` con la clase `card`.
- En el interior de cada card podés incluir un nombre, una opinión y quizás una imagen o ícono (opcional).

2. **Aplicar CSS Grid**

- Usá `display: grid` en la clase `.reseñas`.
- Establecé un número de columnas **responsivo**, es decir, que se adapte al tamaño de pantalla.
- Usá `gap` para separar las tarjetas entre sí.
- Sumá `padding` para que las tarjetas no queden pegadas a los bordes.

3. **Aplicar estilo a las tarjetas `.card`**

- Añadí un `padding` interno y un `border` suave.
- Agregá una sombra (`box-shadow`) para que se destaquen visualmente.
- Usá una **transición suave** para los cambios visuales.

4. **Aplicar interactividad con pseudo-clases (EXTRA)**

- Usá la pseudo-clase `:hover` para generar un **efecto al pasar el mouse** (por ejemplo, que la tarjeta suba levemente).
- Sumá una transición con la propiedad `transition` para que el movimiento no sea brusco.

🛑Este es un gran momento para practicar `:hover`, `transition`, e incluso explorar `:nth-child()` si querés aplicar estilos diferentes a alguna tarjeta.

Resultado esperado:
Tres reseñas visualmente atractivas, distribuidas en pantalla de forma flexible, con interactividad y buena presentación.

Ejercicio práctico #2: Media Queries

¡Buen trabajo con las reseñas! Ahora es momento de asegurarnos de que el diseño funcione bien en celulares, tablets y computadoras. Vamos a practicar el uso de **Media Queries**, una herramienta clave para lograr que tu web se vea bien en todos lados.

Lo que harás:

Crear una media query para pantallas menores a 768px

- Dentro de tu archivo CSS, agregá una regla `@media` que se aplique a `max-width: 768px`.

2. Adaptar la sección `.reseñas`

- Dentro de la media query, indicá que las tarjetas se dispongan en **una sola columna**.
- Esto se hace modificando la propiedad de `grid-template-columns`.

3. Adaptar el formulario de "Contacto"

- Revisá tu HTML y asegurate de tener una estructura semántica clara, por ejemplo:

```
<section id="contacto">
  <form>
    <input type="text" placeholder="Tu nombre">
    <input type="email" placeholder="Tu correo">
    <textarea placeholder="Tu mensaje"></textarea>
  </form>
</section>
```

- Dentro del formulario, agregá campos para:
  - Nombre (input type text)
  - Correo (input type email)
  - Mensaje (textarea)

4. Estilos dentro de la media query

- Asegurate de que los `input` y `textarea` ocupen el **100% del ancho disponible**.
- Sumá `margin-bottom` entre los campos para que no queden pegados.
- Ajustá el `padding` general del formulario para que se vea cómodo en un celular.

🛑Podés usar `:focus` en los campos del formulario para que cambien de color cuando el usuario esté escribiendo.
Aplicá `transition` en los `input` para que el borde o el color cambien suavemente cuando estén activos.

Lucía te recomienda:

> Usá las herramientas del navegador para simular distintos tamaños de pantalla.
> Es la mejor forma de verificar si tu diseño está respondiendo correctamente.

Buenos Aires aprende

Agencia de Habilidades para el Futuro

Buenos Aires Ciudad