Esta clase va a ser

• grabada

Certificados oficialmente por

PedidosYa

CODERHOUSE

Clase 07. Desarrollo Web

GRIDS II

Certificados oficialmente por

CODERHOUSE

CLASE N°06

Glosario

CSS Grid: es el sistema de maquetación más potente que hay disponible. Se trata de un sistema en 2D que permite definir filas y columnas (a diferencia de, por ejemplo, Flexbox, el cual funciona en una única dimensión).

Objetivos de la clase

- **Conocer** qué es un Diseño Responsive
- **Conocer** qué es un Diseño Mobile First
- **Aplicar** Media Queries
- **Generar** diseños utilizando Grids y Flexbox

Temario

06

Grids

- ✓ Grids
- ✓ Grids y Flexbox
- ✓ Implementar Grids

07

Grids II

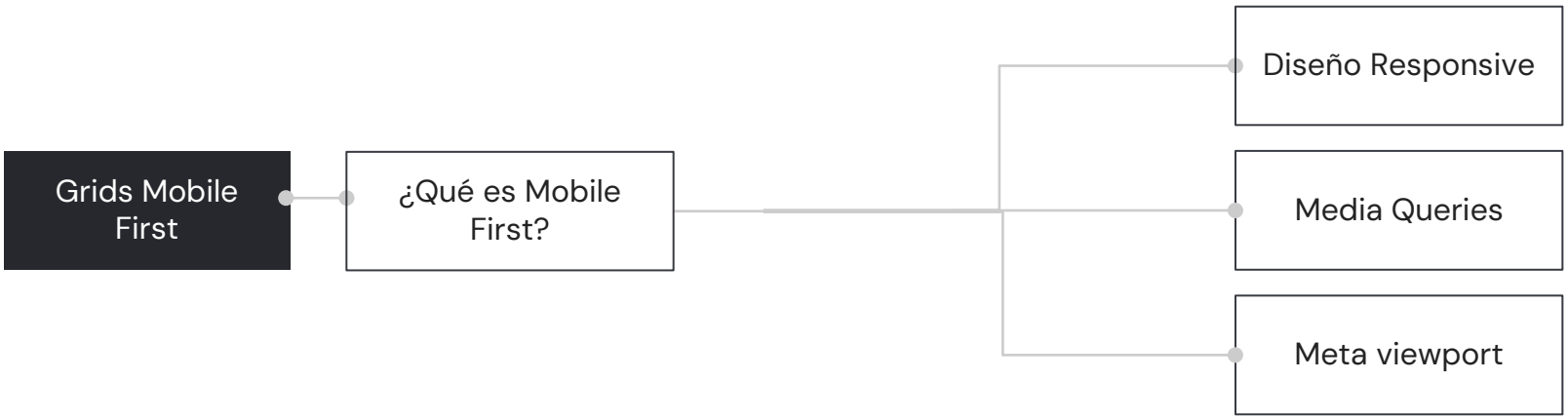
- ✓ Diseño responsive
- ✓ Media queries
- ✓ Mobile first
- ✓ Grids+Flex+@Media

08

Animaciones, transformaciones y transiciones.

- ✓ Gradientes
- ✓ Transformaciones
- ✓ Transiciones
- ✓ Animaciones

MAPA DE CONCEPTOS



Diseño responsive

Diseño responsive

El diseño responsive se refiere a la idea de que un sitio web debería mostrarse igual de bien en todo tipo de dispositivo, desde monitores de pantalla panorámica hasta teléfonos móviles.

Es un enfoque para el diseño y desarrollo web que elimina la distinción entre la versión amigable para dispositivos móviles de un sitio web y su contraparte de escritorio. Con un diseño responsive ambos son lo mismo.

Diseño responsive

En una página responsive, el sistema detecta automáticamente el ancho de la pantalla y, a partir del mismo, adapta todos los elementos de la página, desde el tamaño de letra hasta las imágenes y los menús, ofreciendo al usuario la mejor experiencia posible.
Para lograr esta adaptación, usamos **media queries**.

CODERHOUSE

Media Queries

Media Queries

El **diseño responsive** se logra a través de "Media Queries" de CSS.

👉 Pensemos en las Media Queries como una **forma de aplicar condicionales a las reglas de CSS.**

👉 Estas últimas le dicen al navegador **qué reglas debe ignorar o aplicar** dependiendo del dispositivo del usuario.

Ejemplo de Media Queries

Si quisiera que en las pantallas extra pequeñas (xs) el color de fondo que aplica la clase **.miestilo** sea rojo, y para el resto de tamaños sea verde, podrías hacer:

```
.miestilo {
        background-color: green;
}
@media (max-width: 480px) {
     .miestilo {
          background-color: red;
     }
}
```

Ejemplo de Media Queries

Si quisiera variar la alineación del texto que se aplica en una clase, a partir de las pantallas tipo escritorio:

```
.miestilo {
      text-align: center;
}
@media (min-width: 992px) {
     .miestilo {
          text-align: left;
     }
}
```

Ejemplo de Media Queries

Puedes también modificar el cuerpo de texto, si lo utilizas en px para diferentes pantallas:

```
@media (min-width: 600px) {
     div.example {
          font-size: 60px;
     }
@media (max-width: 599px) {
     div.example {
          font-size: 30px;
     }
```

Recuerda que esto no es necesario si utilizas el font-size en “em”

Operador “AND”

Puedes sumar diferentes indicaciones con las Media Queries, las cuales se combinan con el operador “and”. En este caso, el estilo que definido se reproducirá en pantallas que van de 400px a 700px:

```
@media (min-width: 400px) and (max-width: 700px){
     .miestilo {
          text-align: left;
     }
```

Orientación

En este caso, solo se reproducirá el estilo si la ventana tiene un ancho de 700px o más, y la pantalla está en formato horizontal.

```
@media (min-width: 700px) and (orientation:
landscape) {
     .miestilo {
          text-align: left;
     }
```

Ejemplo de Media Queries

Modificación de menú de posición horizontal en pantallas grandes, a vertical en pantallas chicas.

HTML

```html
<div class="topnav">
  <a href="#">Home</a>
  <a href="#">Nosotros</a>
  <a href="#">Contacto</a>
</div>
```

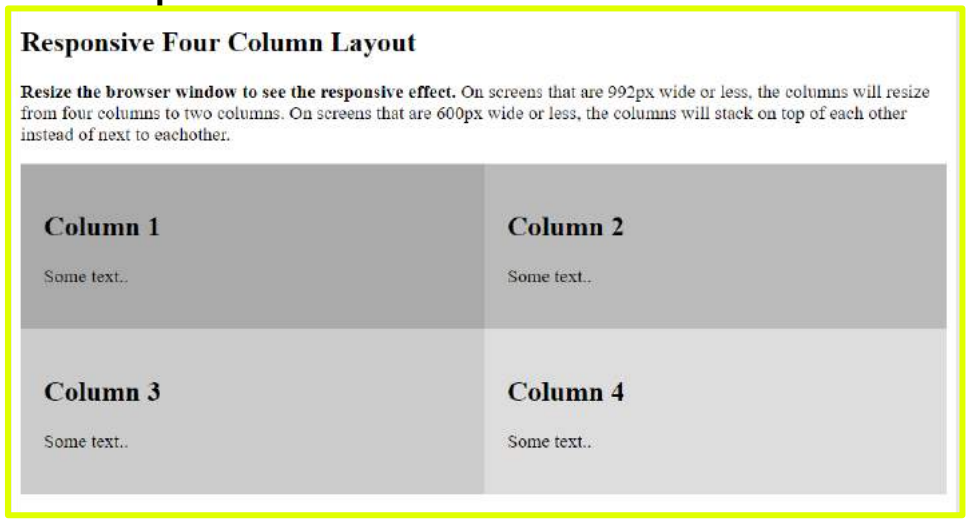
CSS

```css
/* Top navigation bar */
.topnav {
  overflow: hidden;
  background-color: #333;}

/* Topnav links */
.topnav a {
  display: inline-block;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none; }

@media screen and (max-width: 600px) {
.topnav a {
    display:block;
    width: 100%;
  }
}
```

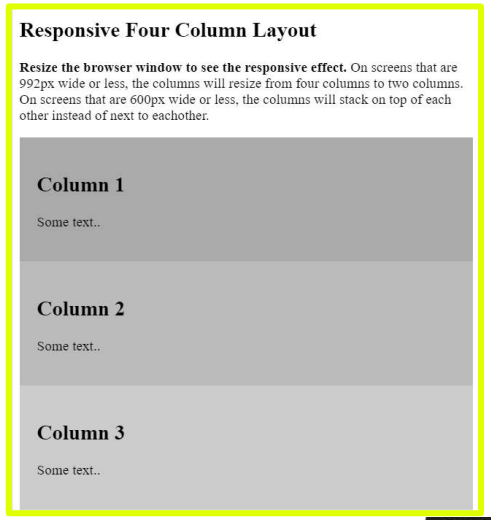
Ejemplo de Media Queries

Puedes modificar la distribución de las columnas para que en pantallas pequeñas se vean una abajo de la otra:

Desktop



Mobile



Ver el código

Media Queries

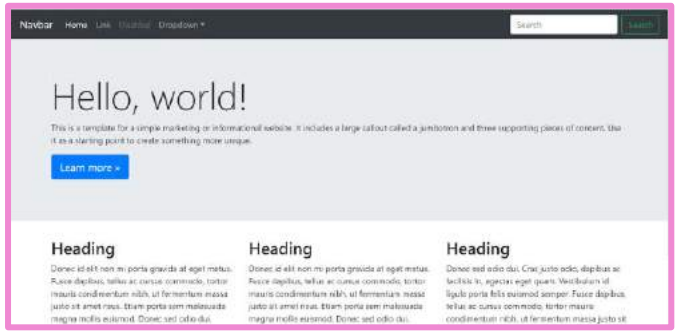
Puedes llegar a hacer cosas muy avanzadas y personalizar completamente el aspecto de una web según el tamaño del dispositivo.

- ✓ Cambiar el tamaño y la posición de una imagen.
- ✓ Modificar la posición de cualquier elemento.
- ✓ Cambiar el tamaño de letra, la fuente o su color.
- ✓ Aplicar combinaciones de estilos avanzados.

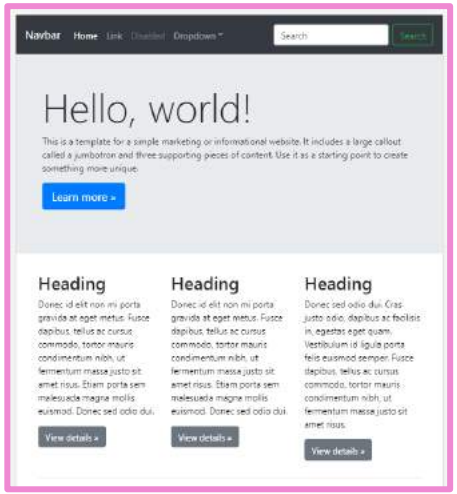
¡Cualquier cosa que se te ocurra! 🚀

Ejemplo

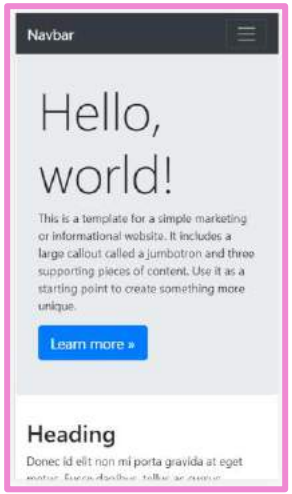
Desktop



Tablet



Mobile

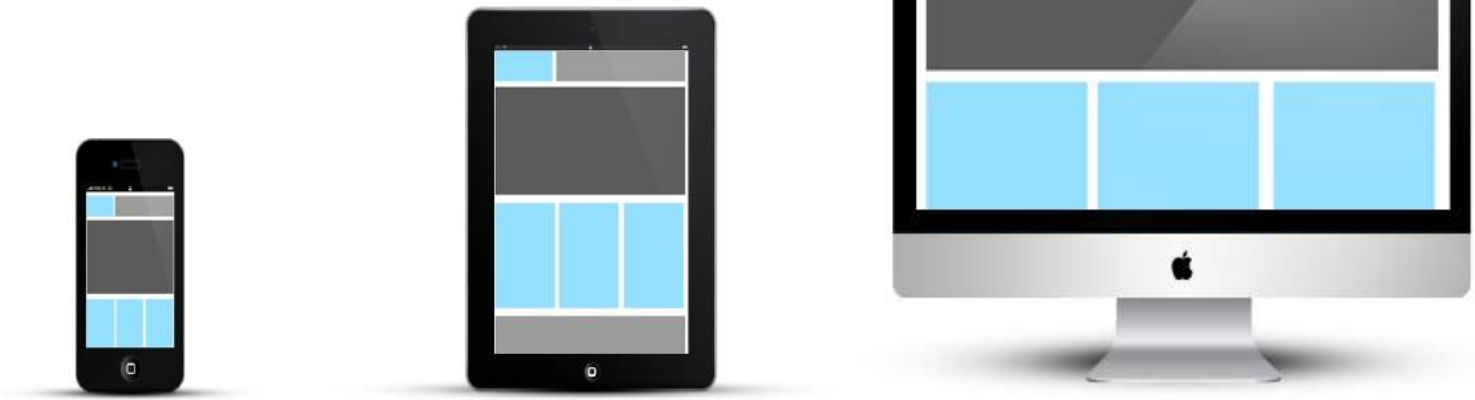


CODERHOUSE

Enfoque
mobile first

Mobile First

Es un enfoque de desarrollo que implica crear primero el código para los dispositivos más pequeños que los usuarios probablemente tengan, como teléfonos o tabletas. Trabajar en el dispositivo más pequeño y luego acumular desde allí todo en el mismo código y el mismo proyecto, en lugar de uno nuevo para cada tamaño de pantalla.



PARA RECORDAR

Mobile First

Se recomienda:

- 👉 Primero trabajar el código para que se reproduzca perfectamente en un teléfono.
- 👉 Segundo, ajustar para que se ejecute en una tableta.
- 👉 Por último, trabajar en un dispositivo de escritorio.

Mobile First y Media Queries

Cualquier estilo dentro del siguiente media query se ejecutará cuando el tamaño de la pantalla sea de al menos 768px de ancho -tablet portrait iPad Mini- pero no cuando el tamaño de la pantalla sea menor:

```
Estilos Mobile
@media screen and (min-width: 768px) {
    .body {
            background-color: #000000;
      }
```

CODERHOUSE

Breakpoints: Categorías

Tamaño	Dispositivo
320px	Para dispositivos con pantallas pequeñas, como los teléfonos en modo vertical
480px	Para dispositivos con pantallas pequeñas, como los teléfonos, en modo horizontal
600px	Tabletas pequeñas, como el Amazon Kindle (600×800) y Barnes & Noble Nook (600×1024), en modo vertical
768px y 1023px	Tabletas de diez pulgadas como el iPad (768×1024), en modo vertical
1024px	Tabletas como el iPad (1024×768), en modo horizontal, así como algunas pantallas de ordenador portátil, netbook, y de escritorio
1200px	Para pantallas panorámicas, principalmente portátiles y de escritorio

Grids Mobile First

Grids Mobile First

Estructura HTML paso a paso

- Lo primero es **asignarle** a nuestro contenedor la propiedad de display: grid;
- Luego, **número de columnas y filas que tendrá nuestra grilla**, y un espacio de separación.
- **Definir** el área que ocupará cada caja de nuestro contenedor, primero le asignaremos un nombre y un color característico.
- **Definir** cómo queremos que cada área sea acomodada en nuestro layout

Grid Mobile First

Retomemos el ejemplo de la clase anterior. Veamos la estructura HTML y CSS que declaramos para un dispositivo móvil (tipo teléfono)

HTML

```
<div id="grilla">
            <header class="border">Header</header>
            <section id="productos" class="border">Section</section>
            <section id="servicios" class="border">Section</section>
            <nav class="border">Navegación</nav>
            <aside class="border">Aside</aside>
            <footer class="border">Pie de página</footer>
</div>
```

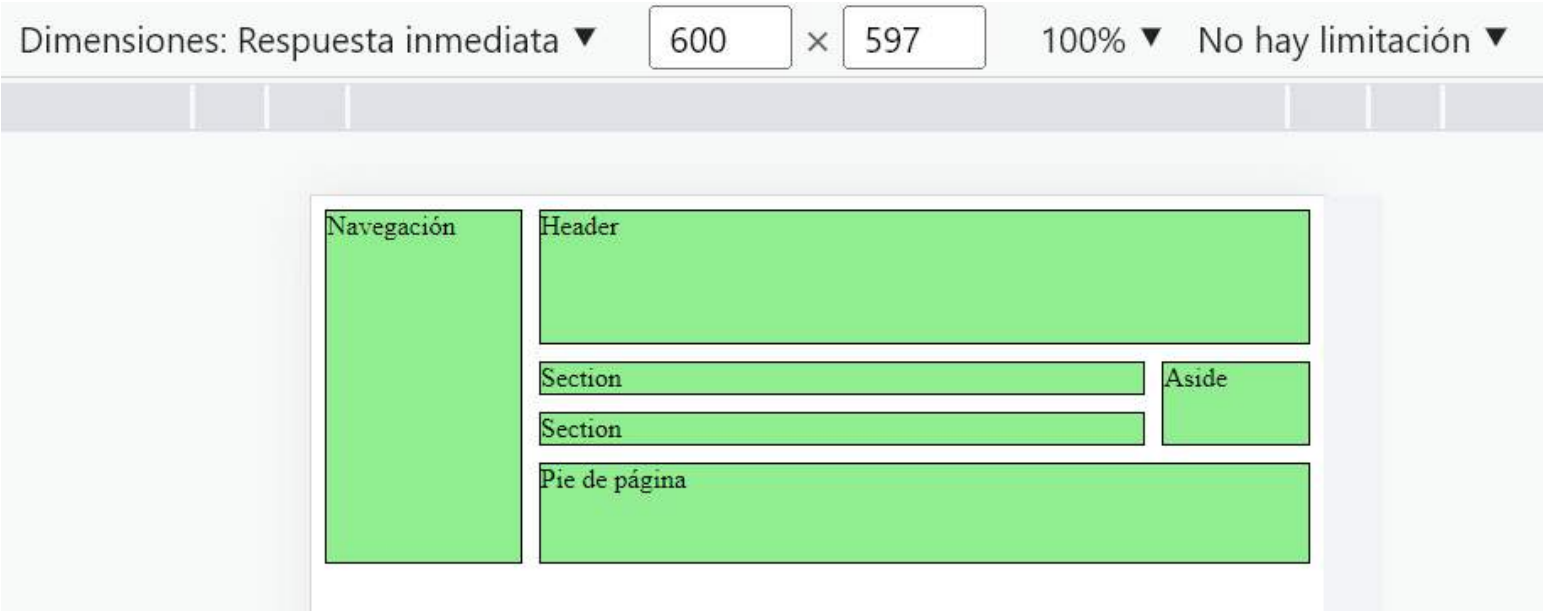
Grid Mobile First

CSS

```css
#grilla {
   display: grid;
   grid-template-areas:
     "nav header header"
     "nav productos publicidad"
     "nav servicios publicidad"
     "nav footer footer";
   grid-template-rows: 80px 1fr 1fr 60px;
   grid-template-columns: 20% auto 15%;
}
.border {
   border: 1px solid black;
   background-color: lightgreen;
}
```

```css
header {
   grid-area: header;
}
footer {
   grid-area: footer;
}
section#productos {
   grid-area: productos;
}
section#servicios {
   grid-area: servicios;
}
nav {
   grid-area: nav;
}
aside {
   grid-area: publicidad;
}
```

Resultado


Dimensiones: Respuesta inmediata
600
×
597
100%
No hay limitación
Navegación
Header
Section
Aside
Section
Pie de página

Tablet

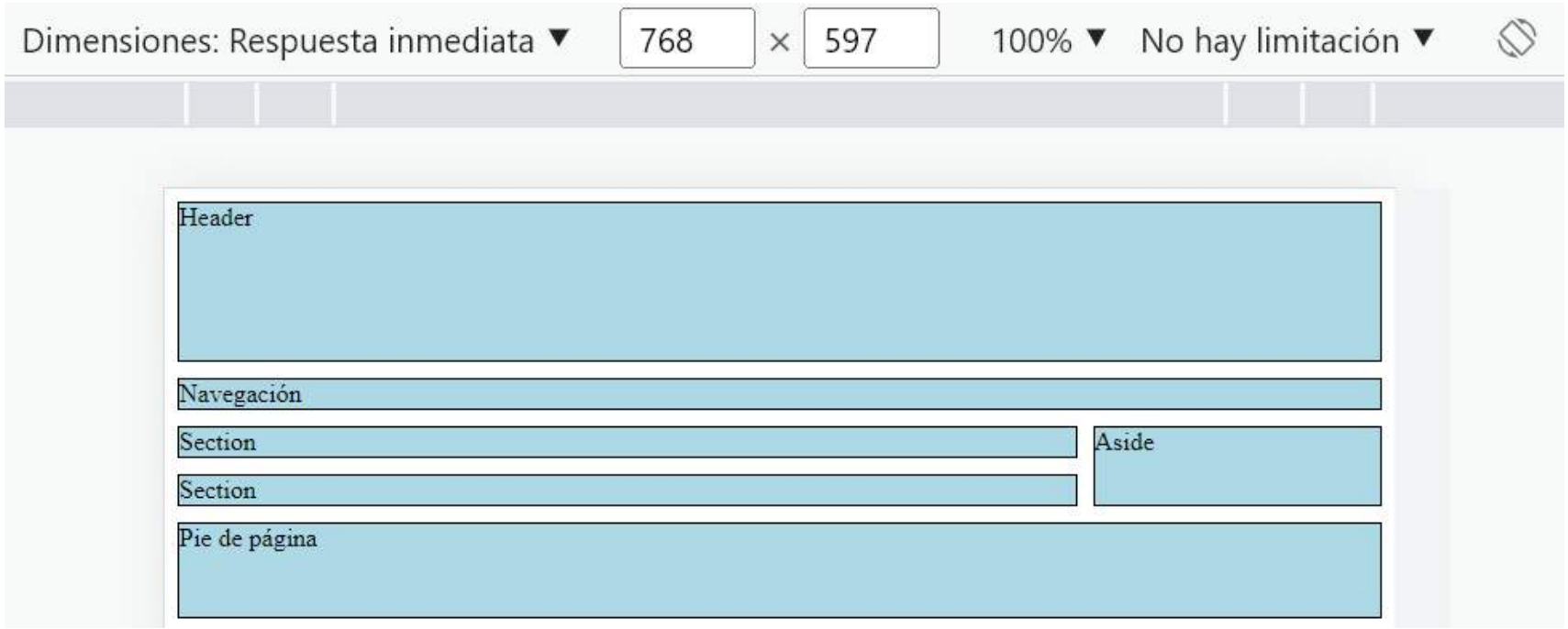
Siguiendo el ejemplo de las Grillas por áreas

👉 Para la versión tablet lo primero que hacemos es cambiar la disposición de las columnas y filas de nuestro Grid.

👉 Luego cambiamos la disposición de los ítems, usando grid-template-areas.

```
@media screen and (min-width: 768px) {
       #grilla {
              grid-template-rows:100px 1fr 1fr 1fr 60px;
              grid-template-columns:repeat(4, 1fr);
              grid-template-areas:
              "header header header header"
              "nav nav nav nav"
              "productos productos productos publicidad"
              "servicios servicios servicios publicidad"
              "footer footer footer footer";
      }
      .border {
              background-color: lightblue;
                                  }
}
```

Resultado



Desktop

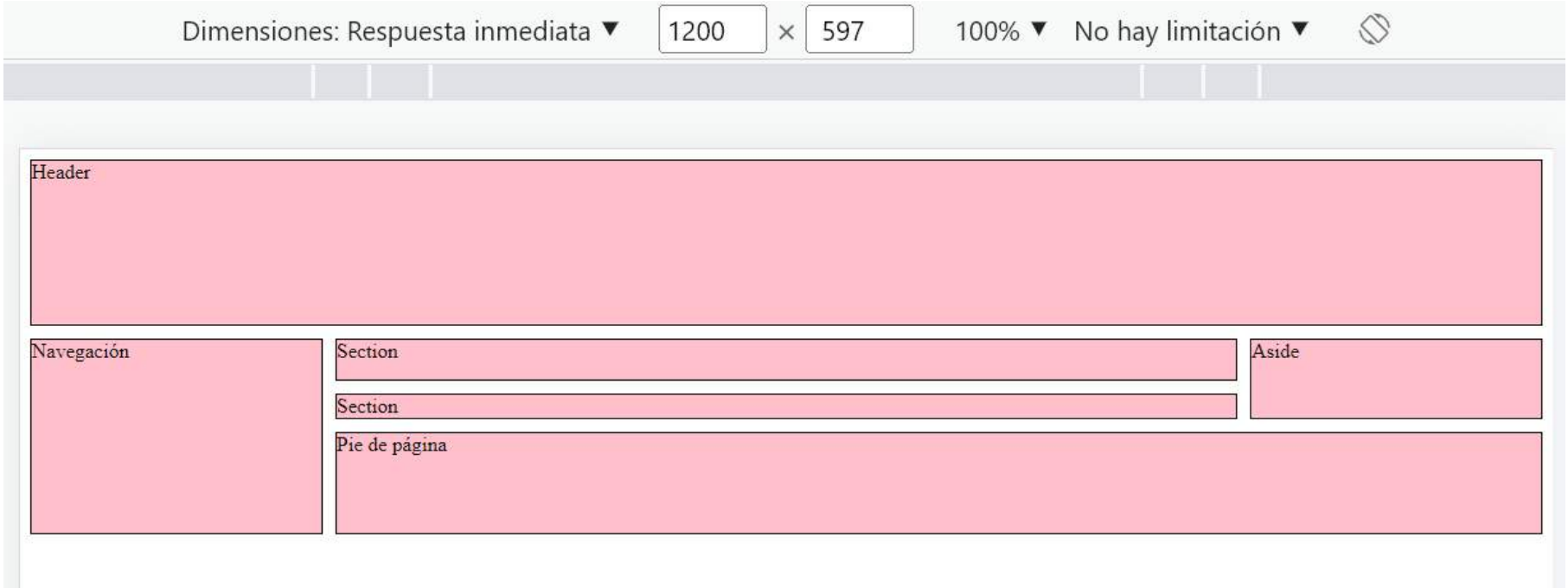
Siguiendo el ejemplo de las Grillas por áreas

👉 Cambiamos la disposición de la grilla.

👉 Y ahora una vez más cambiamos la disposición de los ítems

```
@media screen and (min-width: 1200px) {
       #grilla {
              grid-template-rows:130px 5fr 3fr 80px;
              grid-template-columns:repeat(5, 1fr);
              grid-template-areas:
              "header header header header header"
              "nav productos productos productos publicidad"
              "nav servicios servicios servicios publicidad"
              "footer footer footer footer footer";
      }
      .border {
              background-color: pink;
                                  }
}
```

Resultado



¿Preguntas?

#CoderTip: Ingresa al siguiente link y revisa el material interactivo que preparamos sobre **Preguntas Frecuentes,** estamos seguros de que allí encontrarás algunas respuestas.

Break

¡10 minutos y volvemos!

Meta Viewport

Las páginas optimizadas para diferentes dispositivos deben incluir la etiqueta <meta> viewport en el encabezado del documento HTML. Una etiqueta <meta> viewport da al navegador las instrucciones sobre cómo controlar las dimensiones y el ajuste a escala de la página.

✓ Usa la etiqueta <meta> viewport para controlar el ancho y el ajuste de la ventana de visualización del navegador.

✓ Incluye width=device-width para hacer coincidir el ancho de la pantalla en píxeles independientes del dispositivo.

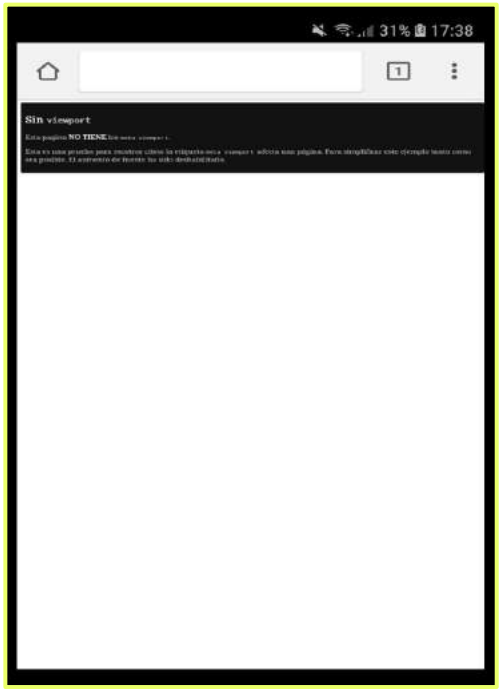
✓ Usa initial-scale=1 para establecer una relación de 1:1 entre los píxeles CSS y los píxeles independientes del dispositivo.

Meta Viewport

El uso del valor de width=device-width indica a la página que debe hacer coincidir el ancho de la pantalla en píxeles independientes del dispositivo. Esto permite que la página realice el reprocesamiento del contenido para adaptarlo a diferentes tamaños de pantalla.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0"/>
```

Meta Viewport



CODERHOUSE

Grids + Flex + @Media

Ejemplo en vivo

¡Vamos a practicar lo visto!

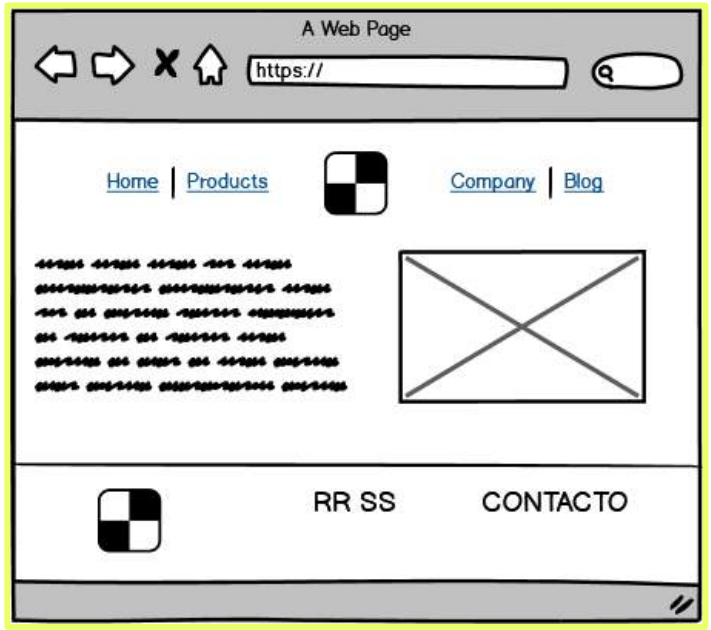
Generando un diseño responsive, haciendo uso de Grids para el Layout & Flexbox para los componentes.

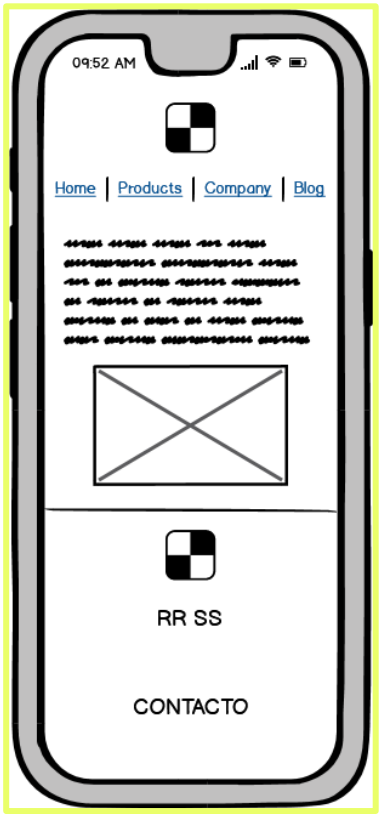
Vamos a combinar todo...

La idea es generar un diseño responsive haciendo uso de grids para el layout + flexbox para los componentes.
Para eso, vamos a tener como referencia los siguientes diseños para vista mobile y desktop.

CODERHOUSE

Wireframes





CoderTips

Si querés ver el código completo, podés buscarlo en la carpeta de recursos de esta misma clase, en Google Drive.

#Coderalert

Ingresa al manual de prácticas y realiza la tercera actividad “Aplicando Grids”. Ten en cuenta que el desarrollo de la misma será importante para la resolución del Proyecto Final.

Actividad Nº 5

Aplicando Grids

Descripción de la actividad.

- ✓ Generar que el index y una página más a elección de nuestro proyecto sea totalmente responsive utilizando grids para el layout, flexbox para los componentes y box modeling para terminar de acomodar los elementos. Es necesaria la utilización de media queries.

Recomendaciones:

- En el html, generar estructura de grid-contenedor-padre e grid-item-hijo para poder trabajar desde el CSS con grid-area.
- Dentro de esos grid-item-hijo deberemos agregar etiquetas para generar los componentes (ej: nav - footer - content - etc) a los cuales acomodaremos aplicando flexbox.
- Si es necesario, aplicamos box modeling para terminar de acomodar y generar nuestro layout completo para la vista desktop.
- Deberás repetir este proceso pero dentro de una media querie mobile.

Podrás encontrar un ejemplo en la carpeta de clase.

¿Preguntas?

Encuesta

Por encuestas de Zoom

¡Terminamos el módulo 3: Responsive!

Cuéntanos qué temas te resultaron más complejos de entender. **Puedes elegir más de uno**. Vamos a retomar aquellos temas que resultaron de mayor dificultad en el próximo AfterClass.

CODERHOUSE

Muchas gracias.

Resumen
de la clase hoy

- ✓ Grids + Flexbox + @media.
- ✓ Nuevas formas de modelar Grids

Opina y valora esta clase

CODERHOUSE

#DemocratizandoLaEducación

CODERHOUSE