«Talento Tech»

Front-End JS

Clase 07

Buenos Aires Ciudad

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

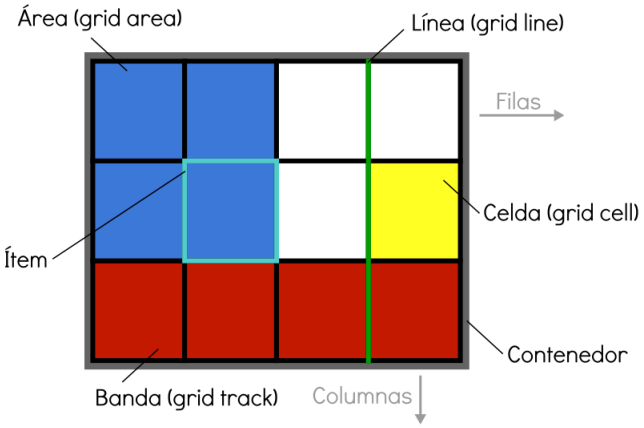
Temario:

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

1. Grid

¿Qué es 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:

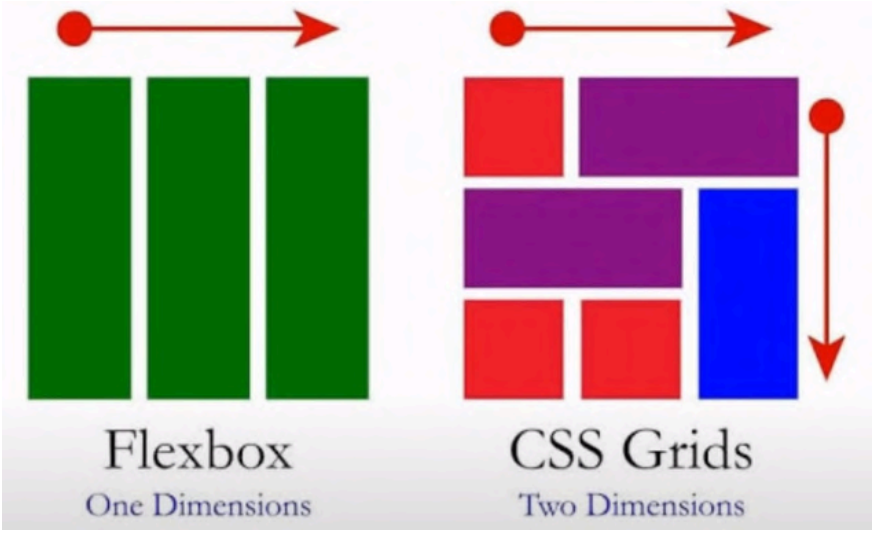
```
.grid-container {

   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:

```
.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 */
}
```

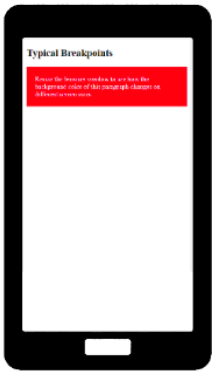
gap: establece el espacio entre las filas y columnas del grid.

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

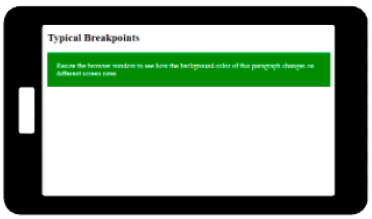
2. Media Queries

¿Qué son 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.

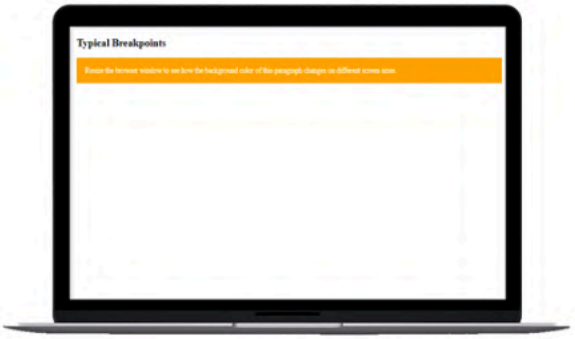


400px X 600px
Extra small devices (phones, 600px and down)



650px X 400px
Small devices (portrait tablets and large phones, 600px and up)

850px X 600px
Medium devices (landscape tablets, 768px and up)



1000px X 800px
Large devices (laptops/desktops, 992px and up)

1300px X 800px
Extra large devices (large laptops and desktops, 1200px and up)

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.

Ejemplo de Media Query básico:

```
/* Estilos generales */

body {

   font-size: 16px;

}

/* Estilos para pantallas pequeñas (menos de 600px) */

@media only screen and (max-width: 600px) {

   body {

        font-size: 14px;

   }

}

/* Estilos para pantallas medianas (más de 768px) */

@media only screen and (min-width: 768px) {

   body {

        font-size: 18px;

   }

}
```

Explicación:

- **@media only screen and (max-width: 600px):** aplica los estilos a pantallas con un ancho máximo de 600px (ideal para celulares).
- **@media only screen and (min-width: 768px):** aplica los estilos a pantallas con un ancho mínimo de 768px (usualmente tablets y computadoras más pequeñas).

Breakpoints en Media Queries

Un **breakpoint** es el punto en el que se aplican diferentes estilos dependiendo del ancho de la pantalla del dispositivo. Estos breakpoints son esenciales para asegurarse de que el diseño sea **responsivo** y adaptable.

Breakpoints comunes:

- **Hasta 600px:** dispositivos móviles pequeños (celulares).
- **600px - 768px:** tablets en modo vertical.
- **768px - 992px:** tablets en modo horizontal y laptops pequeñas.
- **992px - 1200px:** laptops y monitores de escritorio estándar.
- **Más de 1200px:** monitores grandes.

Ejemplo de breakpoints:

```
/* Estilos para pantallas pequeñas */

@media only screen and (max-width: 600px) {

    .container {

        grid-template-columns: 1fr; /* Una sola columna en pantallas
pequeñas */

    }

}

/* Estilos para pantallas medianas */

@media only screen and (min-width: 768px) {

    .container {

        grid-template-columns: 1fr 1fr; /* Dos columnas en pantallas
medianas */

    }

}
```

```
/* Estilos para pantallas grandes */

@media only screen and (min-width: 1200px) {

   .container {

       grid-template-columns: 1fr 1fr 1fr; /* Tres columnas en
pantallas grandes */

   }

}
```

Ejercicio práctico #1:

Secciòn “Reseñas”

Crear al menos 3 cards en la sección de "Reseñas", utilizando Grid, para organizar al menos tres cards de manera responsiva.

1. Estructura HTML: Asegurate de tener una estructura básica que incluya una sección para "Reseñas" dentro de <main>, y que cada reseña esté dentro de un con la clase card.
2. Aplicar CSS Grid: En el archivo styles.css, aplicá display: grid y grid-template-columns con repeat(auto-fit, minmax(250px, 1fr)) para hacer que las cards sean responsivas y se adapten a la pantalla.
3. Márgenes y Espaciado: Usá gap para dar espacio entre las cards y padding para que no queden pegadas a los bordes del contenedor.
4. Interactividad: Añadí un efecto de hover con transform: translateY(-5px) para que las tarjetas se eleven al pasar el mouse.
5. Box Model: Aplicá padding, border y box-shadow para darle estructura a las cards y hacerlas más atractivas visualmente.

Ejercicio práctico #2

Media Queries

Implementar una media queries en la sección “Contacto” que ajuste el diseño de la página para dispositivos móviles, asegurándote de que el layout de las cards de reseñas se adapte a pantallas más pequeñas

1. **Estructura HTML:** Asegurate de tener una estructura básica en el HTML con la sección de "Contacto" y el formulario correspondiente. Usá etiquetas como <section>, <form>, <input>, y <textarea> para los campos del formulario.
2. **Media Queries:** Implementá las media queries en el archivo styles.css usando @media (max-width: 768px) para ajustar el diseño en pantallas más chicas, como tablets o celulares.
3. **Ajustar Cards:** Utilizá grid-template-columns: 1fr en las cards de reseñas para que se apilen en pantallas pequeñas, lo que mejora la legibilidad y el uso en dispositivos móviles.
4. **Formulario Adaptable:** Cambiá el padding y los tamaños de los campos del formulario para que se vean más compactos y legibles en dispositivos con menos espacio.
5. **Probar el Diseño:** Probá el diseño con las herramientas de desarrollador para asegurarte de que las media queries se activan correctamente y el contenido se adapta a diferentes tamaños de pantalla.

Buenos Aires aprende

Agencia de Habilidades para el Futuro

Buenos Aires Ciudad