

HTML Lists

- Listas No Ordenadas
- Listas Ordenadas
- Listas de Descripción
- Listas Anidadas

Tablas

- Estructura Básica de una Tabla
- Atributos de las Tablas
- Tablas con Estilo CSS
- HTML Table Borders
 - Uso del Atributo `border`
 - Uso de CSS para Bordes
 - Propiedades CSS para Bordes
 - Ejemplo de Bordes Diferentes

HTML Table Sizes

- Diferentes Unidades de Medida
- Ancho de Columnas Específicas
- HTML Table Row Height

HTML Table Headers

- HTML Table Colspan & Rowspan
- HTML Tabla - Rowspan

HTML Block and Inline Elements

- Elementos de Nivel de Bloque
- Elementos en Línea

HTML Forms

- HTML Input Types

CSS

Selectores CSS

- Selectores básicos
- Combinadores

Box-sizing, display, visibilidad y posicionamiento

Propiedad `box-sizing`

Contenido (Content)

- Relleno (Padding)
- Borde (Border)
- Margen (Margin)
- Outline

Diferencias entre `outline` y `border`

Propiedades de `outline`

1. `outline-width`
2. `outline-style`

Outline-offset

Padding

Formas de usar `padding`:

- Propiedades individuales:
- Consideraciones
- Compatibilidad:

box-sizing

Valores de `box-sizing`:

Ejemplos:

¿Por qué es útil `box-sizing`?

max-width

Valores comunes:

Unidades Absolutas

Ejemplos de unidades absolutas:

Cuándo usarlas:

Unidades Relativas

Ejemplos de unidades relativas:

Cuándo usarlas:

Diferencias clave:

`inherit`

¿Cómo funciona?

Casos de uso comunes:

Diferencias con otros valores:

Propiedades heredables comunes

Propiedades no heredables

Margin Collapse

¿Cuándo ocurre el margin collapse?

¿Cómo se calcula el margen colapsado?

Taller

Overflow

Valores de `overflow`

`text-overflow`

Valores de `text-overflow`

Requisitos para que `text-overflow` funcione

Tipografía en CSS

`color`

Propiedad `font-family` en CSS

Tipos de fuentes genéricas en CSS

Propiedad `font-size` en CSS

Unidades de medida en `font-size`

`font-size:smaller;`

Lista de palabras clave absolutas disponibles en `font-size`

Conclusión

Propiedad `font-style` en CSS

Valores de `font-style`

Propiedad `font-weight` en CSS

Manejo de fondos

Resumen:

Formato de texto

```
text-align: center;  
text-transform: uppercase;  
text-shadow: 2px 2px 2px #ced4da;  
letter-spacing: 3px;  
word-spacing: 10px;  
text-decoration: overline;
```

Conclusión

`text-indent: Value;`

Valores comunes

Especificidad

Reglas de especificidad

Reglas adicionales:

Resumen de cómo calcular la especificidad:

Ejercicio

Propiedad `display`

Cascada y Especificidad

Ejemplo de especificidad

Herencia

¿Qué propiedades se heredan?

Propiedades que **NO** se heredan

Taller de especificidad

Objetivo:

Duración:

Parte 1: Introducción Teórica a la Especificidad (15 minutos)

Parte 2: Ejemplos Prácticos (30 minutos)

Parte 3: Ejercicios Prácticos (45 minutos)

Ejercicio 1: Calculando la Especificidad

Ejercicio 2: Resolviendo Conflictos de Especificidad

Parte 4: Desafío Final (30 minutos)

Desafío: Diseñando una Página Completa con Estilos Conflictivos

Parte 5: Revisión y Discusión (30 minutos)

Conclusión:

Critical Rendering Path.

Etapas de la Ruta Crítica de Representación

Rol de JavaScript

Resumen

Unidades de medida

1. Unidades Absolutas

2. Unidades Relativas

Limitar medidas y porcentajes

Limitando Ancho y Alto

Usando Limitaciones en Layouts Responsivos

Limitando Font Sizes en CSS

Otros casos de uso

Nueva Propiedad `text-overflow: ellipsis`

Condiciones para que `text-overflow: ellipsis` funcione

Cambio en la Propiedad `overflow`

Posicionamiento de cajas

Position: static (Posicionamiento Estático)

Position: relative (Posicionamiento Relativo)

Position: absolute (Posicionamiento Absoluto)

Position: fixed (Posicionamiento Fijo)

Position: sticky (Posicionamiento Adhesivo)

Cuándo Usar Cada Tipo

Taller

z-index

Conceptos Clave del `z-index`

Ejercicio Posicionamiento

Ejercicio de posicionamiento Menú Básico

Pseudoclases

Taller

Taller 2

Pseudelementos

Introducción a los Pseudelementos en CSS

Tabla de Pseudelementos en CSS

Combinadores en CSS

Tipos de Combinadores en CSS

Combinador Descendiente (`>`)

Combinador Hijo (`>`)

Combinador de Adyacencia (`+`)

Combinador de Hermanos Generales (`~`)

Variables

Uso de variables:

Ejercicio

ResponsiveDesign

¿Cómo Funciona el Responsive Design?

Viewport <https://whatismyviewport.com/>

Importancia del Viewport en Diseño Web

Meta Viewport en HTML

Imagenes responsivas

Técnicas para Crear Imágenes Responsivas

CSS con `max-width` y `height: auto`

Atributo `srcset` en HTML

márgenes de fondo con `background-size: cover` o `contain`

Formato de imagen adaptable (como SVG)

Uso de `picture` para diferentes versiones de imagen

Ventajas de las Imágenes Responsivas

MediaQueries

¿Cómo Funcionan los Media Queries?

Usos Comunes de Media Queries

Diseño adaptable a diferentes tamaños de pantalla

Orientación de pantalla

Resolución de pantalla

Tipos de dispositivo

Componentes de un Media Query

Tipos de medio

Características de Medios (media feature)

`@media (hover: hover)`

¿Por Qué Usar `@media (hover: hover)` ?

Otras Opciones con `hover`

`@media (max-width: 1250px)`

¿Cuándo Usar `max-width` en Media Queries?

`@media (color)`

¿Cuándo Usar `@media (color)` ?

Creación de *media queries* complejas

Combinación `de` múltiples tipos o características

Explicación

Limitar los estilos a los dispositivos con una pantalla

Explicación

¿Cuándo Usar `screen and (min-width: 30em) and (orientation: landscape)` ?

Múltiples consultas

Explicación

¿Cuándo Usar Esta Media Query?

Invertir el significado de una consulta

¿Cómo Funciona `not all and (monochrome)` ?

¿Cuándo Usar `not all and (monochrome)` ?

Media queries Nivel 4

Nuevos Ranges (Rangos de Comparación)

Media Features como Rangos

Uso de Palabras Clave: `or`, `and`, y `not`

Media Queries de `scripting` y `pointer`

Uso de `any-hover` y `any-pointer`

Soporte de Media Queries `update`

Diseño Fluido con Unidades Relativas

Diseño con Flexbox y Grid

Diseño Móvil-Primero (Mobile-First)

Contenedores Flexibles

Uso de `Picture` Element para Imágenes Adaptables

Componentes Condicionales

URL de Interes

Practica Media Queries

FlexBox

Conceptos Básicos de Flexbox

Propiedades del Contenedor Flex

display

flex-direction

justify-content

align-items

flex-wrap

Propiedades de los Elementos Flex

flex

align-self

Taller

html

Aplicando Flex en el CSS

Aplicando Flex-Direction

Taller Nro 2

Taller 3 - <https://flexboxfroggy.com/#es>

Taller 4 - <https://mastery.games/flexboxzombies/>

Grid <https://css-tricks.com/snippets/css/complete-guide-grid/>

Filas y Columnas

Grid container

Grid Item

Grid Cell (Celda)

Grid Lines (Líneas)

Ejercicio

Propiedades GRID

`grid-template-columns`

`grid-template-rows`

Posicionamiento grid items por lines

`grid-column-start`

`grid-column-end`

`grid-row-start y grid-row-end`

Modo Abreviado

`span`

Uso de `span` en CSS Grid

Espaciadores o Gutters (Se definen en el container)

`row-gap`

`column-gap`

`grid-auto-flow`

`grid-auto-rows`

dense en CSS GRID

`grid-template-areas`

Cómo Funciona `grid-template-areas`

Ejercicio

`grid-area`

Asignar nombres a áreas específicas de una cuadrícula

HTML Lists

Listas No Ordenadas

Una lista no ordenada se utiliza para agrupar un conjunto de elementos sin un orden particular. Los elementos de una lista no ordenada suelen estar marcados con viñetas.

```
<ul>
  <li>Elemento 1</li>
  <li>Elemento 2</li>
  <li>Elemento 3</li>
</ul>
```

Ejemplo

```
<ul>
  <li>Manzanas</li>
  <li>Bananas</li>
  <li>Cerezas</li>
</ul>
```

Listas Ordenadas

Una lista ordenada se utiliza cuando el orden de los elementos es importante. Los elementos de una lista ordenada suelen estar marcados con números.

```
<ol>
  <li>Primer Elemento</li>
  <li>Segundo Elemento</li>
  <li>Tercer Elemento</li>
</ol>
```

```
<ol>
  <li>Precalentar el horno a 180°C.</li>
  <li>Mezclar harina y azúcar en un tazón.</li>
  <li>Hornear durante 20 minutos.</li>
</ol>
```

Listas de Descripción

Una lista de descripción se utiliza para crear una lista de términos y sus descripciones.

```
<dl>
  <dt>Término 1</dt>
  <dd>Descripción del Término 1.</dd>
  <dt>Término 2</dt>
  <dd>Descripción del Término 2.</dd>
</dl>
```

```
<dl>
  <dt>HTML</dt>
  <dd>Lenguaje de Marcado de Hipertexto, el lenguaje de marcado estándar para documentos diseñados para ser visualizados en un navegador web.</dd>
  <dt>CSS</dt>
  <dd>Hojas de Estilo en Cascada, un lenguaje de estilo utilizado para describir la presentación de un documento escrito en HTML o XML.</dd>
</dl>
```

Listas Anidadas

También puedes anidar listas dentro de otras listas. Por ejemplo:

```
<ul>
  <li>Frutas
    <ul>
      <li>Manzana</li>
      <li>Banana</li>
    </ul>
  </li>
  <li>verduras
    <ul>
      <li>Zanahoria</li>
      <li>Brócoli</li>
    </ul>
  </li>
</ul>
```

Tablas

Las tablas en HTML son una herramienta esencial para organizar y presentar datos de manera estructurada en filas y columnas.

Estructura Básica de una Tabla

La estructura básica de una tabla en HTML incluye varios elementos:

- `<table>`: Define el inicio y el final de la tabla.
- `<tr>`: Define una fila de la tabla.
- `<th>`: Define una celda de encabezado, que generalmente se utiliza para títulos de columnas.
- `<td>`: Define una celda de datos en la tabla.
- `<caption>` define título de una tabla

Ejemplo

```
<table>
  <tr>
    <th>Nombre</th>
    <th>Edad</th>
    <th>Ciudad</th>
```

```

</tr>
<tr>
    <td>Juan</td>
    <td>25</td>
    <td>Bogotá</td>
</tr>
<tr>
    <td>Ana</td>
    <td>30</td>
    <td>Medellín</td>
</tr>
</table>

```

Nombre	Edad	Ciudad
Juan	25	Bogotá
Ana	30	Medellín

Atributos de las Tablas

Puedes utilizar varios atributos para mejorar el aspecto y la funcionalidad de las tablas:

- `border`: Añade un borde a la tabla.
- `cellpadding`: Establece el espacio entre el contenido de la celda y el borde de la celda.
- `cellspacing`: Establece el espacio entre las celdas.

```

<table border="1" cellpadding="10" cellspacing="0">
    <tr>
        <th>Producto</th>
        <th>Precio</th>
    </tr>
    <tr>
        <td>Manzana</td>
        <td>$1</td>
    </tr>
    <tr>
        <td>Banana</td>
        <td>$0.50</td>
    </tr>
</table>

```

Tablas con Estilo CSS

```

<style>
    table {
        width: 100%;
        border-collapse: collapse;
    }
    th, td {
        border: 1px solid black;
        padding: 8px;
    }

```

```
        text-align: left;
    }
    th {
        background-color: #f2f2f2;
    }
</style>
```

HTML Table Borders

Las tablas en HTML pueden tener bordes que ayudan a definir visualmente las celdas y las filas, mejorando la legibilidad y la organización de los datos. Hay varias maneras de agregar bordes a las tablas en HTML, utilizando atributos HTML o CSS.

Uso del Atributo `border`

Una forma rápida de agregar bordes a una tabla es utilizando el atributo `border` en la etiqueta `<table>`. Este atributo acepta valores numéricos que especifican el grosor del borde.

```
<table border="1">
  <tr>
    <th>Nombre</th>
    <th>Edad</th>
  </tr>
  <tr>
    <td>Juan</td>
    <td>25</td>
  </tr>
  <tr>
    <td>Ana</td>
    <td>30</td>
  </tr>
</table>
```

Uso de CSS para Bordes

El método más flexible y moderno es usar CSS para definir bordes. Puedes aplicar bordes a la tabla, a las celdas y a las filas.

```
<style>
  table {
    border-collapse: collapse; /* Combina bordes adyacentes */
    width: 100%;
  }
  th, td {
    border: 1px solid black; /* Define el borde de las celdas */
    padding: 8px; /* Espacio interior en las celdas */
    text-align: left; /* Alineación del texto */
  }
  th {
    background-color: #f2f2f2; /* Color de fondo para los encabezados */
  }
</style>
```

```





```

Propiedades CSS para Bordes

Puedes utilizar varias propiedades CSS para personalizar los bordes de las tablas:

- `border-collapse`: Controla cómo se manejan los bordes de las celdas adyacentes. Puede ser `collapse` (combina los bordes) o `separate` (mantiene los bordes separados).
- `border-spacing`: Se utiliza cuando `border-collapse` está configurado en `separate` para definir el espacio entre las celdas.
- `border-style`: Especifica el estilo del borde (por ejemplo, `solid`, `dashed`, `dotted`).
- `border-color`: Define el color del borde.
- `border-width`: Establece el grosor del borde.

Ejemplo avanzado

```

<style>
    table {
        border-collapse: collapse;
        width: 100%;
    }
    th, td {
        border: 2px dashed #4CAF50; /* Borde verde, estilo punteado */
        padding: 10px;
    }
    th {
        background-color: #4CAF50; /* Fondo verde para encabezados */
        color: white; /* Texto en blanco */
    }
</style>

<table>
    <tr>
        <th>Nombre</th>
        <th>Edad</th>
    </tr>
    <tr>
        <td>Juan</td>
        <td>25</td>
    </tr>
</table>

```

```
</tr>
<tr>
    <td>Ana</td>
    <td>30</td>
</tr>
</table>
```

Ejemplo de Bordes Diferentes

Puedes aplicar diferentes estilos de borde a diferentes partes de la tabla:

```
<style>
table {
    border-collapse: collapse;
    width: 100%;
}
th {
    border: 2px solid black; /* Borde negro para encabezados */
    background-color: #f2f2f2;
    padding: 10px;
}
td {
    border: 1px solid grey; /* Borde gris para celdas de datos */
    padding: 8px;
}
</style>

<table>
<tr>
    <th>Producto</th>
    <th>Precio</th>
</tr>
<tr>
    <td>Manzana</td>
    <td>$1</td>
</tr>
<tr>
    <td>Banana</td>
    <td>$0.50</td>
</tr>
</table>
```

HTML Table Sizes

Las tablas en HTML pueden tener diferentes tamaños para cada columna, fila o para toda la tabla.

Para establecer el ancho de una tabla en HTML, puedes añadir el atributo `style` al elemento `<table>`. Esto te permite definir el ancho de la tabla utilizando CSS. Aquí tienes cómo hacerlo:

```

<table style="width: 100%;">
  <tr>
    <th>Nombre</th>
    <th>Edad</th>
    <th>Ciudad</th>
  </tr>
  <tr>
    <td>Juan</td>
    <td>25</td>
    <td>Bogotá</td>
  </tr>
  <tr>
    <td>Ana</td>
    <td>30</td>
    <td>Medellín</td>
  </tr>
</table>

```

Diferentes Unidades de Medida

Puedes especificar el ancho en diferentes unidades de medida:

- **Porcentaje (%)**: El ancho se establecerá en relación al ancho del contenedor padre.
- **Píxeles (px)**: El ancho se establecerá en un valor fijo de píxeles.
- **Viewport Width (vw)**: Una unidad relativa que representa un porcentaje del ancho de la ventana de visualización.

Ejemplos:

En porcentaje

```

<table style="width: 50%;">
  <tr>
    <th>Producto</th>
    <th>Precio</th>
  </tr>
  <tr>
    <td>Manzana</td>
    <td>$1</td>
  </tr>
</table>

```

Ancho en Píxeles

```

<table style="width: 400px;">
  <tr>
    <th>Producto</th>
    <th>Precio</th>
  </tr>
  <tr>
    <td>Banana</td>
    <td>$0.50</td>
  </tr>
</table>

```

Ancho de Columnas Específicas

Si se desea establecer el ancho de columnas específicas, puede hacerlo utilizando el atributo `style` en las celdas de encabezado `<th>` o en las celdas de datos `<td>`.

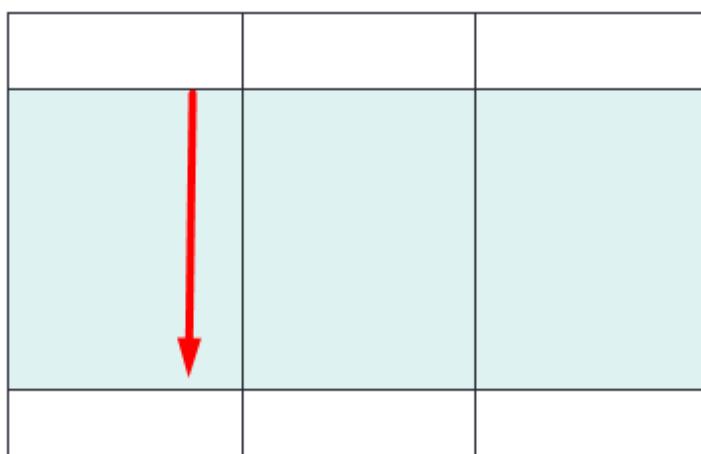
```

<table style="width: 100%;">
  <tr>
    <th style="width: 50%;">Nombre</th>
    <th style="width: 25%;">Edad</th>
    <th style="width: 25%;">Ciudad</th>
  </tr>
  <tr>
    <td>Juan</td>
    <td>25</td>
    <td>Bogotá</td>
  </tr>
  <tr>
    <td>Ana</td>
    <td>30</td>
    <td>Medellín</td>
  </tr>
</table>

```

HTML Table Row Height

Para establecer la altura de una fila específica, añade el atributo `style` en un elemento de fila de la tabla.



A diagram showing a 4x3 grid table. The first row has three empty white cells. The second row has three cells filled with light blue color. A thick red arrow points vertically downwards through the center cell of the second row, indicating the height of that specific row.

Ejemplo obtenido https://www.w3schools.com/html/tryit.asp?filename=tryhtml_table_row_height

```
<!DOCTYPE html>
<html>
<style>
table, th, td {
    border:1px solid black;
    border-collapse: collapse;
}
</style>
<body>

<h2>Set the height of the second row to 200 pixels</h2>

<table style="width:100%">
<tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
</tr>
<tr style="height:200px">
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
</tr>
<tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
</tr>
<tr>
    <td>John</td>
    <td>Doe</td>
    <td>80</td>
</tr>
</table>

</body>
</html>
```

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

HTML Table Headers

EMIL	TOBIAS	LINUS

8:00		
9:00		
10:00		
11:00		
12:00		
13:00		

	MON	TUE	WED	THU	FRI
8:00					
9:00					
10:00					
11:00					
12:00					

DECEMBER		

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>HTML Tables</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f0f0f0;
            padding: 20px;
        }
        .container {
            max-width: 800px;
            margin: 0 auto;
        }
        table {
            width: 100%;
            border-collapse: collapse;
            margin-bottom: 20px;
            background-color: white;
        }
        th, td {
            border: 1px solid #ccc;
            padding: 8px;
            text-align: left;
        }
        th {
            background-color: #f2f2f2;
        }
    </style>
</head>
<body>
    <div class="container">
        <table>
            <thead>
                <tr>
                    <th>EMIL</th>
                    <th>TOBIAS</th>
                    <th>LINUS</th>
                </tr>
            </thead>
            <tbody>
                <tr><td></td></td><td></td></td></td></tr>
                <tr><td></td></td><td></td><td></td></td></tr>
                <tr><td></td><td></td><td></td><td></td></td></tr>
                <tr><td></td><td></td><td></td><td></td></td></tr>
                <tr><td></td><td></td><td></td><td></td></td></tr>
            </tbody>
        </table>
    </div>
</body>
</html>
```

```

        </tbody>
    </table>

    <table>
        <tbody>
            <tr><td>8:00</td><td></td><td></td><td></td></tr>
            <tr><td>9:00</td><td></td><td></td><td></td></tr>
            <tr><td>10:00</td><td></td><td></td><td></td></tr>
            <tr><td>11:00</td><td></td><td></td><td></td></tr>
            <tr><td>12:00</td><td></td><td></td><td></td></tr>
            <tr><td>13:00</td><td></td><td></td><td></td></tr>
        </tbody>
    </table>

    <table>
        <thead>
            <tr>
                <th></th>
                <th>MON</th>
                <th>TUE</th>
                <th>WED</th>
                <th>THU</th>
                <th>FRI</th>
            </tr>
        </thead>
        <tbody>
            <tr><td>8:00</td><td></td><td></td><td></td><td></td><td></td></tr>
            <tr><td>9:00</td><td></td><td></td><td></td><td></td><td></td></tr>
            <tr><td>10:00</td><td></td><td></td><td></td><td></td><td></td></tr>
            <tr><td>11:00</td><td></td><td></td><td></td><td></td><td></td></tr>
            <tr><td>12:00</td><td></td><td></td><td></td><td></td><td></td></tr>
            <tr><td>13:00</td><td></td><td></td><td></td><td></td><td></td></tr>
        </tbody>
    </table>

    <table>
        <thead>
            <tr>
                <th colspan="3" style="text-align: center;">DECEMBER</th>
            </tr>
        </thead>
        <tbody>
            <tr><td></td><td></td><td></td></tr>
            <tr><td></td><td></td><td></td></tr>
        </tbody>
    </table>
</div>
</body>
</html>
```

HTML Table Colspan & Rowspan

HTML Tabla - Colspan

Para hacer que una celda se extienda sobre varias columnas, utiliza el atributo `colspan`

```
<table border="1">
  <tr>
    <th colspan="3">Productos Disponibles</th>
  </tr>
  <tr>
    <th>Nombre</th>
    <th>Precio</th>
    <th>Cantidad</th>
  </tr>
  <tr>
    <td>Manzana</td>
    <td>$1</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Banana</td>
    <td>$0.50</td>
    <td>100</td>
  </tr>
  <tr>
    <td colspan="3">Total de Productos: 150</td>
  </tr>
</table>
```

HTML Tabla - Rowspan

Para hacer que una celda se extienda sobre varias filas, utiliza el atributo `rowspan`

```
<table border="1">
  <tr>
    <th>Producto</th>
    <th>Precio</th>
    <th>Cantidad</th>
  </tr>
  <tr>
    <td rowspan="3">Frutas</td>
    <td>$1</td>
    <td>50</td>
  </tr>
  <tr>
    <td>$0.50</td>
    <td>100</td>
  </tr>
  <tr>
    <td>$2</td>
    <td>75</td>
  </tr>
</table>
```

```

</tr>
<tr>
    <td>Verduras</td>
    <td>$1.50</td>
    <td>30</td>
</tr>
</table>

```

Producto	Precio	Cantidad
Frutas	\$1	50
	\$0.50	100
	\$2	75
Verduras	\$1.50	30

HTML Block and Inline Elements

Cada elemento HTML tiene un valor de visualización predeterminado, dependiendo del tipo de elemento que sea.

Los dos valores de visualización más comunes son bloque (block) y en línea (inline).

Elementos de Nivel de Bloque

Un elemento de nivel de bloque siempre comienza en una nueva línea, y los navegadores añaden automáticamente un espacio (un margen) antes y después del elemento.

Un elemento de nivel de bloque siempre ocupa el ancho completo disponible (se extiende hacia la izquierda y la derecha tanto como puede).

Dos elementos de bloque comúnmente utilizados son: `<p>` y `<div>`.

El elemento `<p>` define un párrafo en un documento HTML.

El elemento `<div>` define una división o una sección en un documento HTML.

```

<address>      <article>      <aside>      <blockquote>  <canvas>      <dd>
<div>          <dl>          <dt>          <fieldset>   <figcaption> <figure>
<footer>        <form>        <h1>-<h6>    <header>    <hr>         <li>
<main>          <nav>          <noscript>   <ol>          <p>          <pre>
<section>       <table>        <tfoot>      <ul>          <video>

```

`<div>`: Se utiliza para agrupar contenido y aplicar estilos o scripts a un conjunto de elementos.

```

<div>
    <h1>Título Principal</h1>
    <p>Este es un párrafo dentro de un div.</p>
</div>

```

`<p>`: Define un párrafo.
`<p>Este es un párrafo de texto.</p>`

`<h1>, <h2>, <h3>, <h4>, <h5>, <h6>`: Se utilizan para definir encabezados de diferentes niveles, donde `<h1>` es el más importante y `<h6>` es el menos.

```
<header>: Define un encabezado para un documento o sección.
```

```
<header>
    <h1>Título de la Página</h1>
</header>
```

```
<footer>: Define un pie de página para un documento o sección.
```

```
<footer>
    <p>© 2024 Mi sitio web</p>
</footer>
```

Elementos en Línea

Un elemento en línea no comienza en una nueva línea.

Un elemento en línea solo ocupa el ancho necesario.

<code><a></code>	<code><abbr></code>	<code><acronym></code>	<code></code>	<code><bdo></code>	<code><big></code>
<code>
</code>	<code><button></code>	<code><cite></code>	<code><code></code>	<code><dfn></code>	<code></code>
<code><i></code>	<code></code>	<code><input></code>	<code><kbd></code>	<code><label></code>	<code><map></code>
<code><object></code>	<code><output></code>	<code><q></code>	<code><samp></code>	<code><script></code>	<code><select></code>
<code><small></code>	<code></code>	<code></code>	<code><sub></code>	<code><sup></code>	<code><textarea></code>
<code><time></code>	<code><tt></code>	<code><var></code>			

HTML Forms

Un formulario HTML se utiliza para recopilar la entrada del usuario. La entrada del usuario se envía con mayor frecuencia a un servidor para su procesamiento.

HTML Input Types

`<input type="button">`: Este tipo de botón se utiliza para crear un botón genérico. No envía un formulario ni realiza ninguna acción por sí mismo, a menos que se le asigne un evento a través de JavaScript.

`<input type="checkbox">`: Este elemento permite al usuario seleccionar uno o más elementos de un conjunto de opciones. Cada casilla de verificación puede ser activada o desactivada de manera independiente.

`<input type="color">`: Permite al usuario seleccionar un color de un selector de color. El valor se representará en formato hexadecimal.

`<input type="date">`: Se utiliza para seleccionar una fecha específica. Este input permite al usuario elegir una fecha a través de un calendario emergente.

`<input type="datetime-local">`: Permite al usuario seleccionar una fecha y una hora, combinando ambos en un solo campo. Esto es útil para situaciones en las que se necesita una marca de tiempo precisa.

`<input type="email">`: Este campo está diseñado para capturar direcciones de correo electrónico. Los navegadores pueden validar automáticamente la entrada para asegurarse de que tenga el formato correcto.

`<input type="file">`: Permite a los usuarios seleccionar y cargar archivos desde su dispositivo. Es comúnmente utilizado en formularios que requieren la carga de documentos, imágenes, etc.

`<input type="hidden">`: Este tipo de input no es visible para el usuario, pero permite almacenar información que se enviará al servidor junto con el formulario.

`<input type="image">`: Utiliza una imagen como un botón de envío para el formulario. Cuando el usuario hace clic en la imagen, se envía el formulario.

`<input type="month">`: Permite al usuario seleccionar un mes y un año específicos. Este tipo de input muestra un selector para elegir un mes del año.

`<input type="number">`: Permite ingresar un número, y puede incluir características como un control de incremento y decrecimiento. Se puede establecer un rango de valores permitidos.

`<input type="password">`: Este campo oculta la entrada del usuario (generalmente para contraseñas) al mostrar asteriscos o puntos en lugar de texto claro.

`<input type="radio">`: Este elemento permite al usuario seleccionar una opción de un grupo de opciones mutuamente excluyentes. Solo se puede seleccionar una opción a la vez.

`<input type="range">`: Se utiliza para crear un control deslizante que permite al usuario seleccionar un valor dentro de un rango específico. Se puede definir el valor mínimo, máximo y el valor por defecto.

`<input type="reset">`: Este tipo de botón reinicia todos los campos de un formulario a sus valores predeterminados cuando se hace clic en él.

`<input type="search">`: Diseñado para capturar consultas de búsqueda. Los navegadores pueden ofrecer características adicionales, como un botón para limpiar el campo.

`<input type="submit">`: Este botón envía el formulario al servidor para su procesamiento. Es el tipo de botón más utilizado en formularios.

`<input type="tel">`: Se utiliza para capturar números de teléfono. Los navegadores pueden optimizar la entrada de este tipo para facilitar la digitación en dispositivos móviles.

`<input type="text">`: Este es un campo de entrada de texto simple que permite al usuario escribir texto. Es el tipo más común de input.

`<input type="time">`: Permite al usuario seleccionar una hora específica. Este input puede mostrar un selector para elegir horas y minutos.

`<input type="url">`: Diseñado para capturar direcciones URL. Los navegadores validan automáticamente la entrada para asegurarse de que tenga el formato correcto.

`<input type="week">`: Permite al usuario seleccionar una semana específica del año. Este tipo de input muestra un selector que facilita la elección de una semana.

Ejemplo

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ejemplo de Formulario con Diferentes Inputs</title>
</head>
```

```
<body>

<h1>Formulario de Ejemplo</h1>
<form>

    <!-- Input type="button" -->
    <input type="button" value="Botón Genérico" onclick="alert('¡Botón presionado!');" ><br><br>

    <!-- Input type="checkbox" -->
    <label>
        <input type="checkbox" name="suscripcion" value="newsletter"> Suscribirse al boletín
    </label><br><br>

    <!-- Input type="color" -->
    <label for="color">Selecciona un color:</label>
    <input type="color" id="color" name="color"><br><br>

    <!-- Input type="date" -->
    <label for="fecha">Selecciona una fecha:</label>
    <input type="date" id="fecha" name="fecha"><br><br>

    <!-- Input type="datetime-local" -->
    <label for="fecha-hora">Selecciona fecha y hora:</label>
    <input type="datetime-local" id="fecha-hora" name="fecha-hora"><br><br>

    <!-- Input type="email" -->
    <label for="email">Correo Electrónico:</label>
    <input type="email" id="email" name="email" required><br><br>

    <!-- Input type="file" -->
    <label for="archivo">Cargar un archivo:</label>
    <input type="file" id="archivo" name="archivo"><br><br>

    <!-- Input type="hidden" -->
    <input type="hidden" name="usuario_id" value="12345">

    <!-- Input type="image" -->
    <input type="image" src="https://fakeimg.pl/250x100/" alt="Enviar Formulario" style="width: 150px; height: 150px;"><br><br>

    <!-- Input type="month" -->
    <label for="mes">Selecciona un mes:</label>
    <input type="month" id="mes" name="mes"><br><br>

    <!-- Input type="number" -->
    <label for="numero">Número:</label>
    <input type="number" id="numero" name="numero" min="1" max="10"><br><br>

    <!-- Input type="password" -->
    <label for="password">Contraseña:</label>
    <input type="password" id="password" name="password" required><br><br>

    <!-- Input type="radio" -->
    <label>Selecciona tu género:</label><br>
```

```

<label>
    <input type="radio" name="genero" value="masculino"> Masculino
</label><br>
<label>
    <input type="radio" name="genero" value="femenino"> Femenino
</label><br><br>

<!-- Input type="range" -->
<label for="rango">Rango de 0 a 100:</label>
<input type="range" id="rango" name="rango" min="0" max="100" value="50"><br>
<br>

<!-- Input type="reset" -->
<input type="reset" value="Reiniciar"><br><br>

<!-- Input type="search" -->
<label for="busqueda">Buscar:</label>
<input type="search" id="busqueda" name="busqueda" placeholder="Buscar... ">
<br><br>

<!-- Input type="submit" -->
<input type="submit" value="Enviar"><br><br>

<!-- Input type="tel" -->
<label for="telefono">Teléfono:</label>
<input type="tel" id="telefono" name="telefono" placeholder="555-1234"><br>
<br>

<!-- Input type="text" -->
<label for="nombre">Nombre:</label>
<input type="text" id="nombre" name="nombre" required><br><br>

<!-- Input type="time" -->
<label for="hora">Selecciona una hora:</label>
<input type="time" id="hora" name="hora"><br><br>

<!-- Input type="url" -->
<label for="url">URL:</label>
<input type="url" id="url" name="url" placeholder="https://example.com"><br>
<br>

<!-- Input type="week" -->
<label for="semana">Selecciona una semana:</label>
<input type="week" id="semana" name="semana"><br><br>

</form>

</body>
</html>

```

CSS

Selectores CSS

Los **selectores** definen sobre qué elementos se aplicará un conjunto de reglas CSS.

Selectores básicos

- [Selector de tipo](#)

Selecciona todos los elementos que coinciden con el nombre del elemento especificado. **Sintaxis:** `eltname` **Ejemplo:** `input` se aplicará a cualquier elemento `input`.

- [Selector de clase](#)

Selecciona todos los elementos que tienen el atributo de `class` especificado. **Sintaxis:** `.classname` **Ejemplo:** `.index` seleccionará cualquier elemento que tenga la clase "index".

- [Selector de ID](#)

Selecciona un elemento basándose en el valor de su atributo `id`. Solo puede haber un elemento con un determinado ID dentro de un documento. **Sintaxis:** `#idname` **Ejemplo:** `#toc` se aplicará a cualquier elemento que tenga el ID "toc".

- [Selector universal](#)

Selecciona todos los elementos. Opcionalmente, puede estar restringido a un espacio de nombre específico o a todos los espacios de nombres. **Sintaxis:** `* ns | * * | *` **Ejemplo:** `*` se aplicará a todos los elementos del documento.

- [Selector de atributo](#)

Selecciona elementos basándose en el valor de un determinado atributo. **Sintaxis:** `[attr] [attr=value] [attr~=value] [attr|=value] [attr^=value] [attr$=value] [attr*=value]` **Ejemplo:** `[autoplay]` seleccionará todos los elementos que tengan el atributo "autoplay" establecido (a cualquier valor).

Combinadores

- [Combinador de hermanos adyacentes](#)

El combinador `+` selecciona hermanos adyacentes. Esto quiere decir que el segundo elemento sigue directamente al primero y ambos comparten el mismo elemento padre. **Sintaxis:** `A + B` **Ejemplo:** La regla `h2 + p` se aplicará a todos los elementos [\[1\]](https://developer.mozilla.org/es/docs/Web/HTML/Element/p_que_siguen_directamente_a_un_elemento) [\[2\]](https://developer.mozilla.org/es/docs/Web/HTML/Element/p_que_siguen_directamente_a_un_elemento).

- [Combinador general de hermanos](#)

El combinador `~` selecciona hermanos. Esto quiere decir que el segundo elemento sigue al primero (no necesariamente de forma inmediata) y ambos comparten el mismo elemento padre. **Sintaxis:** `A ~ B` **Ejemplo:** La regla `p ~ span` se aplicará a todos los elementos [\[1\]](https://developer.mozilla.org/es/docs/Web/HTML/Element/span_que_siguen_un_elemento) [\[2\]](https://developer.mozilla.org/es/docs/Web/HTML/Element/span_que_siguen_un_elemento).

- [Combinador de hijo](#)

El combinador `>` selecciona los elementos que son hijos directos del primer elemento. **Sintaxis:** `A > B` **Ejemplo:** La regla `ul > li` se aplicará a todos los elementos `li` (<https://developer.mozilla.org/es/docs/Web/HTML/Element/li>) que son hijos directos de un elemento `ul`.

- [Combinador de descendientes](#)

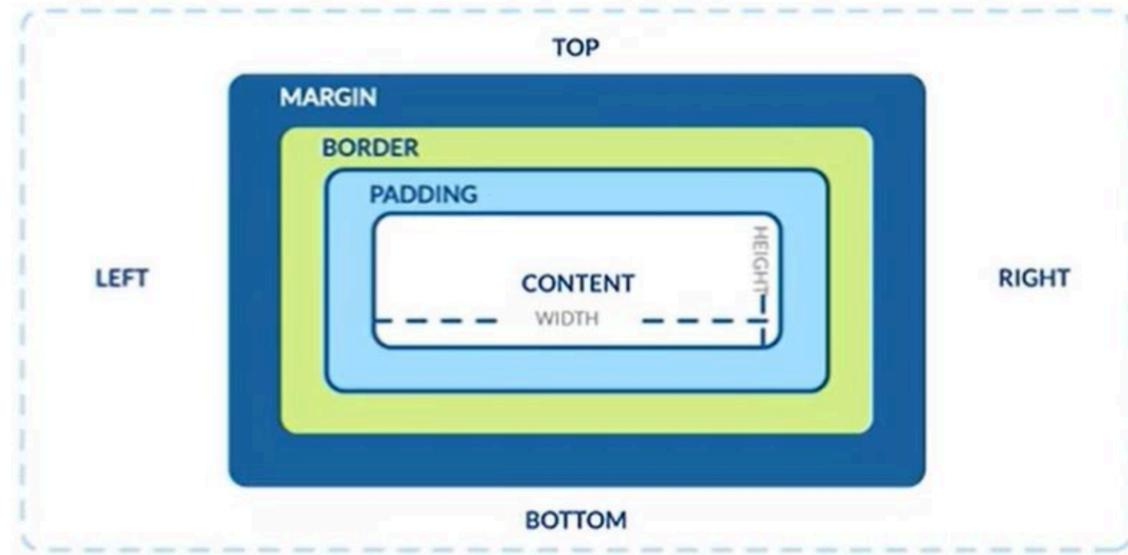
El combinador `(espacio)` selecciona los elementos que son descendientes del primer elemento. **Sintaxis:** `A B` **Ejemplo:** La regla `div span` se aplicará a todos los elementos `span` (<https://developer.mozilla.org/es/docs/Web/HTML/Element/span>) que están dentro de un elemento `div`.

- [Combinador de columna Experimental](#)

El combinador `||` selecciona los elementos especificados pertenecientes a una columna. **Sintaxis:** `A || B` **Ejemplo:** `col || td` seleccionará todos los elementos `]` (<https://developer.mozilla.org/es/docs/Web/HTML/Element/td>) que pertenezcan al ámbito de .

Box-sizing, display, visibilidad y posicionamiento

El modelo de caja (*box model*) es un concepto fundamental en CSS que describe cómo se calculan y organizan los elementos en una página web. Cada elemento en HTML se representa como una caja, y el modelo de caja establece cómo se determina el tamaño de estos elementos, incluyendo el contenido, los bordes, el relleno (*padding*) y los márgenes (*margins*).



Propiedad `box-sizing`

La propiedad `box-sizing` permite controlar cómo se calculan el ancho y el alto de los elementos, influyendo en si el relleno y el borde se incluyen en el cálculo de las dimensiones.

- **content-box (valor por defecto):** Las dimensiones (`width` y `height`) se aplican solo al contenido, sin incluir el borde ni el relleno. Es decir, si se define `width: 200px` y el elemento tiene un borde y relleno, el tamaño total del elemento será mayor.

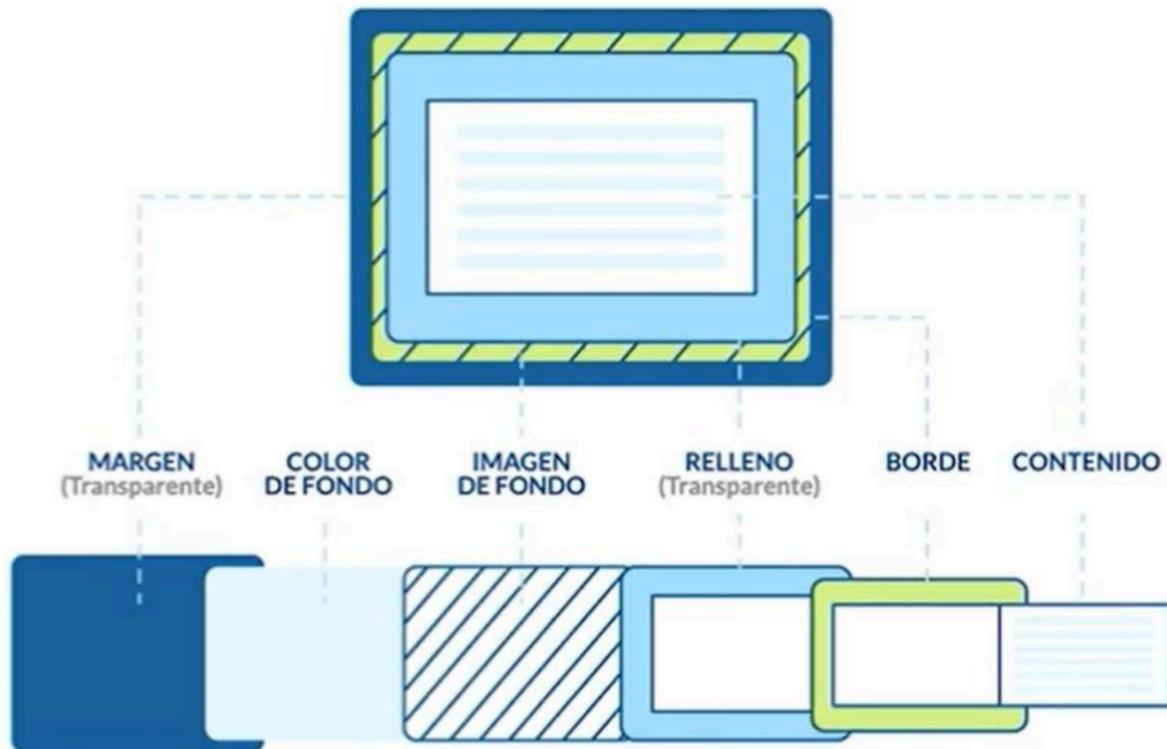
```
.mi-caja {  
    box-sizing: content-box;  
    width: 200px;  
    padding: 20px;  
    border: 10px solid black;  
}
```

El **tamaño total** será: 200px (contenido) + 20px (relleno) + 10px (borde) = 250px.

- **border-box**: El relleno y el borde se incluyen en las dimensiones (`width` y `height`), por lo que si se define `width: 200px`, el tamaño total del elemento será de 200px, incluyendo el relleno y el borde.

```
.mi-caja {  
    box-sizing: border-box;  
    width: 200px;  
    padding: 20px;  
    border: 10px solid black;  
}
```

El **tamaño total** será exactamente 200px, ya que el borde y el relleno están incluidos dentro de ese valor.



Contenido (Content)

- Es el área donde se muestra el contenido real del elemento, como texto o imágenes. El tamaño de este área puede ajustarse utilizando propiedades como `width` y `height`.

Relleno (Padding)

- El relleno es el espacio entre el contenido y el borde del elemento. Puedes usar la propiedad `padding` para añadir espacio dentro del borde, pero fuera del contenido.

Borde (Border)

- El borde rodea tanto el contenido como el relleno. Puedes personalizar su grosor, estilo y color con la propiedad `border`.

Margen (Margin)

- El margen es el espacio exterior que separa el elemento de otros elementos a su alrededor. Puedes usar la propiedad `margin` para agregar espacio entre el borde del elemento y los elementos cercanos.

Ejemplo

```
background-color: #purple;
color: white;
height: 200px;
width: 200px;
margin-left: auto;
margin-right: auto;
box-sizing: border-box;
border: 5px solid red;

padding: 20px;
```



Outline

En CSS, `outline` es una propiedad que se utiliza para dibujar un contorno alrededor de un elemento sin afectar su tamaño o flujo en la página. Es similar a `border`, pero con algunas diferencias clave:

Diferencias entre `outline` y `border`

- 1. No afecta el tamaño del elemento:** `outline` se dibuja fuera del borde del elemento sin modificar su ancho o altura.
- 2. No se pueden definir lados individuales:** A diferencia de `border`, `outline` se aplica alrededor del elemento como un solo contorno, no se pueden especificar lados individuales como `outline-left` o `outline-top`.
- 3. No influye en el modelo de caja:** No afecta el `box-sizing`, ni ocupa espacio en el diseño.

Propiedades de outline

1. outline-width

Define el grosor del contorno. Puede tener valores como:

- `thin` (delgado)
- `medium` (medio)
- `thick` (grueso)
- Un valor específico en `px`, `em`, `rem`, etc.

2. outline-style

Define el estilo del contorno. Puede ser:

- `solid` (línea continua)
- `dotted` (puntos)
- `dashed` (líneas discontinuas)
- `double` (doble línea)
- `groove`, `ridge`, `inset`, `outset` (efectos en relieve)
- `none` (sin contorno)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Box Model en CSS</title>
    <link href="css/estilos.css" rel="stylesheet"/>
</head>
<body>
    <h1>Box Model en CSS - Aprende Diseño Web</h1>
    <div>My Content</div>
</body>
</html>
```

```
h1{
    text-align: center;
    color: #e63946;
}
div{
    padding: 10px;
    border: 15px solid #457b9d;
    margin: 20px;
    background-color: #f1faee;
    width: 300px;
    outline: 10px outset #a8dadc;
}
```

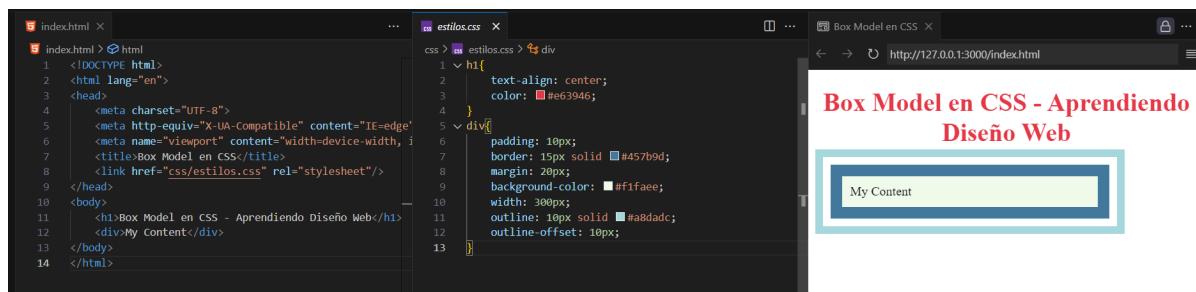
Outline-offset

La propiedad `outline-offset` en CSS se utiliza para establecer la distancia entre el borde (`outline`) de un elemento y su borde exterior. A diferencia de `border`, el `outline` no ocupa espacio en el modelo de caja, por lo que no afecta el diseño de la página.

Sintaxis

```
outline-offset: <longitud> | inherit;
```

- `<longitud>`: Especifica la distancia entre el `outline` y el borde del elemento. Puede ser un valor positivo, negativo o cero.
- `inherit`: Hereda el valor de la propiedad `outline-offset` del elemento padre.



Padding

La propiedad `padding` en CSS se utiliza para definir el espacio entre el contenido de un elemento y su borde. A diferencia de `margin`, que establece el espacio fuera del borde, `padding` afecta el espacio interno del elemento.

Sintaxis

```
padding: <valor>;
```

- `<valor>`: Puede ser una longitud (como `px`, `em`, `%`), un valor relativo (como `auto`), o una combinación de valores para cada lado del elemento.

Formas de usar padding:

1. Un solo valor:

Aplica el mismo `padding` a los cuatro lados (arriba, derecha, abajo, izquierda).

```
div {  
    padding: 20px;  
}
```

2. Dos valores:

- El primer valor aplica al `padding` superior e inferior.
- El segundo valor aplica al `padding` izquierdo y derecho.

```
div {  
    padding: 10px 20px;  
}
```

3. Tres valores:

- El primer valor aplica al `padding` superior.
- El segundo valor aplica al `padding` izquierdo y derecho.
- El tercer valor aplica al `padding` inferior.

```
div {  
    padding: 10px 20px 15px;  
}
```

4. Cuatro valores:

- Aplica `padding` en el orden: superior, derecha, inferior, izquierda (sentido de las agujas del reloj).

```
div {  
    padding: 10px 20px 15px 5px;  
}
```

Propiedades individuales:

También puedes definir el `padding` para cada lado por separado:

- `padding-top`: Espacio superior.
- `padding-right`: Espacio derecho.
- `padding-bottom`: Espacio inferior.
- `padding-left`: Espacio izquierdo.

Ejemplo

```
div {  
    padding-top: 10px;  
    padding-right: 20px;  
    padding-bottom: 15px;  
    padding-left: 5px;  
}
```

Consideraciones

- El `padding` aumenta el tamaño total del elemento si se usa con `box-sizing: content-box` (valor por defecto).
- Si se usa `box-sizing: border-box`, el `padding` se incluye dentro del tamaño total del elemento.
- El `padding` no tiene valores negativos.

Compatibilidad:

- La propiedad `padding` es compatible con todos los navegadores modernos y antiguos.

```

index.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Box Model en CSS</title>
8   <link href="css/estilos.css" rel="stylesheet"/>
9 </head>
10 <body>
11   <h1>Box Model en CSS</h1>
12   <div>Mi contenido</div>
13 </body>
14 </html>

```

```

estilos.css
1 h1{
2   text-align: center;
3   color: #e63946;
4 }
5 div{
6   padding: 10px 15px 5px 3px;
7   border: 0px solid #457b9d;
8   margin: 20px;
9   background-color: #fifaae;
10  width: 300px;
11  outline: 10px solid #a8adac;
12  outline-offset: 10px;
13 }
14

```

Box Model en CSS

Mi contenido

box-sizing

La propiedad `box-sizing` en CSS define cómo se calcula el tamaño total de un elemento, específicamente cómo se tienen en cuenta el `padding` y el `border` en relación con el ancho (`width`) y alto (`height`) del elemento.

Valores de `box-sizing`:

1. `content-box` (valor por defecto):

- El `width` y `height` solo incluyen el contenido.
- El `padding` y el `border` se añaden fuera del tamaño especificado.
- Fórmula:

$$\text{Ancho total} = \text{width} + \text{padding} + \text{border}$$

$$\text{Alto total} = \text{height} + \text{padding} + \text{border}$$

2. `border-box`:

- El `width` y `height` incluyen el contenido, el `padding` y el `border`.
- El tamaño total del elemento es exactamente el valor que especificas en `width` y `height`.
- Fórmula:

$$\text{Ancho total} = \text{width} \text{ (incluye padding y border)}$$

$$\text{Alto total} = \text{height} \text{ (incluye padding y border)}$$

Sintaxis

```
box-sizing: content-box | border-box;
```

Ejemplos:

1. `content-box` (comportamiento por defecto)

```

div {
  width: 200px;
  padding: 20px;
  border: 10px solid black;
  box-sizing: content-box;
}

```

- Ancho total = `200px` (contenido) + `20px` (padding izquierdo) + `20px` (padding derecho)
+ `10px` (borde izquierdo) + `10px` (borde derecho) = **260px**.

2. `border-box`:

```
div {
    width: 200px;
    padding: 20px;
    border: 10px solid black;
    box-sizing: border-box;
}
```

- Ancho total = `200px` (incluye el contenido, el padding y el borde).
- El contenido interno tendrá un ancho de `200px - 20px (padding) - 20px (padding)`
`- 10px (borde) - 10px (borde) = 140px`.

¿Por qué es útil `box-sizing`?

- **Facilita el diseño:** Con `border-box`, se puede establecer un tamaño fijo para un elemento sin preocuparse por que el `padding` o el `border` lo desborden.
- **Responsive design:** Es más fácil calcular tamaños y márgenes cuando el `padding` y el `border` están incluidos en el tamaño total del elemento.

max-width

La propiedad `max-width` en CSS se utiliza para establecer el ancho máximo que puede tener un elemento. Si el contenido del elemento es más grande que el valor de `max-width`, el elemento se ajustará para no superar ese ancho máximo. Si el contenido es más pequeño, `max-width` no tendrá efecto.

Sintaxis

```
max-width: <valor> | none | initial | inherit;
```

- `<valor>`: Puede ser una longitud (como `px`, `em`, `%`), un valor relativo (como `vw`), o `auto`.
- `none`: No se establece un ancho máximo (valor por defecto).
- `initial`: Restablece el valor a su predeterminado (`none`).
- `inherit`: Hereda el valor de la propiedad `max-width` del elemento padre.

Valores comunes:

1. Longitud fija:

```
div {
    max-width: 500px;
}
```

2. Porcentaje:

```
div {
    max-width: 80%;
}
```

3. Unidad relativa (vw):

```
div {
    max-width: 50vw;
}
```

The screenshot shows a code editor with two files: `index.html` and `estilos.css`. The `index.html` file contains basic HTML structure including a `<meta name="viewport" content="width=device-width, initial-scale=1.0" />`. The `estilos.css` file contains CSS rules for `h1` and `div` elements. A red arrow points to the `max-width: 100%;` rule in the `div` selector. The browser preview on the right shows a green box with the text "Mi contenido".

```
index.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7     <title>Box Model en CSS</title>
8     <link href="css/estilos.css" rel="stylesheet"/>
9 </head>
10 <body>
11     <h1>Box Model en CSS</h1>
12     <div>Mi contenido</div>
13 </body>
14 </html>
```

```
estilos.css
1 h1{
2     text-align: center;
3     color: #e63946;
4 }
5 div{
6     padding: 10px 15px 5px 3px;
7     border: 1px solid #457b9d;
8     margin: 0px;
9     background-color: #fifafee;
10    max-width: 100%; // Red arrow points here
11    box-sizing: border-box;
12 }
13 }
```

Unidades Absolutas

Las unidades absolutas son fijas y no cambian en función del contexto o del tamaño de otros elementos. Siempre tienen el mismo tamaño, independientemente del dispositivo o la pantalla.

Ejemplos de unidades absolutas:

1. px (píxeles):

- Unidad más común.
- 1px equivale a un punto en la pantalla.
- Ejemplo: `font-size: 16px;`.

2. in (pulgadas):

- 1in = 2.54cm.
- Ejemplo: `width: 2in;`.

3. cm (centímetros):

- Basado en el sistema métrico.
- Ejemplo: `height: 5cm;`.

4. mm (milímetros):

- 1cm = 10mm.
- Ejemplo: `padding: 10mm;`.

5. pt (puntos):

- Usado comúnmente en impresión.
- 1pt = 1/72 de una pulgada.
- Ejemplo: `font-size: 12pt;`.

6. pc (picas):

- 1pc = 12 puntos.
- Ejemplo: `line-height: 1.5pc;`.

Cuándo usarlas:

- Cuando necesitas un tamaño fijo que no cambie en ningún contexto.
 - Para diseños impresos, donde las medidas físicas (como `cm` o `in`) son importantes.
-

Unidades Relativas

Las unidades relativas dependen de otro valor de referencia, como el tamaño de la fuente, el tamaño de la ventana del navegador o el tamaño del contenedor padre. Esto las hace más flexibles y adecuadas para diseños responsivos.

Ejemplos de unidades relativas:

1. `em`:
 - Relativo al tamaño de la fuente del elemento.
 - Si el `font-size` del elemento es 16px, `1em = 16px`.
 - Ejemplo: `padding: 2em;` (si `font-size` es 16px, el padding será 32px).
2. `rem (root em)`:
 - Relativo al tamaño de la fuente del elemento raíz (`<html>`).
 - Si el `font-size` del elemento raíz es 16px, `1rem = 16px`.
 - Ejemplo: `font-size: 1.5rem;` (si el `font-size` raíz es 16px, el tamaño será 24px).
3. `% (porcentaje)`:
 - Relativo al tamaño del contenedor padre.
 - Ejemplo: `width: 50%;` (el elemento tendrá la mitad del ancho de su contenedor).
4. `vw (viewport width)`:
 - Relativo al ancho de la ventana del navegador.
 - $1vw = 1\%$ del ancho de la ventana.
 - Ejemplo: `width: 50vw;` (el elemento tendrá el 50% del ancho de la ventana).
5. `vh (viewport height)`:
 - Relativo al alto de la ventana del navegador.
 - $1vh = 1\%$ del alto de la ventana.
 - Ejemplo: `height: 100vh;` (el elemento tendrá el 100% del alto de la ventana).
6. `vmin y vmax`:
 - `vmin`: Relativo al valor más pequeño entre el ancho y el alto de la ventana.
 - `vmax`: Relativo al valor más grande entre el ancho y el alto de la ventana.
 - Ejemplo: `font-size: 5vmin;` (el tamaño de la fuente será el 5% del lado más pequeño de la ventana).

Cuándo usarlas:

- Para diseños responsivos que se adapten a diferentes tamaños de pantalla.
- Cuando quieres que los elementos escalen en función del tamaño de la fuente o del contenedor.

Diferencias clave:

Característica	Unidades Absolutas	Unidades Relativas
Tamaño	Fijo	Depende de un valor de referencia
Flexibilidad	Poca	Alta
Uso común	Impresión, tamaños fijos	Diseños responsivos
Ejemplos	<code>px, in, cm, pt</code>	<code>em, rem, %, vw, vh</code>

inherit

En CSS, la palabra clave `inherit` se utiliza para hacer que un elemento herede el valor de una propiedad específica de su elemento padre. Esto es útil cuando quieres que un elemento tenga el mismo valor que su contenedor padre para una propiedad determinada, en lugar de usar un valor predeterminado o uno explícito.

Sintaxis

```
propiedad: inherit;
```

- `propiedad`: La propiedad CSS de la cual deseas heredar el valor (por ejemplo, `color`, `font-size`, `border`, etc.).
- `inherit`: Indica que el valor debe tomarse del elemento padre.

¿Cómo funciona?

Cuando usas `inherit`, el elemento toma el valor calculado de la propiedad del elemento padre. Esto es especialmente útil cuando:

1. Se quiere que un elemento tenga el mismo estilo que su padre.
2. Necesita anular un valor predeterminado o un valor específico asignado previamente.

Casos de uso comunes:

1. Componentes reutilizables:

Si tienes un componente que debe adaptarse al estilo de su contenedor, puedes usar `inherit` para asegurarte de que herede propiedades como `color`, `font-size`, o `background-color`.

2. Anular estilos predeterminados:

Algunos elementos tienen estilos predeterminados (por ejemplo, los enlaces `<a>` suelen ser azules). Puede usar `inherit` para que adopten el estilo de su contenedor.

3. Diseños flexibles:

En diseños responsivos, puedes usar `inherit` para que los elementos secundarios se ajusten automáticamente a los cambios en el contenedor padre.

Diferencias con otros valores:

- `initial`: Establece la propiedad a su valor predeterminado (no hereda del parente).
- `unset`: Combina `initial` e `inherit`. Si la propiedad es heredable, se comporta como `inherit`; de lo contrario, se comporta como `initial`.
- `revert`: Restablece la propiedad al valor definido por el user-agent (navegador) o la hoja de estilo del usuario.

Propiedades heredables comunes

Estas son algunas de las propiedades CSS que **sí son heredables**:

1. Texto:

- `color`
- `font-family`
- `font-size`
- `font-weight`
- `font-style`
- `line-height`
- `text-align`
- `text-transform`
- `letter-spacing`
- `word-spacing`

2. Listas:

- `list-style-type`
- `list-style-position`
- `list-style-image`

3. Tablas:

- `border-collapse`
- `border-spacing`
- `caption-side`

4. Otros:

- `visibility`
 - `cursor`
 - `direction`
 - `white-space`
 - `quotes`
-

Propiedades no heredables

Estas son algunas de las propiedades CSS que **no son heredables**:

1. Modelo de caja:

- `width`
- `height`
- `margin`
- `padding`
- `border`
- `box-sizing`

2. Posicionamiento:

- `position`
- `top`
- `right`
- `bottom`
- `left`
- `z-index`

3. Fondo:

- `background-color`
- `background-image`
- `background-position`
- `background-size`

4. Otros:

- `display`
- `float`
- `clear`
- `overflow`

Margin Collapse

Margin Collapse (colapso de márgenes) es un comportamiento específico de CSS que ocurre cuando los márgenes verticales (superior e inferior) de dos o más elementos adyacentes se superponen, combinándose en un solo margen en lugar de sumarse. Este comportamiento solo aplica a márgenes verticales, no a márgenes horizontales.

¿Cuándo ocurre el margin collapse?

El colapso de márgenes ocurre en tres situaciones principales:

1. Márgenes adyacentes de elementos hermanos:

- Cuando dos elementos hermanos (uno después del otro en el flujo del documento) tienen márgenes verticales, estos márgenes se colapsan.

```
<div style="margin-bottom: 20px;">Elemento 1</div>
<div style="margin-top: 30px;">Elemento 2</div>
```

- El espacio entre los dos elementos no será `50px` (`20px + 30px`), sino `30px` (el margen más grande).

2. Márgenes de un parent y su primer/último hijo:

- Si un elemento parent no tiene borde, padding o contenido que lo separe de su primer hijo, el margen superior del hijo colapsa con el margen superior del parent.
- Lo mismo ocurre con el margen inferior del último hijo y el margen inferior del parent.
- Ejemplo:

```
<div style="margin-top: 20px;">
  <div style="margin-top: 30px;">Hijo</div>
</div>
```

3. Márgenes de elementos vacíos:

- Si un elemento no tiene contenido, borde, padding o altura, sus márgenes superior e inferior colapsan.
- Ejemplo:

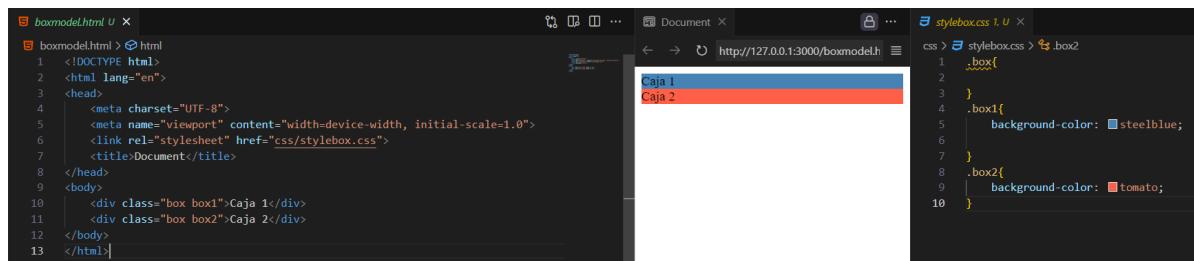
```
<div style="margin-top: 20px; margin-bottom: 30px;"></div>
```

¿Cómo se calcula el margen colapsado?

- Cuando dos márgenes colapsan, el tamaño del margen resultante es el **mayor** de los dos márgenes.
- Si uno de los márgenes es negativo, se resta del margen positivo.
- Si ambos márgenes son negativos, se usa el valor más negativo.

Taller

1



2. Quitando márgenes por defecto

```

boxmodel.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="css/stylebox.css">
7   <title>Document</title>
8 </head>
9 <body>
10  <div class="box box1">Caja 1</div>
11  <div class="box box2">Caja 2</div>
12 </body>
13 </html>

```

```

stylebox.css
1 body{
2   margin: 0;
3 }
4 .box{
5   background-color: #steelblue;
6   padding: 10px;
7 }
8 .box1{
9   height: 100px;
10  width: 200px;
11 }
12 .box2{
13   height: 100px;
14  width: 200px;
15 }

```

3. Agregando padding y border

```

boxmodel.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="css/stylebox.css">
7   <title>Document</title>
8 </head>
9 <body>
10  <div class="box box1">Caja 1</div>
11  <div class="box box2">Caja 2</div>
12 </body>
13 </html>

```

```

stylebox.css
1 body{
2   margin: 0;
3 }
4 .box{
5   border: 1px solid #black;
6   padding: 10px;
7 }
8 .box1{
9   background-color: #steelblue;
10 }
11 .box2{
12   background-color: #tomato;
13 }

```

4. Agregando margenes

```

boxmodel.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="css/stylebox.css">
7   <title>Document</title>
8 </head>
9 <body>
10  <div class="box box1">Caja 1</div>
11  <div class="box box2">Caja 2</div>
12 </body>
13 </html>

```

```

stylebox.css
1 body{
2   margin: 0;
3 }
4 .box{
5   border: 1px solid #black;
6   padding: 10px;
7   margin: 10px;
8 }
9 .box1{
10  background-color: #steelblue;
11 }
12 .box2{
13  background-color: #tomato;
14 }

```

5. Aplicando Margenes

```

boxmodel.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="css/stylebox.css">
7   <title>Document</title>
8 </head>
9 <body>
10  <div class="box box1"></div>
11  <div class="box box2"></div>
12 </body>
13 </html>

```

```

stylebox.css
1 body{
2   margin: 0;
3 }
4 .box{
5   width: 200px;
6   height: 100px;
7 }
8 .box1{
9   background-color: #steelblue;
10  margin-top: 20px;
11  margin-bottom: 20px;
12  margin-left: 10px;
13  margin-right: 15px;
14 }
15 .box2{
16  background-color: #tomato;
17 }

```

```

boxmodel.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="css/stylebox.css">
7   <title>Document</title>
8 </head>
9 <body>
10  <div class="box box1"></div>
11  <div class="box box2"></div>
12 </body>
13 </html>

```

```

stylebox.css
1 body{
2   margin: 0;
3 }
4 .box{
5   width: 200px;
6   height: 100px;
7 }
8 .box1{
9   background-color: #steelblue;
10  margin-top: 20px;
11  margin-bottom: 20px;
12  margin-left: 10px;
13  margin-right: 15px;
14 }
15 .box2{
16  background-color: #tomato;
17 }

```

6. Centrar elemento Margin:auto (Debe ser elemento bloque y tener ancho definido)

boxmodel.html

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.>
6   <link rel="stylesheet" href="css/stylebox.css">
7   <title>Document</title>
8 </head>
9 <body>
10  <div class="box box1"></div>
11  <div class="box box2"></div>
12 </body>
13 </html>

```

stylebox.css

```

1 body{
2   margin: 0;
3 }
4 .box{
5   width: 200px;
6   height: 100px;
7 }
8 .box1{
9   background-color: steelblue;
10  margin: auto;
11 }
12 .box2{
13   background-color: tomato;
14 }
15

```

Recomendación centrar elementos con margin auto.

boxmodel.html

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.>
6   <link rel="stylesheet" href="css/stylebox.css">
7   <title>Document</title>
8 </head>
9 <body>
10  <div class="box box1"></div>
11  <div class="box box2"></div>
12 </body>
13 </html>

```

stylebox.css

```

1 body{
2   margin: 0;
3 }
4 .box{
5   width: 200px;
6   height: 100px;
7 }
8 .box1{
9   background-color: steelblue;
10  margin-left: auto;
11  margin-right: auto;
12 }
13 .box2{
14   background-color: tomato;
15 }
16

```

Usando modo corto

boxmodel.html

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.>
6   <link rel="stylesheet" href="css/stylebox.css">
7   <title>Document</title>
8 </head>
9 <body>
10  <div class="box box1"></div>
11  <div class="box box2"></div>
12 </body>
13 </html>

```

stylebox.css

```

1 body{
2   margin: 0;
3 }
4 .box{
5   width: 200px;
6   height: 100px;
7 }
8 .box1{
9   background-color: steelblue;
10  margin: 0 auto;
11 }
12 .box2{
13   background-color: tomato;
14 }
15

```

Nombres de Colores HTML Roja

COLOR	NAME	HEX CODE	RGB CODE
	IndianRed	#CD5C5C	rgb(205, 92, 92)
	LightCoral	#F08080	rgb(240, 128, 128)
	Salmon	#FA8072	rgb(250, 128, 114)
	DarkSalmon	#E9967A	rgb(233, 150, 122)
	LightSalmon	#FFA07A	rgb(255, 160, 122)
	Crimson	#DC143C	rgb(220, 20, 60)
	Red	#FF0000	rgb(255, 0, 0)
	FireBrick	#B22222	rgb(178, 34, 34)
	DarkRed	#8B0000	rgb(139, 0, 0)

Nombres de Colores HTML Rosa

COLOR	NAME	HEX CODE	RGB CODE
	Pink	#FFC0CB	rgb(255, 192, 203)
	LightPink	#FFB6C1	rgb(255, 182, 193)
	HotPink	#FF69B4	rgb(255, 105, 180)
	DeepPink	#FF1493	rgb(255, 20, 147)
	MediumVioletRed	#C71585	rgb(199, 21, 133)
	PaleVioletRed	#DB7093	rgb(219, 112, 147)

Nombres de Colores HTML Naranja

COLOR	NAME	HEX CODE	RGB CODE
	LightSalmon	#FFA07A	rgb(255, 160, 122)
	Coral	#FF7F50	rgb(255, 127, 80)
	Tomato	#FF6347	rgb(255, 99, 71)
	OrangeRed	#FF4500	rgb(255, 69, 0)
	DarkOrange	#FF8C00	rgb(255, 140, 0)
	Orange	#FFA500	rgb(255, 165, 0)

Nombres de Colores HTML Amarilla

COLOR	NAME	HEX CODE	RGB CODE
	Gold	#FFD700	rgb(255, 215, 0)
	Yellow	#FFFF00	rgb(255, 255, 0)
	LightYellow	#FFFFE0	rgb(255, 255, 224)
	LemonChiffon	#FFFACD	rgb(255, 250, 205)
	LightGoldenrodYellow	#FAFAD2	rgb(250, 250, 210)
	PapayaWhip	#FFEFD5	rgb(255, 239, 213)
	Moccasin	#FFE4B5	rgb(255, 228, 181)
	PeachPuff	#FFDAB9	rgb(255, 218, 185)
	PaleGoldenrod	#EEE8AA	rgb(238, 232, 170)
	Khaki	#F0E68C	rgb(240, 230, 140)

Nombres de Colores HTML Púrpura

COLOR	NAME	HEX CODE	RGB CODE
	Lavender	#E6E6FA	rgb(230, 230, 250)
	Thistle	#D8BFD8	rgb(216, 191, 216)
	Plum	#DDA0DD	rgb(221, 160, 221)
	Violet	#EE82EE	rgb(238, 130, 238)
	Orchid	#DA70D6	rgb(218, 112, 214)
	Fuchsia	#FF00FF	rgb(255, 0, 255)
	Magenta	#FF00FF	rgb(255, 0, 255)
	MediumOrchid	#BA55D3	rgb(186, 85, 211)
	MediumPurple	#9370DB	rgb(147, 112, 219)
	RebeccaPurple	#663399	rgb(102, 51, 153)

Nombres de Colores HTML Marrón

COLOR	NAME	HEX CODE	RGB CODE
	Cornsilk	#FFF8DC	rgb(255, 248, 220)
	BlanchedAlmond	#FFEBCD	rgb(255, 235, 205)
	Bisque	#FFE4C4	rgb(255, 228, 196)
	NavajoWhite	#FFDEAD	rgb(255, 222, 173)
	Wheat	#F5DEB3	rgb(245, 222, 179)
	BurlyWood	#DEB887	rgb(222, 184, 135)
	Tan	#D2B48C	rgb(210, 180, 140)
	RosyBrown	#BC8F8F	rgb(188, 143, 143)
	SandyBrown	#F4A460	rgb(244, 164, 96)
	Goldenrod	#DAA520	rgb(218, 165, 32)

<https://htmlcolorcodes.com/es/nombres-de-los-colores/>

Overflow

La propiedad `overflow` en CSS se usa para controlar el comportamiento del contenido cuando este es más grande que el contenedor que lo contiene. Define cómo se muestra y maneja el contenido desbordado.

Valores de `overflow`

Valor	Descripción
<code>visible</code>	El contenido desbordado se muestra sin recorte. (Valor por defecto).
<code>hidden</code>	El contenido desbordado se oculta sin agregar barra de desplazamiento.
<code>scroll</code>	Siempre agrega barras de desplazamiento (horizontal y vertical), incluso si no es necesario.
<code>auto</code>	Agrega barras de desplazamiento solo si el contenido se desborda.
<code>clip</code>	Similar a <code>hidden</code> , pero sin permitir el desplazamiento con teclado o scroll.
<code>overlay</code>	Similar a <code>auto</code> , pero en algunos navegadores oculta las barras hasta que se necesiten. (No ampliamente soportado).

Box Model en CSS

Mi contenido Lorem ipsum dolor sit amet consectetur adipisicing elit. Eaque libero corporis minus? Officia ipsa laboriosam ut, aspernatur labore et voluptatum nihil incident ut, aspernatur labore et voluptatum est ea commodi.

overflow: hidden;

Box Model en CSS

Mi contenido Lorem ipsum dolor sit amet consectetur adipisicing elit. Eaque libero corporis minus? Officia ipsa laboriosam ut, aspernatur labore et voluptatum nihil incident ut, aspernatur labore et voluptatum est ea commodi.

overflow: scroll;

Box Model en CSS

Mi contenido Lorem ipsum dolor sit amet consectetur adipisicing elit. Eaque libero corporis minus? Officia ipsa laboriosam ut, aspernatur labore et voluptatum nihil incident ut, aspernatur labore et voluptatum est ea commodi.

overflow: auto;

Box Model en CSS

Mi contenido Lorem ipsum dolor sit amet consectetur adipisicing elit. Eaque libero corporis minus? Officia ipsa laboriosam ut, aspernatur labore et voluptatum nihil incident ut, aspernatur labore et voluptatum est ea commodi.

text-overflow

La propiedad `text-overflow` en CSS se usa para manejar el desbordamiento del texto cuando no cabe en su contenedor y se corta de diferentes maneras.

Valores de `text-overflow`

Valor	Descripción
<code>clip</code>	El texto se corta sin agregar ningún indicador visual. (Valor por defecto).
<code>ellipsis</code>	Muestra "..." cuando el texto no cabe en su contenedor.
<code>string</code>	Se puede usar una cadena personalizada para indicar el recorte (poco soportado en navegadores).

Requisitos para que `text-overflow` funcione

Para que esta propiedad tenga efecto, se deben cumplir estas condiciones:

1. El contenedor debe tener un ancho definido (`width` o `max-width`).
2. Se debe establecer `white-space: nowrap;` para evitar que el texto se divida en múltiples líneas.
3. Se debe aplicar `overflow: hidden;` para ocultar el texto desbordado.

The screenshot shows a code editor with three tabs: 'index.html', 'estilos.css', and a browser preview window titled 'Box Model en CSS'.

index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Box Model en CSS</title>
<link href="css/estilos.css" rel="stylesheet"/>
</head>
<body>
<h1>Box Model en CSS</h1>
<div>
    Mi contenido
    Lorem ipsum dolor sit amet consectetur adipisicing elit. Eaque libero ...
</div>
</body>
</html>
```

estilos.css:

```
h1{
    text-align: center;
    color: #e63946;
}
div{
    box-sizing: border-box;
    width: 200px;
    background-color: #f1faee;
    padding: 5px;
    white-space: nowrap;
    overflow: hidden; /* Oculta el texto que se desborda */
    text-overflow: ellipsis; /* Muestra los tres puntos (...) */
}
div:hover {
    white-space: normal; /* Permite que el texto se divida en varias líneas */
    overflow: visible; /* Muestra todo el contenido */
    max-height: 200px; /* Se expande automáticamente */
}
```

Browser Preview:

Box Model en CSS

Mi contenido Lorem ipsum ...

The browser shows a centered h1 'Box Model en CSS'. Below it is a div containing the text 'Mi contenido' followed by a long string of placeholder text. The div has a fixed width of 200px, a yellow background, and a black border. The text is nowrap and hidden, with an ellipsis at the end. When the mouse hovers over the div, the width increases to 200px, the text wraps onto multiple lines, and the border expands to fit the new height.

Tipografía en CSS

color

Define el color del texto.

Propiedad `font-family` en CSS

La propiedad `font-family` en CSS se usa para definir la fuente del texto de un elemento. Se puede especificar una lista de fuentes de respaldo en caso de que la primera no esté disponible en el sistema del usuario.

```
elemento {
    font-family: "Nombre de la fuente", fuente-alternativa, tipo-genérico;
}
```

```
p {  
    font-family: "Arial", "Helvetica", sans-serif;  
}
```

Tipos de fuentes genéricas en CSS

Las fuentes genéricas son un tipo de respaldo que indica el estilo general de la fuente:

Tipo genérico	Descripción
<code>serif</code>	Fuentes con remates o adornos (Ej: Times New Roman).
<code>sans-serif</code>	Fuentes sin remates, más limpias (Ej: Arial, Helvetica).
<code>monospace</code>	Fuentes de ancho fijo, donde cada carácter ocupa el mismo espacio (Ej: Courier New).
<code>cursive</code>	Fuentes con apariencia manuscrita (Ej: Comic Sans MS).
<code>fantasy</code>	Fuentes decorativas y poco convencionales.

Propiedad `font-size` en CSS

La propiedad `font-size` en CSS se usa para definir el tamaño del texto en un elemento. Se pueden utilizar valores en diferentes unidades, como píxeles (`px`), em (`em`), porcentajes (`%`), y más.

Sintaxis

```
elemento {  
    font-size: tamaño;  
}
```

Unidades de medida en `font-size`

Unidad	Descripción	Ejemplo
<code>px</code>	Tamaño en píxeles. Valor fijo.	<code>font-size: 16px;</code>
<code>em</code>	Relativo al tamaño de la fuente del elemento padre.	<code>font-size: 1.5em;</code> (1.5 veces el tamaño del padre)
<code>rem</code>	Relativo al tamaño de la fuente del <code>html</code> (raíz).	<code>font-size: 2rem;</code> (2 veces el tamaño base del <code>html</code>)
<code>%</code>	Relativo al tamaño del contenedor padre.	<code>font-size: 120%;</code> (120% del tamaño heredado)
<code>vw</code>	Relativo al ancho de la ventana del navegador.	<code>font-size: 5vw;</code> (5% del ancho de la pantalla)

Unidad	Descripción	Ejemplo
vh	Relativo a la altura de la ventana del navegador.	font-size: 5vh; (5% de la altura de la pantalla)

```

html {
    font-size: 16px;
}

h1 {
    font-size: 2rem; /* 32px (16px * 2) */
}

p {
    font-size: 1.2em; /* 1.2 veces el tamaño del parent */
}

```

❖ Diferencia clave:

- `em` se basa en el tamaño del **elemento padre**.



1. El contenedor padre tiene un tamaño de fuente de `18px`.
2. Cada elemento hijo usa `1.2em` como `font-size`, lo que significa que cada elemento hereda el tamaño de su parent y lo multiplica por `1.2`.

Cálculo de los tamaños de fuente:

1. Primer nivel (fondo verde):

- `1.2em` sobre `18px` → $1.2 \times 18\text{px} = 21.6\text{px}$

2. Segundo nivel (fondo rojo):

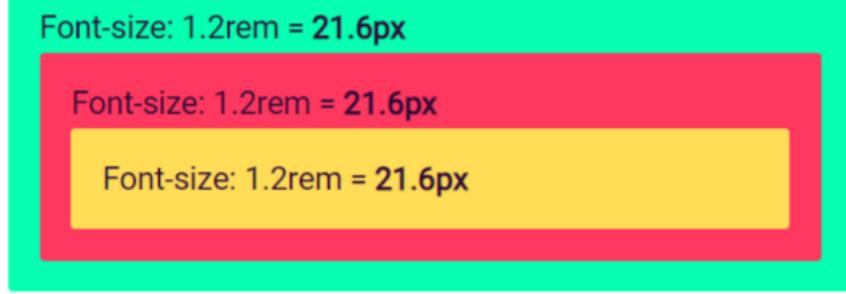
- `1.2em` sobre `21.6px` → $1.2 \times 21.6\text{px} = 25.92\text{px}$

3. Tercer nivel (fondo amarillo):

- `1.2em` sobre `25.92px` → $1.2 \times 25.92\text{px} = 31.104\text{px}$

- `rem` se basa en el tamaño de la **raíz** (`html`).

<html> root element: 18px



Detalles de la imagen:

1. El tamaño de fuente del <html> es 18px .
2. Todos los elementos usan 1.2rem como font-size .
3. Cada elemento mantiene el mismo tamaño de fuente (21.6px) en todos los niveles.

Cálculo del tamaño de fuente:

- 1.2rem se calcula como $1.2 \times 18\text{px} = 21.6\text{px}$ en todos los niveles.
- A diferencia de em, donde el tamaño crece exponencialmente si los elementos están anidados, aquí el tamaño se mantiene constante.

Conclusión:

- rem es más predecible y evita el crecimiento acumulativo de em.
- Es ideal para un diseño consistente, especialmente en layouts responsivos.

font-size:smaller;

Parent container: 18px



Se pueden usar **palabras clave relativas**. El valor es relativo al **elemento padre**.

Las siguientes opciones están disponibles:

- larger (más grande)
- smaller (más pequeño)

La propiedad font-size en CSS permite usar **palabras clave relativas** (larger y smaller) en lugar de valores fijos (px , em , rem , etc.).

❖ ¿Cómo funciona?

- larger : Aumenta el tamaño de la fuente en comparación con el parent.
- smaller : Reduce el tamaño de la fuente en comparación con el parent.

```

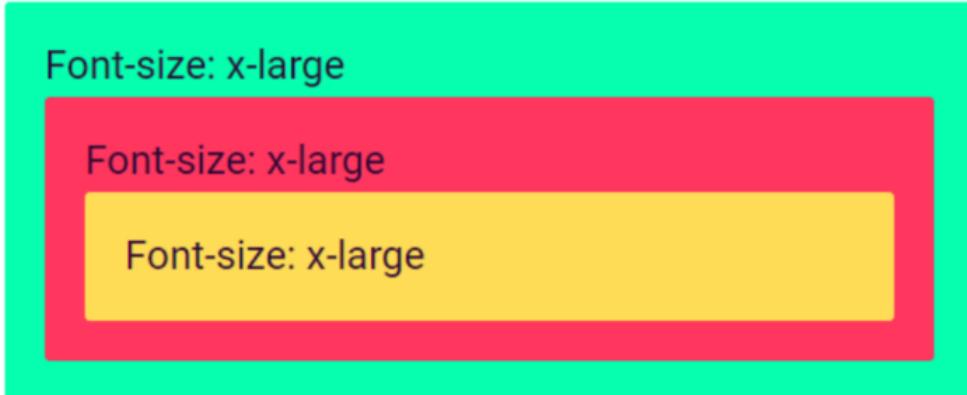
p {
  font-size: larger; /* Más grande que el texto del elemento padre */
}

span {
  font-size: smaller; /* Más pequeño que el texto del elemento padre */
}

```

Tambien se **Pueden usar palabras clave absolutas**. El valor es relativo al **elemento raíz** `<html>`.

`<html> root element: 18px`



Las siguientes opciones están disponibles:

Lista de palabras clave absolutas disponibles en `font-size`

Palabra clave	Tamaño relativo
<code>xx-small</code>	Muy pequeño
<code>x-small</code>	Más pequeño
<code>small</code>	Pequeño
<code>medium</code>	Normal (por defecto en los navegadores, usualmente <code>16px</code>)
<code>large</code>	Grande
<code>x-large</code>	Más grande
<code>xx-large</code>	Muy grande

- Las **palabras clave absolutas** permiten establecer tamaños predefinidos sin necesidad de usar `px`, `em` o `rem`.
- Son útiles cuando quieres mantener **una escala de tamaño de fuente sencilla y consistente**.
- Sin embargo, **no son tan flexibles como `em` o `rem`**, ya que dependen de cómo el navegador los interprete.

Conclusión

- Usa **px** para tamaños fijos.
- Usa **em/rem** para escalabilidad.
- Usa **vw/vh** para tamaños adaptables en diseños responsivos.

Propiedad `font-style` en CSS

La propiedad `font-style` en CSS se usa para definir el estilo de la fuente del texto, permitiendo cambiar entre normal, itálico o oblicuo.

Valores de `font-style`

Valor	Descripción
<code>normal</code>	Texto con estilo de fuente normal (sin cursiva).
<code>italic</code>	Aplica un estilo cursivo (si la fuente tiene versión en itálica).
<code>oblique</code>	Aplica un estilo inclinando artificialmente el texto.

Diferencia entre `italic` y `oblique`

- `italic` usa una versión **real** en cursiva de la fuente si está disponible.
- `oblique` inclina artificialmente el texto si no existe una versión cursiva en la fuente.

Propiedad `font-weight` en CSS

La propiedad `font-weight` en CSS se usa para **definir el grosor de la fuente** del texto. Permite desde fuentes delgadas hasta negritas o extra gruesas.

📌 Lista de valores disponibles:

Valor	Descripción
100	Thin (Muy delgado)
200	Extra Light (Extra ligero)
300	Light (Ligero)
400	Normal (Predeterminado)
500	Medium (Medio)
600	Semi Bold (Semi negrita)
700	Bold (Negrita)
800	Extra Bold (Extra negrita)
900	Ultra Bold (Ultra negrita)

Ejemplo

```

@import url('https://fonts.googleapis.com/css2?
family=Roboto:wght@100;300;500;700;900&display=swap');

body {
    font-family: 'Roboto', sans-serif;
}

h1 {
    font-weight: 900; /* Extra grueso */
}

p {
    font-weight: 300; /* Más delgado */
}

```

Manejo de fondos

The screenshot shows a code editor with two files: `index.html` and `estilos.css`. The `index.html` file contains basic HTML structure including a title and a paragraph. The `estilos.css` file contains CSS rules for the body element, setting the background color to `dodgerblue`, the color to `cornsilk`, and the background image to `/img/fondo.png` with no repeat. The browser preview on the right shows the resulting page with a blue gradient background and white text.

`background-color: dodgerblue;`

- **Propósito:** Establece el color de fondo de un elemento.
- **Valor:** `dodgerblue` es un color predefinido en CSS, que es un tono de azul claro.

`color: cornsilk;`

- **Propósito:** Define el color del texto dentro de un elemento.
- **Valor:** `cornsilk` es un color predefinido en CSS, que es un tono de blanco amarillento.

`background-image: url("/img/fondo.png");`

- **Propósito:** Establece una imagen como fondo de un elemento.
- **Valor:** `url("/img/fondo.png")` especifica la ruta de la imagen que se usará como fondo. En este caso, la imagen está en la carpeta `/img` y se llama `fondo.png`.

`background-repeat: no-repeat;`

- **Propósito:** Controla si la imagen de fondo se repite o no.
- **Valor:** `no-repeat` indica que la imagen de fondo no se repetirá. Solo se mostrará una vez.
- **Otras opciones:**
 - `repeat`: La imagen se repite tanto horizontal como verticalmente (valor por defecto).
 - `repeat-x`: La imagen se repite solo horizontalmente.
 - `repeat-y`: La imagen se repite solo verticalmente.

```

index.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Manejo de Fondos en CSS</title>
8   <link href="/css/estilos.css" rel="stylesheet"/>
9 </head>
10 <body>
11   <h1>Manejo de Fondos en CSS</h1>
12   <p>Texto de la página web</p>
13 </body>

```

```

estilos.css
1 body{
2   color: #cornsilk;
3   /*
4   background-color: dodgerblue;
5   background-image: url("img/fondo.png");
6   background-repeat: no-repeat;
7   background-position: right top;
8   */
9   background: #dodgerblue url("/img/fondo.png") no-repeat;
10  background-size: cover; /* La imagen cubre toda el área */
11  background-position: center; /* Centra la imagen */
12  background-repeat: no-repeat; /* Evita que la imagen se repita */
13  height: 100vh; /* Ocupa el 100% del alto de la pantalla */
14  margin: 0; /* Elimina el margen predeterminado del body */
15  background-attachment: fixed;
16 }
17 h1{
18   text-align: center;
19   background-color: #dodgerblue;
20 }

```

1. `background: dodgerblue url("/img/fondo.png") no-repeat;`

- **Propiedad:** `background`

- **Valor:** `dodgerblue url("/img/fondo.png") no-repeat`

- `dodgerblue`: Color de fondo (azul claro) que se muestra si la imagen no se carga.
- `url("/img/fondo.png")`: Ruta de la imagen de fondo.
- `no-repeat`: Evita que la imagen de fondo se repita.

2. `background-size: cover;`

- **Propiedad:** `background-size`

- **Valor:** `cover`

- Escala la imagen para que cubra todo el espacio del elemento, manteniendo su proporción. Si la imagen es más pequeña que el elemento, se estirará; si es más grande, se recortará.

3. `background-position: center;`

- **Propiedad:** `background-position`

- **Valor:** `center`

- Centra la imagen de fondo tanto horizontal como verticalmente dentro del elemento.

4. `background-repeat: no-repeat;`

- **Propiedad:** `background-repeat`

- **Valor:** `no-repeat`

- Evita que la imagen de fondo se repita. Solo se mostrará una vez.

5. `height: 100vh;`

- **Propiedad:** `height`

- **Valor:** `100vh`

- Establece la altura del elemento al 100% del alto de la ventana del navegador (viewport height).

6. `margin: 0;`

- **Propiedad:** margin

- **Valor:** 0

- Elimina el margen predeterminado del elemento (en este caso, el body).

7. background-attachment: fixed;

- **Propiedad:** background-attachment

- **Valor:** fixed

- Fija la imagen de fondo en su posición, de modo que no se desplace al hacer scroll.

Resumen:

- **background**: Combina color de fondo, imagen y repetición en una sola propiedad.

- **background-size: cover**: Asegura que la imagen cubra todo el espacio.

- **auto** (valor por defecto):
 - **Comportamiento:** La imagen de fondo se muestra en su tamaño original.

- **cover**:
 - **Comportamiento:** La imagen se escala para cubrir todo el espacio del elemento, manteniendo su proporción. Si la imagen es más pequeña que el elemento, se estirará; si es más grande, se recortará.

- **contain**:
 - **Comportamiento:** La imagen se escala para que quepa completamente dentro del elemento, manteniendo su proporción. Esto puede dejar espacios vacíos si la imagen no coincide con las dimensiones del elemento.

- **Valores explícitos (longitud o porcentaje)**:
 - **Comportamiento:** Puedes especificar el ancho y el alto de la imagen de fondo usando valores como **px**, **em**, **%**, etc.

Sintaxis

```
background-size: <ancho> <alto>;
```

```
div {  
    background-image: url("/img/fondo.png");  
    background-size: 200px 100px; /* Ancho: 200px, Alto: 100px */  
}
```

◦ Ejemplos prácticos:

1. Cubrir todo el espacio (cover):

```
body {  
    background-image: url("/img/fondo.png");  
    background-size: cover;  
}
```

2. Ajustar la imagen dentro del contenedor (contain):

```
div {  
    background-image: url("/img/fondo.png");  
    background-size: contain;  
}
```

La imagen se ajustará para que quepa completamente dentro del `div`, sin recortarse.

- `background-position: center`: Centra la imagen.
- `background-repeat: no-repeat`: Evita la repetición de la imagen.
- `height: 100vh`: Hace que el elemento ocupe toda la altura de la pantalla.
- `margin: 0`: Elimina márgenes predeterminados.
- `background-attachment: fixed`: Fija la imagen para que no se mueva al hacer scroll.
 - `scroll` (valor por defecto):
 - **Comportamiento:** La imagen de fondo se desplaza junto con el contenido de la página.
 - `fixed`:
 - **Comportamiento:** La imagen de fondo permanece fija en su posición, incluso cuando se desplaza la página.
 - `local`:
 - **Comportamiento:** La imagen de fondo se desplaza junto con el contenido del elemento, pero si el elemento tiene un scroll interno, la imagen se moverá con él.

```
div {  
    background-image: url("/img/fondo.png");  
    background-attachment: local;  
    overflow: auto; /* Permite el scroll interno */  
    height: 200px;  
}
```

Formato de texto

The screenshot shows a code editor interface with three tabs: 'index.html', 'estilos.css', and a browser preview window titled 'Formato de Texto en CSS'.

The 'index.html' tab contains the following HTML code:

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta http-equiv="X-UA-Compatible" content="IE-edge">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Formato de Texto con CSS</title>  
    <link href="/css/estilos.css" rel="stylesheet"/>  
</head>  
<body>  
    <h1>Formato de Texto en CSS</h1>  
    <div>  
        <p>Excepteur velit laboris laboris ullamco adipisicing  
        irure minima aliqua. Nostrud exercitation veniam dolor  
        consectetur cupidatat elit officia nostrud quis in  
        nostrud. Officia pariatur aute proident aute ad non  
        consectetur magna sit. Consectetur lorem aliqua duis  
        anim dolor Lorem irure est in aliquip enim aute.  
    </p>  
    </div>  
</body>  
</html>
```

The 'estilos.css' tab contains the following CSS code:

```
h1 {  
    color: #ef476f;  
    background-color: #ebe;  
    text-align: center;  
    text-transform: uppercase;  
    text-shadow: 2px 2px 2px #ced4da;  
}
```

The browser preview window shows the heading 'FORMATO DE TEXTO EN CSS' in a large, bold, uppercase font with a purple shadow effect. Below the heading, there is a block of placeholder text in a smaller, black font.

`text-align: center;`

- **Propósito:** Alinea el texto horizontalmente dentro de un elemento.
- **Valores admitidos:**
 - `left`: Alinea el texto a la izquierda (valor por defecto).
 - `right`: Alinea el texto a la derecha.
 - `center`: Centra el texto.
 - `justify`: Justifica el texto, ajustando el espacio entre palabras para que todas las líneas tengan el mismo ancho.
 - `start`: Alinea el texto al inicio del contenedor (depende de la dirección del texto, `ltr` o `rtl`).
 - `end`: Alinea el texto al final del contenedor (depende de la dirección del texto, `ltr` o `rtl`).

`text-transform: uppercase;`

- **Propósito:** Transforma el texto a mayúsculas, minúsculas o capitalizado.
- **Valores admitidos:**
 - `none`: No aplica ninguna transformación (valor por defecto).
 - `uppercase`: Convierte todo el texto a mayúsculas.
 - `lowercase`: Convierte todo el texto a minúsculas.
 - `capitalize`: Convierte la primera letra de cada palabra a mayúscula.
 - `full-width`: Convierte los caracteres a su versión de ancho completo (útil para idiomas como el japonés o el chino).

`text-shadow: 2px 2px 2px #ced4da;`

- **Propósito:** Añade una sombra al texto.

```
text-shadow: <desplazamiento-x> <desplazamiento-y> <desenfoque> <color>;
```

Valores admitidos:

- `<desplazamiento-x>`: Distancia horizontal de la sombra (puede ser positiva o negativa).
- `<desplazamiento-y>`: Distancia vertical de la sombra (puede ser positiva o negativa).
- `<desenfoque>`: Radio de desenfoque de la sombra (opcional). Un valor mayor crea una sombra más difusa.
- `<color>`: Color de la sombra (puede ser un nombre, código hexadecimal, RGB, etc.).

Valores adicionales y detalles:

1. `text-align`:

- `start` y `end`: Útiles para idiomas con direcciones de texto diferentes (por ejemplo, árabe o hebreo).

- `justify`: Puede crear espacios desiguales entre palabras, especialmente en líneas cortas.

2. `text-transform`:

- `capitalize`: Solo afecta a la primera letra de cada palabra, no a las letras siguientes.
- `full-width`: Convierte caracteres a su versión de ancho completo, útil para alinear texto en idiomas asiáticos.

3. `text-shadow`:

- Puedes usar colores transparentes (por ejemplo, `rgba(206, 212, 218, 0.5)`) para sombras más sutiles.
- Las sombras múltiples se apilan en el orden en que se escriben (la primera sombra aparece encima).

```

index.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Formato de Texto con CSS</title>
8   <link href="/css/estilos.css" rel="stylesheet"/>
9 </head>
10 <body>
11   <h1>Formato de Texto en CSS</h1>
12   <div class="caja">
13     <p class="texto">
14       <span>Curso de Diseño web.</span>
15       <span>Molit dolore dolor aute dolor ex excepteur occaecat enim fugiat irure cupidatat irure aliquip. Id veniam proident culpa ex qui culpa sint et incididunt aliquip adipisicing. Occaecat officia in tempor amet sit lorem deserunt mollit eu reprehenderit dolor proident. Sunt veniam consectetur eu labore aute veniam fugiat duis consequat est cupidatat nisi ut aliquip ex ea commodo consequat sit nulla aliquip. Ex nostrud Lorem eiusmod qui aute excepteur qui ipsum voluptate Lorem et esse pariatur incididunt. Ex duis labore ea adipisicing consequat sit nulla aliquip. Dolore ullamco do eu velit labore excepteur eu qui quis aute.
16     </p>
17   </div>
18 </body>
19 </html>

```

```

estilos.css
1 h1{
2   color: #ef476f;
3   text-align: center;
4   text-transform: uppercase;
5   text-shadow: 2px 2px 2px #ced4da;
6   letter-spacing: 3px;
7   word-spacing: 10px;
8   text-decoration: overline;
9 }
10 .caja{
11   padding: 15px;
12   margin: auto;
13   width: 70%;
14   background-color: #219ebc;
15   border-radius: 5px;
16   color:white;
17 }
18 .texto{
19   text-align: justify;
20   /*direction: rtl;*/
21 }

```

FORMATO DE TEXTO EN CSS

Curso de Diseño web.

Molit dolore dolor aute dolor ex excepteur occaecat enim fugiat irure cupidatat irure aliquip. Id veniam proident culpa ex qui culpa sint et incididunt aliquip adipisicing. Occaecat officia in tempor amet sit lorem deserunt mollit eu reprehenderit dolor proident. Sunt veniam consectetur eu labore aute veniam fugiat duis consequat est cupidatat nisi ut aliquip ex ea commodo consequat sit nulla aliquip. Ex nostrud Lorem eiusmod qui aute excepteur qui ipsum voluptate Lorem et esse pariatur incididunt. Ex duis labore ea adipisicing consequat sit nulla aliquip. Dolore ullamco do eu velit labore excepteur eu qui quis aute.

```

h1{
  color: #ef476f;
  text-align: center;
  text-transform: uppercase;
  text-shadow: 2px 2px 2px #ced4da;
  letter-spacing: 3px;
  word-spacing: 10px;
  text-decoration: overline;
}
.caixa{
  padding: 15px;
  margin: auto;
  width: 70%;
  background-color: #219ebc;
  border-radius: 5px;
  color:white;
}
.texto{
  text-align: justify;
  /*direction: rtl;*/
}

```

`letter-spacing: 3px;`

- **Propósito:** Controla el espacio entre los caracteres (letras) de un texto.
- **Valores admitidos:**
 - `normal`: Espaciado predeterminado (valor por defecto).
 - `<longitud>`: Un valor en unidades como `px`, `em`, `rem`, etc. Puede ser positivo (aumenta el espacio) o negativo (reduce el espacio).
 - `inherit`: Hereda el valor de la propiedad `letter-spacing` del elemento padre.

`word-spacing: 10px;`

- **Propósito:** Controla el espacio entre las palabras de un texto.
- **Valores admitidos:**
 - `normal`: Espaciado predeterminado (valor por defecto).
 - `<longitud>`: Un valor en unidades como `px`, `em`, `rem`, etc. Puede ser positivo (aumenta el espacio) o negativo (reduce el espacio).
 - `inherit`: Hereda el valor de la propiedad `word-spacing` del elemento padre.
- **Efecto:**
 - Un valor positivo aumenta el espacio entre palabras.
 - Un valor negativo reduce el espacio entre palabras (puede hacer que las palabras se junten).

`text-decoration: overline;`

- **Propósito:** Añade una decoración al texto, como subrayado, tachado o línea superior.
- **Valores admitidos:**
 - `none`: Sin decoración (valor por defecto).
 - `underline`: Subraya el texto.
 - `overline`: Añade una línea encima del texto.
 - `line-through`: Tacha el texto con una línea horizontal.
 - `blink`: Hace que el texto parpadee (no es compatible con todos los navegadores y no se recomienda su uso).
 - `inherit`: Hereda el valor de la propiedad `text-decoration` del elemento padre.
- **Efecto:**
 - Puedes combinar varios valores, por ejemplo: `text-decoration: underline overline;` para subrayar y poner una línea encima del texto.

Conclusión

- `letter-spacing`: Controla el espacio entre caracteres.
- `word-spacing`: Controla el espacio entre palabras.
- `text-decoration`: Añade decoraciones al texto, como líneas superiores, inferiores o tachados.

text-indent: Value:

```
index.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Formato de Texto con CSS</title>
8   <link href="/css/estilos.css" rel="stylesheet">
9 </head>
10 <body>
11   <h1>Formato de Texto en CSS</h1>
12   <div class="caja">
13     <p class="texto">
14       Entrenamiento en diseño web CSS Lorem ipsum dolor sit, amet consectetur adipisicing elit. Corrupti, amet quae. Magni accusamus cumque quos animi illum eligendi ad explicabo. Quisquam dolores optio excepturi minimal iste animi possimus nobis dolorum!
15     </p>
16     <p class="textito">
17       Mollit dolore dolor autem dolor ex excepteur occaecat enim fugiat occaecat irure cupidatat irure aliquip. Id veniam praident enim culpa ex qui culpa sint et incididunt aliquip adipiscinc. Occaecat officia in tempor amet sit Lorem deserunt mollit eu reprehenderit dolor praident. Sunt veniam consectetur du labore eu reprehenderit dolor duis consequunt ut incididunt consetetur non reprehenderit sit. Ex nostrud Lorem eiusmod qui autem excepteur qui ipsum voluptate Lorem et esse pariatur incididunt. Ex duis labore ea adipiscinc consequent sit nulla aliquip. Dolore ullamco do eu velit labore excepteur eu qui quis autem.
18   </div>
19 </body>
20 </html>
```

```
estilos.css
1 h1{
2   color: #f4f4f6;
3   text-align: center;
4   text-decoration: overline;
5   text-decoration-color: black;
6   text-decoration-style: none;
7   text-decoration-thickness: 2px;
8   letter-spacing: 3px;
9   word-spacing: 3px;
10  text-decoration: underline;
11 }
12 .caja{
13   padding: 15px;
14   margin: auto;
15   width: 70px;
16   background-color: #219ebc;
17   border-radius: 5px;
18   color: white;
19 }
20 .textito{
21   text-align: justify;
22   /*direction: rtl;*/
23   text-indent: 30px;
24   line-height: 1.5;
25   white-space: normal;
26   text-decoration: 1px 1px 2px #ced4da;
27 }
```

FORMATO DE TEXTO EN CSS

Entrenamiento en diseño web CSS Lorem ipsum dolor sit, amet consectetur adipisicing elit. Corrupti, amet quae. Magni accusamus cumque quos animi illum eligendi ad explicabo. Quisquam dolores optio excepturi minimal! Iste animi possimus nobis dolorum!

Mollit dolore autem dolor ex excepteur occaecat enim fugiat occaecat irure cupidatat irure aliquip. Id veniam praident culpa ex qui culpa sint et incididunt aliquip adipiscinc. Occaecat officia in tempor amet sit Lorem deserunt mollit eu reprehenderit dolor praident. Sunt veniam consectetur du labore eu reprehenderit dolor duis consequent ut incididunt consetetur non reprehenderit sit. Ex nostrud Lorem eiusmod qui autem excepteur qui ipsum voluptate Lorem et esse pariatur incididunt. Ex duis labore ea adipiscinc consequent sit nulla aliquip. Dolore ullamco do eu velit labore excepteur eu qui quis autem.

En CSS, `text-indent` es una propiedad que se utiliza para especificar la sangría de la primera línea de un bloque de texto. Se puede definir en diferentes unidades como píxeles (`px`), ems (`em`), porcentajes (%), etc.

```
p {
  text-indent: 20px;
}
```

Valores comunes

- Longitud fija:** `text-indent: 50px;` → Sangría de 50 píxeles.
- Relativa al tamaño de fuente:** `text-indent: 2em;` → Sangría de 2 veces el tamaño de la fuente.
- Porcentaje:** `text-indent: 10%;` → Sangría del 10% del ancho del contenedor.
- Negativo:** `text-indent: -20px;` → La primera línea se mueve hacia la izquierda.
- Heredado:** `text-indent: inherit;` → Toma el valor de su elemento padre.

Especificidad

La **especificidad en CSS** es una medida que determina qué reglas de estilo se aplicarán a un elemento cuando hay múltiples reglas que coinciden con ese elemento. El navegador utiliza esta especificidad para decidir cuál de las reglas tendrá prioridad.

La especificidad se calcula sumando diferentes valores según el tipo de selector que estés utilizando. A continuación, se detallan los tipos de selectores y cómo se les asignan puntos de especificidad:

Reglas de especificidad

Selectores de estilo en línea (atributo `style` directamente en el HTML):

- Especificidad = 1000 puntos.

Selectores de ID:

- Especificidad = 100 puntos.

Selectores de clase, pseudoclases (`:hover`, `:nth-child`) y atributos:

- Especificidad = 10 puntos.

Selectores de elementos y pseudoelementos (`::before`, `::after`):

- Especificidad = 1 punto.

Selectores universales (`*`), combinadores (`+, >, ~`) y pseudoclases negadas (`:not()`):

- No afectan la especificidad.

Reglas adicionales:

- **Importancia (`!important`)**: Una regla marcada con `!important` sobrescribe otras reglas, independientemente de la especificidad.

Resumen de cómo calcular la especificidad:

- Estilos en línea: 1000
- ID: 100
- Clase, pseudoclase, atributo: 10
- Elemento, pseudoelemento: 1

Ejercicio

The screenshot shows a browser window with the URL `http://127.0.0.1:3000`. On the left, the HTML code for `especificidad.html` is visible, containing a navigation bar with four links, each with a different class: "enlaces", "enlaces", "enlaces block", and "enlaces". On the right, the rendered output shows four blue links, each labeled "Enlace".

The screenshot shows a browser window with the URL `http://127.0.0.1:3000`. On the left, the CSS file `estilos.css` is shown with a single selector for the universal selector (*). On the right, the rendered output shows four blue links, each labeled "Enlace".

Propiedad display

La propiedad **display** de CSS permite modificar el comportamiento de un elemento HTML, cambiándolo al que le indiquemos, como por ejemplo **inline** o **block**

Tipo de caja	Características
<code>block</code>	Se apila en vertical. Ocupa todo el ancho disponible de su etiqueta contenedora.
Versión en línea	
<code>inline</code>	Se coloca en horizontal. Se adapta al ancho de su contenido. Ignora <code>width</code> o <code>height</code> .
<code>inline-block</code>	Combinación de los dos anteriores. Se comporta como <code>inline</code> pero no ignora <code>width</code> o <code>height</code> .
Versión flexible	
<code>flex</code>	Utiliza el modelo de cajas flexibles de CSS. Ideal para estructuras de 1 dimensión.
<code>inline-flex</code>	Versión en línea (ocupa sólo su contenido) del modelo de cajas flexibles de CSS.
Versión de cuadrículas	
<code>grid</code>	Utiliza cuadrículas o rejillas con el modelo de cajas Grid CSS.
<code>inline-grid</code>	La versión en línea (ocupa sólo su contenido) del modelo de cajas grid css.
Versión de listas y tablas	
Versión de listas y tablas	
<code>list-item</code>	Actúa como un ítem de una lista. Es el comportamiento de etiquetas como <code></code> .
<code>table</code>	Actúa como una tabla. Es el comportamiento de etiquetas como <code><table></code> .
<code>table-cell</code>	Actúa como la celda de una tabla. Es el comportamiento de etiquetas como <code><th></code> o <code><td></code> .
<code>table-row</code>	Actúa como la fila de una tabla. Es el comportamiento de etiquetas como <code><tr></code> .
Otros	
<code>contents</code>	Ignora la caja del elemento. Útil para mantener Grid/Flex aún teniendo un wrapper intermedio.
<code>none</code>	Oculta el elemento visualmente, como si no existiera en el HTML.

The screenshots demonstrate the application of CSS rules across different elements and their descendants. In the first screenshot, four links are styled by a general class. In the second, an h2 element changes the styling of the entire nav. In the third, a more specific selector (the link inside the h2) overrides the general class.

```

    Box Model HTML (Left):
    <!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="css/estilos.css">
        <title>Document</title>
    </head>
    <body>
        <nav>
            <a href="#" class="enlaces">Enlace</a>
            <a href="#" class="enlaces">Enlace</a>
            <a href="#" class="enlaces" block>Enlace</a>
            <a href="#" class="enlaces">Enlace</a>
        </nav>
    </body>
    </html>
  
```

```

    Document (Middle):
    http://127.0.0.1:3000/especificidad.html
    Enlace Enlace Enlace Enlace
  
```

```

    estilos.css (Right):
    .enlaces {
        margin: 0;
        padding: 0;
        box-sizing: border-box;
    }
    .enlaces {
        text-decoration: none;
        background-color: #crimson;
        color: #white;
        padding: 30px 20px;
    }
  
```



```

    Box Model HTML (Left):
    <!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="css/estilos.css">
        <title>Document</title>
    </head>
    <body>
        <nav>
            <h2>Heyy</h2>
            <a href="#" class="enlaces">Enlace</a>
            <a href="#" class="enlaces">Enlace</a>
            <a href="#" class="enlaces" block>Enlace</a>
            <a href="#" class="enlaces">Enlace</a>
        </nav>
    </body>
    </html>
  
```

```

    Document (Middle):
    Heyy
    Enlace Enlace Enlace Enlace
  
```

```

    estilos.css (Right):
    .enlaces {
        margin: 0;
        padding: 0;
        box-sizing: border-box;
    }
    .enlaces {
        display: inline-block;
        text-decoration: none;
        background-color: #crimson;
        color: #white;
        padding: 30px 20px;
    }
  
```



```

    Box Model HTML (Left):
    <!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="css/estilos.css">
        <title>Document</title>
    </head>
    <body>
        <nav>
            <h2>Heyy</h2>
            <a href="#" class="enlaces">Enlace</a>
            <a href="#" class="enlaces">Enlace</a>
            <a href="#" class="enlaces" block>Enlace</a>
            <a href="#" class="enlaces">Enlace</a>
        </nav>
    </body>
    </html>
  
```

```

    Document (Middle):
    Heyy
    Enlace Enlace Enlace
  
```

```

    estilos.css (Right):
    .enlaces {
        margin: 0;
        padding: 0;
        box-sizing: border-box;
    }
    .enlaces {
        display: inline-block;
        text-decoration: none;
        background-color: #crimson;
        color: #white;
        padding: 30px 20px;
    }
    .block {
        display: none;
    }
  
```

Cascada y Especificidad

La **cascada** se refiere al proceso mediante el cual CSS aplica las reglas de estilo en función de su origen y orden de declaración. Este proceso sigue un conjunto de reglas para determinar qué estilo debe aplicarse, basado en los siguientes factores:

- **Origen del estilo:** Los estilos pueden venir de:
 - CSS del navegador (estilos predeterminados).
 - CSS del usuario (configuraciones que el usuario puede haber establecido en el navegador).
 - CSS del autor (los estilos que defines en tu hoja de estilo).

Skill diseño web

Lore ipsum dolor sit amet consectetur adipisicing elit. Molestiae laudantium libero ea esse, non repellat deleniti recusandae quis? Libero, id. Doloremque, omnis repellat. Tenetur illum, quaerat aliquid laboriosam atque distinctio?

```

css > estilos.css > .enlaces
1  *{
2    margin: 0;
3    padding: 0;
4    box-sizing: border-box;
5  }
6 .enlaces{
7   display: inline-block;
8   text-decoration: none;
9   background-color: #crimson;
10  color: white;
11  padding: 30px 20px;
12 }
13 .block{
14   display: none;
15 }

```

Ejemplo de especificidad

Skill diseño web

Lore ipsum dolor sit amet consectetur adipisicing elit. Molestiae laudantium libero ea esse, non repellat deleniti recusandae quis? Libero, id. Doloremque, omnis repellat. Tenetur illum, quaerat aliquid laboriosam atque distinctio?

```

css > estilos.css > ...
1  *{
2    margin: 0;
3    padding: 0;
4    box-sizing: border-box;
5  }
6 /* .enlaces{
7   display: inline-block;
8   text-decoration: none;
9   background-color: crimson;
10  color: white;
11  padding: 30px 20px;
12 }
13 .block{
14   display: none;
15 }

```

Skill diseño web

Lore ipsum dolor sit amet consectetur adipisicing elit. Molestiae laudantium libero ea esse, non repellat deleniti recusandae quis? Libero, id. Doloremque, omnis repellat. Tenetur illum, quaerat aliquid laboriosam atque distinctio?

```

css > estilos.css > h1
1  /* .enlaces{
2   display: inline-block;
3   text-decoration: none;
4   background-color: crimson;
5   color: white;
6   padding: 30px 20px;
7 }
8 .block{
9   display: none;
10 }
11 .header{
12   height: 200px;
13   background-color: antiquewhite;
14   color: blueviolet;
15 }

```

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport"
6     content="width=device-width,
7     initial-scale=1.0">
8   <link rel="stylesheet" href="css/estilos.css">
9   <title>Document</title>
10 </head>
11 <body>
12   <!-- <nav> ... -->
13   <header class="header">
14     <h2 class="title" id="title">Skill
15       diseño web</h2>
16     <p class="copy">Lorem ipsum dolor sit
17       amet consectetur adipisicing elit.
18       Molestiae laudantium libero ea esse,
19       non repellat deleniti recusandae quis?
20       Libero, id. Doloremque, omnis repellat.
21       Tenetur illum, quaerat aliquid
22       laboriosam atque distinctio?</p>
23   </header>
24 </body>
25 </html>

```

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport"
6     content="width=device-width,
7     initial-scale=1.0">
8   <link rel="stylesheet" href="css/estilos.css">
9   <title>Document</title>
10 </head>
11 <body>
12   <!-- <nav> ... -->
13   <header class="header">
14     <h2 class="title" id="title">Skill
15       diseño web</h2>
16     <p class="copy">Lorem ipsum dolor sit
17       amet consectetur adipisicing elit.
18       Molestiae laudantium libero ea esse,
19       non repellat deleniti recusandae quis?
20       Libero, id. Doloremque, omnis repellat.
21       Tenetur illum, quaerat aliquid
22       laboriosam atque distinctio?</p>
23   </header>
24 </body>
25 </html>

```

Herencia

La **herencia en CSS** es el mecanismo por el cual ciertos estilos aplicados a un elemento **padre** son transmitidos automáticamente a sus **hijos**. No todos los estilos se heredan, pero los que sí lo hacen permiten que puedas aplicar un estilo de forma general en un contenedor y que este se propague a los elementos dentro de él.

¿Qué propiedades se heredan?

No todas las propiedades CSS son heredables. Las propiedades que más comúnmente se heredan incluyen:

- Propiedades relacionadas con el texto:

- `color`
- `font-family`
- `font-size`
- `line-height`
- `text-align`
- `visibility`

- Propiedades relacionadas con el espacio:

- `letter-spacing`
- `word-spacing`

Propiedades que NO se heredan

Las propiedades relacionadas con el **modelo de caja** (box model) y el diseño de la página no se heredan automáticamente. Estas incluyen:

- margin
- padding
- border
- width, height
- display
- position

Para aplicar estas propiedades a los elementos hijos, tienes que hacerlo explícitamente en cada uno o en su selector correspondiente.

The screenshot illustrates the concept of inheritance in CSS. In the top window, the HTML code defines a header with a height of 200px and a dashed border. Inside this header, there is a title element with a padding of 30px and a background color of crimson. In the bottom window, the CSS file 'estilos.css' contains a selector for '.title' that sets both 'border: inherit;' and 'padding: inherit;'. This means that the title element will inherit the border and padding properties from its parent header. As a result, the title element has a pink background, a 3px dashed border, and 30px padding, matching the appearance of the header it is contained within.

especificidad.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport"
6     content="width=device-width,
7     initial-scale=1.0">
8   <link rel="stylesheet" href="css/estilos.css">
9   <title>Document</title>
10 </head>
11 <body>
12   <!-- <nav> ... -->
13   <header class="header">
14     <h2 class="title" id="title">Skill
15       diseño web</h2>
16     <p class="copy">Lorem ipsum dolor sit
17       amet consectetur adipisicing elit.
18       Molestiae laudantium libero ea esse,
19       non repellat deleniti recusandae quis?
20       Libero, id. Doloremque, omnis repellat.
21       Tenetur illum, quærat aliquid
22       laboriosam atque distinctio?</p>
23   </header>
24 </body>
25 </html>
```

estilos.css

```
7   .title{
8     display: inline-block;
9     text-decoration: none;
10    background-color: crimson;
11    color: white;
12    padding: 30px 20px;
13  }
14  .block{
15    display: none;
16  } */
17
18 .header{
19   height: 200px;
20   background-color: #antiquewhite;
21   color: #blueviolet;
22   padding: 10px;
23   border: 3px dashed ;
24 }
25
26 .title{
27   border: inherit;
28   padding: inherit;
29 }
30
31
```

especificidad.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport"
6     content="width=device-width,
7     initial-scale=1.0">
8   <link rel="stylesheet" href="css/estilos.css">
9   <title>Document</title>
10 </head>
11 <body>
12   <!-- <nav> ... -->
13   <header class="header">
14     <h2 class="title" id="title">Skill
15       diseño web</h2>
16     <p class="copy">Lorem ipsum dolor sit
17       amet consectetur adipisicing elit.
18       Molestiae laudantium libero ea esse,
19       non repellat deleniti recusandae quis?
20       Libero, id. Doloremque, omnis repellat.
21       Tenetur illum, quærat aliquid
22       laboriosam atque distinctio?</p>
23   </header>
24 </body>
25 </html>
```

estilos.css

```
7   .title{
8     display: inline-block;
9     text-decoration: none;
10    background-color: crimson;
11    color: white;
12    padding: 30px 20px;
13  }
14  .block{
15    display: none;
16  } */
17
18 .header{
19   height: 200px;
20   background-color: #antiquewhite;
21   color: #blueviolet;
22   padding: 10px;
23   border: 3px dashed ;
24   text-align: center;
25 }
26
27 .title{
28   border: inherit;
29   padding: inherit;
30 }
31
```

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport"
6     content="width=device-width,
7     initial-scale=1.0">
8   <link rel="stylesheet" href="css/estilos.css">
9   <title>Document</title>
10 </head>
11 <body>
12   <!-- <nav> ... -->
13   <header class="header">
14     <h2 class="title" id="title">Skill
15       diseño web</h2>
16     <p class="copy">Lorem ipsum dolor sit
17       amet consectetur adipisicing elit.
18       Molestiae laudantium libero ea esse,
19       non repellat deleniti recusandae quis?
20       Libero, id. Doloremque, omnis repellat.
21       Tenetur illum, quaeat aliquid
22       laboriosam atque distinctio?</p>
23   </header>
24 </body>
25 </html>

```

css > estilos.css > title
7 display: inline-block;
8 text-decoration: none;
9 background-color: crimson;
10 color: white;
11 padding: 30px 20px;
12 }
13 .block{
14 display: none;
15 } /*
16 .header{
17 height: 200px;
18 background-color: #antiquewhite;
19 color: #blueviolet;
20 padding: 10px;
21 border: 3px dashed ;
22 text-align: center;
23 }
24 .title{
25 border: inherit;
26 padding: inherit;
27 text-align: initial;
28 }
29 }
30 }
31 }
32 }

Taller de especificidad

Objetivo:

El objetivo del taller es que los participantes comprendan cómo funciona la especificidad en CSS y aprendan a utilizarla de manera efectiva para resolver conflictos de estilos.

Duración:

Aproximadamente 2 horas.

Parte 1: Introducción Teórica a la Especificidad (15 minutos)

1. ¿Qué es la especificidad?

- Explicación de cómo el navegador decide qué reglas CSS aplicar cuando hay conflictos.
- Introducción a los selectores básicos y cómo afectan la especificidad.

2. Reglas de cálculo de especificidad:

- Selectores de ID (100 puntos).
- Selectores de clase, atributos y pseudoclases (10 puntos).
- Selectores de elementos y pseudoelementos (1 punto).
- `!important` y cómo afecta la cascada.

Materiales:

- Diapositivas con ejemplos visuales de especificidad.
- Tabla de especificidad con diferentes tipos de selectores.

Parte 2: Ejemplos Prácticos (30 minutos)

1. Ejemplo básico:

- Demuestra cómo los diferentes selectores afectan la especificidad.

```

<div class="container">
  <p id="parrafo">Hola Mundo</p>
</div>

```

```

/* Estilo por etiqueta */
p {
  color: blue;
}

/* Estilo por clase */
.container p {
  color: green;
}

/* Estilo por ID */
#parrafo {
  color: red;
}

```

Resultado esperado: El color del texto será rojo, ya que el selector de ID tiene mayor especificidad que los selectores de clase y etiqueta.

2. Demostración de `!important`:

```

p {
  color: blue !important;
}

#parrafo {
  color: red;
}

```

Resultado esperado: El color será azul, porque `!important` tiene prioridad, independientemente de la especificidad.

Parte 3: Ejercicios Prácticos (45 minutos)

Distribuye a los participantes los siguientes ejercicios para que los realicen por su cuenta o en grupos pequeños:

Ejercicio 1: Calculando la Especificidad

Dado el siguiente código, pide a los participantes que calculen la especificidad y determinen qué estilos se aplicarán:

```

<div id="main" class="content">
  <h1>Título</h1>
  <p>Este es un párrafo.</p>
</div>
/* Estilo por etiqueta */
h1 {
  color: blue;
}

/* Estilo por clase */
.content h1 {
  color: green;
}

```

```
}
```

```
/* Estilo por ID */
```

```
#main h1 {
```

```
    color: red;
```

```
}
```

Pregunta:

- ¿Qué color tendrá el título `<h1>` ?

Respuesta esperada:

- El título será rojo, ya que el selector de ID tiene mayor especificidad.

Ejercicio 2: Resolviendo Conflictos de Especificidad

Modifica el siguiente código para que el párrafo tenga color amarillo, sin usar `!important`.

```
<div id="box" class="content">
```

```
    <p class="text">Texto aquí</p>
```

```
</div>
```

```
cssCopiar código#box p {
```

```
    color: blue;
```

```
}
```



```
.content .text {
```

```
    color: green;
```

```
}
```

Respuesta esperada:

Una solución es agregar un selector con mayor especificidad:

```
#box .text {
```

```
    color: yellow;
```

```
}
```

Parte 4: Desafío Final (30 minutos)

Desafío: Diseñando una Página Completa con Estilos Conflictivos

Dales a los participantes un archivo HTML con múltiples elementos y clases, y pídeles que agreguen estilos CSS para lograr un diseño específico. Deberán aplicar todo lo aprendido sobre especificidad para resolver los conflictos y obtener el resultado correcto.

HTML:

```
<div class="header" id="top">
  <h1>Bienvenido</h1>
  <p id="intro">Este es el sitio web.</p>
</div>

<div class="content">
  <h2>Contenido principal</h2>
  <p class="highlight">Texto destacado</p>
</div>

<footer id="footer">
  <p>Pie de página</p>
</footer>
```

Desafío CSS:

- El `<h1>` en el `.header` debe ser de color blanco.
- El texto del `<p>` en `.content` debe ser rojo.
- El texto del `<footer>` debe ser gris.

Pistas:

- Usar selectores combinados (clases y etiquetas).
- Resolver conflictos de estilo con la especificidad correcta.

Parte 5: Revisión y Discusión (30 minutos)

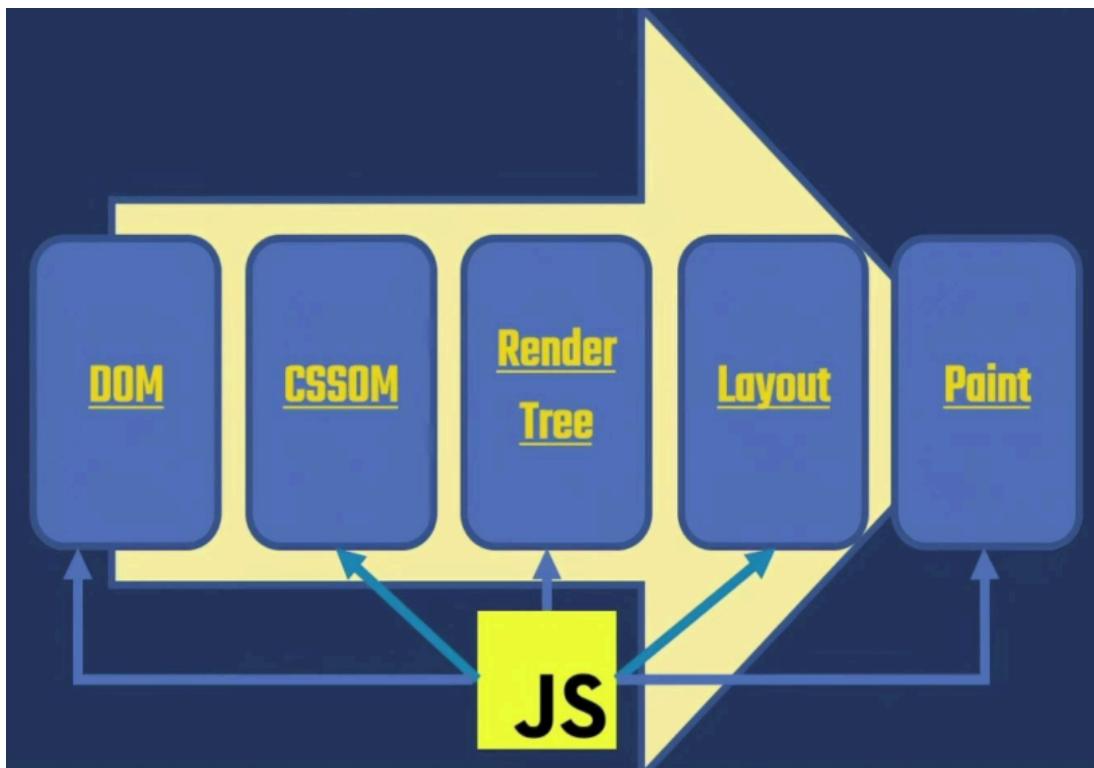
- Revisar las soluciones de los participantes.
 - Discutir errores comunes y cómo evitarlos.
 - Aclarar dudas sobre cómo calcular la especificidad y cómo usarla correctamente.
-

Conclusión:

Al final del taller, los participantes deberán:

- Entender cómo se calcula la especificidad.
- Resolver conflictos de estilo aplicando la especificidad correcta.
- Conocer cómo y cuándo usar `!important` y otras técnicas avanzadas.

Critical Rendering Path.



Este gráfico muestra el flujo de trabajo del navegador para renderizar una página web en la pantalla, desde la carga del HTML y CSS hasta que los elementos visuales aparecen en la pantalla.

Etapas de la Ruta Crítica de Representación

1. DOM (Document Object Model):

- Cuando el navegador carga el HTML, lo convierte en un árbol estructurado llamado **DOM**. Este árbol representa todos los elementos y nodos en el documento HTML.
- JavaScript puede afectar el DOM si se ejecuta en esta fase, por ejemplo, al manipular el contenido HTML.

2. CSSOM (CSS Object Model):

- Similar al DOM, pero para los estilos. El navegador convierte las reglas CSS en una estructura que llama **CSSOM**.
- JavaScript también puede afectar el CSSOM si manipula los estilos dinámicamente.

3. Render Tree:

- Una vez creados el DOM y el CSSOM, el navegador los combina para formar el **árbol de representación o Render Tree**.
- Este árbol contiene solo los elementos visibles y sus estilos, excluyendo aquellos que tienen `display: none`.

4. Layout:

- En esta etapa, el navegador calcula el **tamaño y la posición** de cada elemento visible en la página según el árbol de representación.
- Los cálculos son necesarios para saber cómo debe disponerse cada elemento en relación con los demás y con el tamaño de la pantalla.

5. Paint:

- Finalmente, en la etapa de **pintado** o **Paint**, el navegador dibuja cada elemento en la pantalla.

Rol de JavaScript

- JavaScript puede afectar tanto el **DOM** como el **CSSOM** en diferentes momentos del proceso de renderizado.
- Si JavaScript modifica el DOM o CSSOM después de haber creado el Render Tree, puede provocar **re-flujos** o **re-paints**, que son costosos en términos de rendimiento.

Resumen

- **DOM y CSSOM**: Estructuras del contenido y estilos de la página.
- **Render Tree**: Combina DOM y CSSOM para mostrar solo los elementos visibles.
- **Layout**: Calcula posición y tamaño de cada elemento.
- **Paint**: Renderiza en pantalla los elementos con sus estilos.

Unidades de medida

En CSS, existen varias unidades de medida que puedes utilizar para definir tamaños, márgenes, espacios, entre otros. Estas se dividen principalmente en unidades **relativas** y **absolutas**

1. Unidades Absolutas

Estas unidades tienen un tamaño fijo y no cambian en función del entorno.

- **px (píxeles)**: es la unidad más utilizada y representa un píxel en la pantalla. Es ideal para elementos con tamaños fijos.
- **cm (centímetros), mm (milímetros), in (pulgadas)**: unidades físicas, menos comunes en la web, pero útiles en impresión.
- **pt (puntos) y pc (picas)**: principalmente usados en diseño gráfico e impresión. Un punto equivale a 1/72 de pulgada.

2. Unidades Relativas

Estas unidades dependen del contexto, como el tamaño de la pantalla, el tamaño de fuente, o el tamaño del contenedor.

- **% (porcentaje)**: es relativo al tamaño del elemento contenedor. Útil para diseños responsivos.
- **em**: relativo al tamaño de fuente del elemento padre. Si el elemento padre tiene `font-size: 16px`, entonces `1em` será igual a `16px`.

- **rem (root em)**: similar a `em`, pero en este caso, se basa en el tamaño de fuente del elemento raíz (`html`). Es consistente en toda la página.
- **vw (viewport width) y vh (viewport height)**: porcentajes del tamaño de la ventana gráfica. `1vw` es el 1% del ancho de la pantalla y `1vh` es el 1% de la altura.
- **vmin y vmax**: son relativos a la dimensión más pequeña o más grande de la ventana gráfica, respectivamente. `1vmin` es el 1% de la dimensión menor, y `1vmax` es el 1% de la dimensión mayor.
- **ch**: basado en el ancho del carácter "0" de la fuente utilizada en el elemento.
- **ex**: basado en la altura de la "x" minúscula de la fuente del elemento.

Ejemplo

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ejemplo de Unidades CSS</title>
    <style>
        /* Estilos base */
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            font-size: 16px; /* Tamaño base para cálculos rem */
        }

        /* Encabezado con altura fija (unidad absoluta) */
        header {
            background-color: #3498db;
            color: white;
            padding: 20px; /* Unidad absoluta */
            height: 60px; /* Unidad absoluta */
        }

        h1 {
            margin: 0;
            font-size: 2rem; /* Unidad relativa al tamaño de fuente del root
(html) */
        }

        /* Contenedor principal con ancho relativo */
        .container {
            width: 80%; /* Unidad relativa al ancho del contenedor padre */
            margin: 0 auto;
            padding: 20px;
        }

        /* Caja con dimensiones mixtas */
        .box {
            background-color: #ecf0f1;
            width: 300px; /* Unidad absoluta */
            height: 15em; /* Unidad relativa al tamaño de fuente del elemento */
        }
    </style>
</head>
<body>
    <header>
        <h1>Encabezado</h1>
    </header>
    <div class="container">
        <p>Este es el contenido principal de la página.</p>
        <div class="box">
            <h2>Caja mixta</h2>
            <p>Este es el contenido de la caja mixta.</p>
        </div>
    </div>
</body>
</html>
```

```

        margin-bottom: 20px;
        padding: 1em; /* Unidad relativa al tamaño de fuente del elemento */
        border: 2px solid #34495e;
    }

    /* Texto con tamaño relativo */
    .text {
        font-size: 1.2em; /* Unidad relativa al tamaño de fuente del elemento
padre */
        line-height: 1.5;
    }

    /* Botón con dimensiones relativas y absolutas */
    .button {
        display: inline-block;
        padding: 10px 20px; /* Unidades absolutas */
        background-color: #2ecc71;
        color: white;
        text-decoration: none;
        border-radius: 5px;
        font-size: 1rem; /* Unidad relativa al tamaño de fuente del root */
        transition: background-color 0.3s;
    }

    .button:hover {
        background-color: #27ae60;
    }

    /* Elemento con ancho basado en el viewport */
    .viewport-width {
        background-color: #e74c3c;
        color: white;
        padding: 10px;
        width: 50vw; /* Unidad relativa al ancho del viewport */
        margin-top: 20px;
    }

```

</style>

</head>

<body>

 <header>

 <h1>Ejemplo de Unidades CSS</h1>

 </header>

 <div class="container">

 <div class="box">

 <p class="text">Este es un párrafo dentro de una caja. La caja tiene
un ancho fijo en píxeles y una altura en em.</p>

 </div>

 Botón de Ejemplo

 <div class="viewport-width">Este elemento tiene un ancho del 50% del
viewport.</div>

 </div>

</body>

</html>

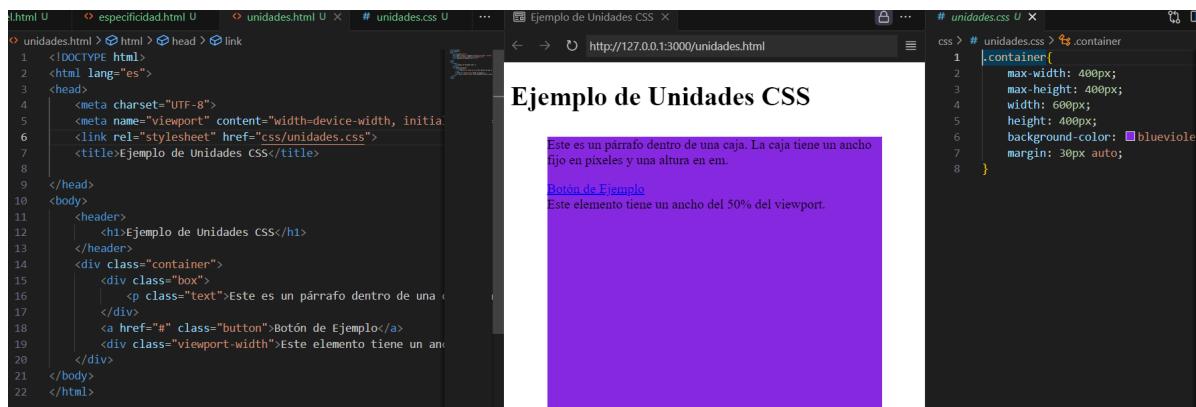
Ejemplo de Unidades CSS

Este es un párrafo dentro de una caja. La caja tiene un ancho fijo en píxeles y una altura en em.

Botón de Ejemplo

Este elemento tiene un ancho del 50% del viewport.

Limitar medidas y porcentajes



The screenshot shows a browser window with developer tools. On the left, the HTML code is visible, including the container class definition. In the center, the page content is displayed with the specified styling. On the right, the CSS file is shown with the container class definition.

```
html U   especificidad.html U   unidades.html U   # unidades.css U ...
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <link rel="stylesheet" href="css/unidades.css">
7      <title>Ejemplo de Unidades CSS</title>
8
9  </head>
10 <body>
11     <header>
12         <h1>Ejemplo de Unidades CSS</h1>
13     </header>
14     <div class="container">
15         <div class="box">
16             <p class="text">Este es un párrafo dentro de una caja. La caja tiene un ancho fijo en píxeles y una altura en em.</p>
17         </div>
18         <a href="#" class="button">Botón de Ejemplo</a>
19     <div class="viewport-width">Este elemento tiene un ancho del 50% del viewport.</div>
20 </div>
21 </body>
22 </html>
```

```
# unidades.css U ...
# unidades.css > .container
1 .container{
2     max-width: 400px;
3     max-height: 400px;
4     width: 600px;
5     height: 400px;
6     background-color: blueviolet;
7     margin: 30px auto;
8 }
```

```
.container{
    max-width: 400px;
    max-height: 400px;
    width: 600px;
    height: 400px;
    background-color: blueviolet;
    margin: 30px auto;
}
```

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="css/unidades.css">
    <title>Ejemplo de Unidades CSS</title>

</head>
<body>
```

```

<header>
    <h1>Ejemplo de Unidades CSS</h1>
</header>
<div class="container">
    <div class="box">
        <p class="text">Este es un párrafo dentro de una caja. La caja tiene
        un ancho fijo en píxeles y una altura en em.</p>
    </div>
    <a href="#" class="button">Botón de Ejemplo</a>
    <div class="viewport-width">Este elemento tiene un ancho del 50% del
    viewport.</div>
</div>
</body>
</html>

```

Limitando Ancho y Alto

```

.responsive-box {
    width: 80%;           /* El ancho es del 80% del contenedor */
    max-width: 500px;      /* No puede ser mayor de 500px */
    min-width: 200px;      /* No puede ser menor de 200px */
    height: 50vh;          /* La altura es del 50% de la pantalla */
    max-height: 400px;      /* No puede ser mayor de 400px */
    min-height: 150px;      /* No puede ser menor de 150px */
    background-color: lightblue;
}

```

Usando Limitaciones en Layouts Responsivos

```

.column {
    width: 70%;
    min-width: 300px;      /* Ancho mínimo de 300px */
    max-width: 800px;      /* Ancho máximo de 800px */
    background-color: lightcoral;
}

```

```

.card {
    width: 90vw;           /* 90% del ancho de la ventana */
    max-width: 600px;       /* Máximo 600px de ancho */
    min-width: 280px;       /* Mínimo 280px de ancho */
    height: 50vh;          /* 50% de la altura de la ventana */
    max-height: 400px;      /* Máximo 400px de alto */
    min-height: 200px;      /* Mínimo 200px de alto */
    padding: 20px;
    background-color: lightgreen;
    border: 2px solid #333;
    border-radius: 10px;
}

```

Limitando Font Sizes en CSS

```
.responsive-text {  
    font-size: clamp(1rem, 2vw + 1rem, 2.5rem);  
}
```

1rem es el tamaño mínimo.

2vw + 1rem es el tamaño preferido (una fórmula que ajusta según el ancho del viewport).

2.5rem es el tamaño máximo.

Otros casos de uso

```
.container {  
    width: 150px; /* Define un ancho fijo de 150 píxeles para el  
    contenedor */  
    height: 54px; /* Define una altura fija de 54 píxeles para el  
    contenedor */  
    background-color: bisque; /* Establece el color de fondo del contenedor a  
    bisque (un tono claro) */  
    padding: 10px; /* Añade un espacio interno de 10 píxeles en todos  
    los lados del contenido */  
    border: solid 5px black; /* Aplica un borde sólido de 5 píxeles de grosor y  
    color negro */  
    font-size: 28px; /* Establece el tamaño de la fuente en 28 píxeles  
*/  
    box-sizing: border-box; /* Incluye el padding y el borde en el cálculo del  
    ancho y alto total */  
    overflow: visible; /* Permite que el contenido se desborde fuera de  
    los límites si es mayor */  
}  
-----  
.container {  
    width: 150px; /* Define un ancho fijo de 150 píxeles para el  
    contenedor */  
    height: 54px; /* Define una altura fija de 54 píxeles para el  
    contenedor */  
    background-color: bisque; /* Establece el color de fondo del contenedor a  
    bisque (un tono claro) */  
    padding: 10px; /* Añade un espacio interno de 10 píxeles en todos  
    los lados del contenido */  
    border: solid 5px black; /* Aplica un borde sólido de 5 píxeles de grosor y  
    color negro */  
    font-size: 28px; /* Establece el tamaño de la fuente en 28 píxeles  
*/  
    box-sizing: border-box; /* Incluye el padding y el borde en el cálculo del  
    ancho y alto total */  
    overflow: hidden; /* Permite que el contenido se oculte y no se  
    desborde si es mayor */  
}
```

```
.container {  
    width: 150px; /* Define un ancho fijo de 150 píxeles para el  
contenedor */  
    height: 54px; /* Define una altura fija de 54 píxeles para el  
contenedor */  
    background-color: bisque; /* Establece el color de fondo del contenedor a  
bisque */  
    padding: 10px; /* Añade un espacio interno de 10 píxeles en todos  
los lados del contenido */  
    border: solid 5px black; /* Aplica un borde sólido de 5 píxeles de grosor y  
color negro */  
    font-size: 28px; /* Establece el tamaño de la fuente en 28 píxeles  
*/  
    box-sizing: border-box; /* Incluye el padding y el borde en el cálculo del  
ancho y alto total */  
    overflow: scroll; /* Agrega barras de desplazamiento si el contenido  
es mayor que el área visible */  
}  
/*-----  
.container {  
    width: 160px; /* Define un ancho fijo de 160 píxeles para el  
contenedor */  
    height: 74px; /* Define una altura fija de 74 píxeles para el  
contenedor */  
    background-color: bisque; /* Establece el color de fondo a bisque */  
    padding: 10px; /* Añade un espacio interno de 10 píxeles alrededor  
del contenido */  
    border: solid 5px black; /* Aplica un borde sólido de 5 píxeles de grosor y  
color negro */  
    font-size: 28px; /* Establece el tamaño de fuente en 28 píxeles */  
    box-sizing: border-box; /* Incluye el padding y el borde en el cálculo del  
ancho y alto total */  
    overflow: auto; /* Agrega barras de desplazamiento solo si el  
contenido excede el área visible */  
}  
/*-----  
.container {  
    width: 160px; /* Define un ancho fijo de 160 píxeles para el  
contenedor */  
    height: 74px; /* Define una altura fija de 74 píxeles para el  
contenedor */  
    background-color: bisque; /* Establece el color de fondo a bisque */  
    padding: 10px; /* Añade un espacio interno de 10 píxeles alrededor  
del contenido */  
    border: solid 5px black; /* Aplica un borde sólido de 5 píxeles de grosor y  
color negro */  
    font-size: 28px; /* Establece el tamaño de fuente en 28 píxeles */  
    box-sizing: border-box; /* Incluye el padding y el borde en el cálculo del  
ancho y alto total */
```

```
overflow: auto;           /* Agrega barras de desplazamiento solo si el
                           contenido excede el área visible */
text-overflow: ellipsis; /* Muestra "..." si el texto es más largo que el
                           ancho del contenedor */
}
```

Nueva Propiedad `text-overflow: ellipsis`

- **text-overflow: ellipsis:** Esta propiedad es útil cuando tienes texto que podría desbordarse fuera de los límites del contenedor. Al agregar `text-overflow: ellipsis`, el texto que no cabe en el ancho especificado será truncado y se mostrará con puntos suspensivos (...) al final, indicando que hay más contenido.

Condiciones para que `text-overflow: ellipsis` funcione

Para que esta propiedad funcione correctamente, es necesario cumplir con dos condiciones:

1. **overflow: hidden o overflow: auto:** Debe estar definido el `overflow` para restringir el contenido del contenedor.
2. **white-space: nowrap:** El texto debe estar en una sola línea, lo que significa que el contenedor no permite el salto de línea.

```
.container {
  width: 160px;
  height: 74px;
  background-color: bisque;
  padding: 10px;
  border: solid 5px black;
  font-size: 28px;
  box-sizing: border-box;
  overflow: auto;
  white-space: nowrap;      /* Evita el salto de línea */
  text-overflow: ellipsis;  /* Muestra "..." si el texto es más largo que el
                           contenedor */
}
```

Cambio en la Propiedad `overflow`

- **overflow: scroll:** Con esta configuración, el contenedor mostrará barras de desplazamiento (horizontal y vertical) incluso si el contenido cabe dentro de su área visible.

Posicionamiento de cajas

En CSS, puedes posicionar cajas (elementos) de varias maneras, usando diferentes valores de la propiedad `position`. Cada uno permite un control distinto sobre cómo se posicionan los elementos en el flujo del documento y en relación con otros elementos.

Position: static (Posicionamiento Estático)

Es el valor predeterminado de `position` en CSS. Los elementos con `position: static` siguen el flujo normal del documento y no pueden ser desplazados mediante `top`, `right`, `bottom`, o `left`.

Ejemplo

```
.static-box {  
    position: static;  
    width: 200px;  
    height: 100px;  
    background-color: lightblue;  
}
```

Position: relative (Posicionamiento Relativo)

Con `position: relative`, el elemento permanece en el flujo del documento, pero se puede mover respecto a su posición original usando `top`, `right`, `bottom`, y `left`. Sin embargo, deja su espacio en blanco en su posición original, como si no se hubiera movido.

```
.relative-box {  
    position: relative;  
    top: 20px; /* Se mueve 20px hacia abajo */  
    left: 10px; /* Se mueve 10px hacia la derecha */  
    width: 200px;  
    height: 100px;  
    background-color: lightcoral;  
}
```

Position: absolute (Posicionamiento Absoluto)

Un elemento con `position: absolute` se posiciona en relación con el primer contenedor padre que tenga `position` distinto de `static` (puede ser `relative`, `absolute` o `fixed`). Si no tiene un contenedor padre posicionado, se posicionará en relación al `body` o al viewport. El elemento se saca del flujo normal del documento.

```
.absolute-box {  
    position: absolute;  
    top: 30px;  
    right: 20px;  
    width: 200px;  
    height: 100px;  
    background-color: lightgreen;  
}
```

Position: fixed (Posicionamiento Fijo)

Con `position: fixed`, el elemento se posiciona en relación con el viewport, por lo que permanecerá en la misma posición cuando se haga scroll. Como `absolute`, también se saca del flujo normal del documento.

```
.fixed-box {  
    position: fixed;  
    bottom: 10px;  
    right: 10px;  
    width: 150px;  
    height: 50px;  
    background-color: lightyellow;  
}
```

Position: sticky (Posicionamiento Adhesivo)

`position: sticky` permite que un elemento actúe como `relative` mientras esté dentro del viewport, pero se "pegará" a una posición específica cuando se haga scroll y esa posición sea alcanzada. Para que funcione, es necesario especificar al menos una de las propiedades `top`, `right`, `bottom`, o `left`.

```
.sticky-box {  
    position: sticky;  
    top: 0; /* Se pega a la parte superior cuando se alcanza */  
    width: 100%;  
    height: 50px;  
    background-color: lightpink;  
}
```

Cuándo Usar Cada Tipo

- **Static:** Para el posicionamiento estándar de elementos en el flujo normal.
- **Relative:** Para mover elementos en relación con su posición original.
- **Absolute:** Para elementos que necesitan una posición exacta dentro de un contenedor específico.
- **Fixed:** Para elementos que deben permanecer visibles mientras se hace scroll (ej., menús flotantes).
- **Sticky:** Para elementos que deben "pegarse" a la pantalla al hacer scroll (ej., cabeceras en tablas o secciones).

Taller

Cree un documento Html con el siguiente código:

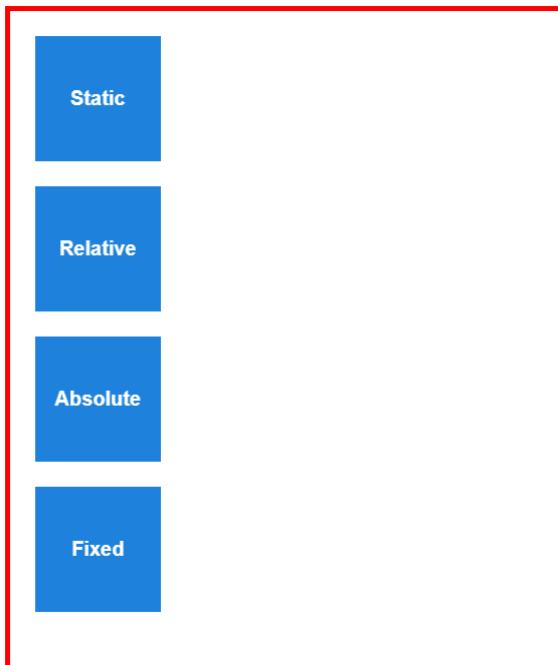
```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```

<link rel="stylesheet" href="css/posicion.css">
<title>Document</title>
</head>
<body>
  <div class="container">
    <div class="box static">Static</div>
    <div class="box relative">Relative</div>
    <div class="box absolute">Absolute</div>
    <div class="box fixed">Fixed</div>
  </div>
</body>
</html>

```

Cree una hoja de estilos para lograr el siguiente aspecto de la pagina:



z-index

`z-index` es una propiedad en CSS que define el orden de apilamiento de elementos HTML en el eje Z, o en otras palabras, determina qué elementos se ven "encima" o "debajo" de otros cuando se superponen en una página web.

Conceptos Clave del `z-index`

1. **Orden de Apilamiento:** Los elementos con un `z-index` más alto se mostrarán sobre aquellos con un `z-index` más bajo.
2. **Necesidad de Posicionamiento:** Solo funciona en elementos que tienen una propiedad `position` configurada en algo distinto de `static` (como `relative`, `absolute`, `fixed`, o `sticky`).
3. **Valores:** Puede ser positivo, negativo o cero:
 - **Positivos:** Colocan el elemento encima de otros con un `z-index` menor.
 - **Negativos:** Ponen el elemento por debajo.
 - **Cero o Auto:** Sigue el orden natural del flujo de HTML si no se especifica un valor.

4. Contexto de Apilamiento: Un elemento con `position` crea un nuevo "contexto de apilamiento" para sus elementos hijos, lo que significa que el `z-index` solo afectará el orden entre elementos dentro de ese contexto, sin influir fuera de él.

```
.cuadro1 {  
    position: relative;  
    z-index: 1; /* Se apilará debajo */  
}  
  
.cuadro2 {  
    position: relative;  
    z-index: 2; /* Se apilará encima */  
}
```

Ejercicio Posicionamiento

1. Cree una nueva estructura de proyecto web
2. Ingrese a la Url <https://unsplash.com/es> y descargue una imagen
3. Busque una letra de su preferencia.
4. En el Html Agrega el siguiente codigo

```
index.html > html > body > header.hero > div.container-text > a.hero_cta  
1  !DOCTYPE html  
2  <html lang="en">  
3  <head>  
4      <meta charset="UTF-8">  
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">  
6      <link rel="stylesheet" href="css/styles.css">  
7      <title>Document</title>  
8  </head>  
9  <body>  
10     <header class="hero">  
11         <div class="container-text">  
12             <h1 class="hero__title">Desarrollo web desde Cero a Experto</h1>  
13             <p class="hero__copy">Desarrolla tus habilidades en CSS estudiante Desarrollo web</p>  
14             <a href="#" class="hero__cta">Ingresa Ahora</a>  
15         </div>  
16     </header>  
17  </body>  
18  </html>
```

5. En el CSS agregue el siguiente código tenga en cuenta que el nombre de la fuente depende del seleccionado en google fonts

```
css > styles.css > body  
1  @import url('https://fonts.googleapis.com/css2?family=Raleway:ital,wght@0,100..900;1,100..900&  
family=Roboto:ital,wght@0,100;0,300;0,400;0,500;0,700;0,900;1,100;1,300;1,400;1,500;1,700;1,900&display=swap');
```

6. Ejecute el html usando Live server



Desarrollo web desde Cero a Experto

Desarrolla tus habilidades en CSS estudiante Desarrollo web

[Ingresá Ahora](#)

Recurso web generación waves <https://smooth.ie/blogs/news/svg-wavey-transitions-between-sections>

7. Construya una hoja de estilo que permita generar el siguiente diseño final



Ejercicio de posicionamiento Menú Básico

URL Recursos : <https://webgradients.com>

`:nth-of-type` en CSS permite seleccionar elementos en función de su posición dentro de su tipo específico (es decir, su etiqueta) en un contenedor. Esto es útil para aplicar estilos a elementos específicos de un mismo tipo sin tener que añadir clases adicionales.

The screenshot shows a web browser window with the URL 127.0.0.1:5501/menu.html#inicio in the address bar. The page has a purple header with navigation links: Inicio, Acerca de, and Contacto. The main content area has a purple gradient background and displays the word "Inicio" in white.

Pseudoclases

Las pseudoclases en CSS son palabras clave que se añaden a selectores para aplicar estilos en un estado específico de un elemento. Ayudan a mejorar la interacción y la experiencia visual, ya que permiten cambiar el estilo de un elemento en función de su estado, posición, o de la interacción del usuario.

Pseudoclase	Función
<code>:hover</code>	Aplica estilos cuando el usuario pasa el cursor sobre un elemento.
<code>:active</code>	Aplica estilos cuando un elemento está en estado activo, por ejemplo, al hacer clic en él.
<code>:focus</code>	Aplica estilos cuando un elemento recibe el foco (por ejemplo, al hacer clic en un input).
<code>:nth-child(n)</code>	Selecciona el enésimo hijo de un elemento padre, donde <code>n</code> puede ser un número o expresión.
<code>:first-child</code>	Selecciona el primer hijo de un elemento padre.
<code>:last-child</code>	Selecciona el último hijo de un elemento padre.
<code>:not(selector)</code>	Selecciona elementos que no cumplen con el selector especificado dentro de <code>:not()</code> .
<code>:checked</code>	Aplica estilos a elementos de formulario (<code>checkbox</code> o <code>radio</code>) cuando están marcados.
<code>:disabled</code>	Aplica estilos a elementos de formulario que están deshabilitados.
<code>:enabled</code>	Aplica estilos a elementos de formulario que están habilitados.
<code>:required</code>	Aplica estilos a elementos de formulario que son obligatorios.
<code>:optional</code>	Aplica estilos a elementos de formulario que no son obligatorios.
<code>:first-of-type</code>	Selecciona el primer elemento de su tipo dentro de su elemento padre.
<code>:last-of-type</code>	Selecciona el último elemento de su tipo dentro de su elemento padre.
<code>:only-child</code>	Selecciona elementos que son el único hijo de su elemento padre.

Pseudoclase	Función
<code>:empty</code>	Selecciona elementos que no tienen hijos, incluyendo texto o elementos hijos.
<code>:root</code>	Selecciona el elemento raíz del documento (generalmente el elemento <code><html></code>).
<code>:before</code>	Inserta contenido antes del contenido de un elemento, usando <code>content</code> .
<code>:after</code>	Inserta contenido después del contenido de un elemento, usando <code>content</code> .
<code>:link</code>	Aplica estilos a enlaces (<code><a></code>) que aún no han sido visitados.
<code>:visited</code>	Aplica estilos a enlaces (<code><a></code>) que han sido visitados.

Ejercicio

```

body {
    margin: 0;
    min-height: 100vh;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%);
}

.container {
    max-width: 800px;
    margin: 0 auto;
    padding: 2rem;
}

h1 {
    text-align: center;
    color: #2d3436;
}

.card {
    background: white;
    border-radius: 8px;
    padding: 1.5rem;
    margin: 1rem 0;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

/* Hover Button */
.hover-button {
    padding: 10px 20px;
    border: none;
    border-radius: 4px;
    background-color: #6c5ce7;
    color: white;
    cursor: pointer;
    transition: all 0.3s ease;
}

```

```
}

.hover-button:hover {
  background-color: #5b4cdb;
  transform: translateY(-2px);
  box-shadow: 0 4px 8px rgba(0,0,0,0.2);
}

.hover-button:active {
  transform: translateY(0);
}

/* Custom Link */
.custom-link {
  color: #0984e3;
  text-decoration: none;
  position: relative;
}

.custom-link:visited {
  color: #6c5ce7;
}

.custom-link:hover {
  color: #74b9ff;
}

.custom-link:hover::after {
  content: '';
  position: absolute;
  width: 100%;
  height: 2px;
  bottom: -2px;
  left: 0;
  background-color: currentColor;
  transition: width 0.3s ease;
}

/* Custom Input */
.custom-input {
  padding: 8px 12px;
  border: 2px solid #dfe6e9;
  border-radius: 4px;
  width: 200px;
  transition: all 0.3s ease;
}

.custom-input:focus {
  outline: none;
  border-color: #6c5ce7;
  box-shadow: 0 0 0 3px rgba(108, 92, 231, 0.2);
}

.custom-input:invalid {
  border-color: #ff7675;
}
```

```
/* Custom List */
.custom-list {
  list-style: none;
  padding: 0;
}

.custom-list li {
  padding: 8px 12px;
  margin: 4px 0;
  background-color: #f5f6fa;
  border-radius: 4px;
  transition: all 0.3s ease;
}

.custom-list li:first-child {
  background-color: #a8e6cf;
}

.custom-list li:last-child {
  background-color: #ffd3b6;
}

.custom-list li:nth-child(even) {
  background-color: #dfe6e9;
}

.custom-list li:hover {
  transform: translateX(10px);
  background-color: #81ecec;
}

/* Checkbox and Radio Styles */
.checkbox-group, .radio-group {
  margin: 1rem 0;
}

.custom-checkbox, .custom-radio {
  display: block;
  position: relative;
  padding-left: 35px;
  margin-bottom: 12px;
  cursor: pointer;
  user-select: none;
}

.custom-checkbox input, .custom-radio input {
  position: absolute;
  opacity: 0;
  cursor: pointer;
  height: 0;
  width: 0;
}

.checkmark, .radio-mark {
  position: absolute;
```

```
top: 0;
left: 0;
height: 25px;
width: 25px;
background-color: #eee;
border-radius: 4px;
transition: all 0.2s ease;
}

.radio-mark {
  border-radius: 50%;
}

.custom-checkbox:hover input ~ .checkmark,
.custom-radio:hover input ~ .radio-mark {
  background-color: #ccc;
}

.custom-checkbox input:checked ~ .checkmark,
.custom-radio input:checked ~ .radio-mark {
  background-color: #6c5ce7;
}

.checkmark:after,
.radio-mark:after {
  content: "";
  position: absolute;
  display: none;
}

.custom-checkbox input:checked ~ .checkmark:after,
.custom-radio input:checked ~ .radio-mark:after {
  display: block;
}

.custom-checkbox .checkmark:after {
  left: 9px;
  top: 5px;
  width: 5px;
  height: 10px;
  border: solid white;
  border-width: 0 3px 3px 0;
  transform: rotate(45deg);
}

.custom-radio .radio-mark:after {
  top: 9px;
  left: 9px;
  width: 8px;
  height: 8px;
  border-radius: 50%;
  background: white;
}

/* Required and Optional Fields */
.form-example {
```

```
display: flex;
flex-direction: column;
gap: 1rem;
}

.input-group input {
  padding: 8px 12px;
  border: 2px solid #dfe6e9;
  border-radius: 4px;
  width: 100%;
  box-sizing: border-box;
}

.input-group input:required {
  border-left: 4px solid #6c5ce7;
}

.input-group input:optional {
  border-left: 4px solid #00b894;
}

/* Enabled vs Disabled Buttons */
.state-button {
  padding: 10px 20px;
  margin: 0 10px;
  border: none;
  border-radius: 4px;
  background-color: #00b894;
  color: white;
  cursor: pointer;
  transition: all 0.3s ease;
}

.state-button:disabled {
  background-color: #b2bec3;
  cursor: not-allowed;
  opacity: 0.7;
}

/* Empty and Not Empty */
.empty-box {
  border: 2px solid #dfe6e9;
  padding: 1rem;
  margin: 0.5rem 0;
  min-height: 50px;
  border-radius: 4px;
}

.empty-box:empty {
  background-color: #ffeaaf;
  border-style: dashed;
}

/* Not Selector */
.not-example p {
  padding: 0.5rem;
```

```
margin: 0.5rem 0;  
background-color: #dfe6e9;  
border-radius: 4px;  
}  
  
.not-example p:not(.special) {  
background-color: #fab1a0;  
color: #2d3436;  
}
```

```
<!DOCTYPE html>  
<html lang="en">  
  <head>  
    <meta charset="UTF-8" />  
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />  
    <title>CSS Pseudo-classes Demo</title>  
    <link rel="stylesheet" href="style.css">  
  </head>  
  <body>  
    <div class="container">  
      <h1>CSS Pseudo-classes Demo</h1>  
  
      <div class="card">  
        <h2>Hover Effects</h2>  
        <button class="hover-button">Hover me!</button>  
      </div>  
  
      <div class="card">  
        <h2>Links</h2>  
        <a href="#" class="custom-link">Click me</a>  
      </div>  
  
      <div class="card">  
        <h2>Input States</h2>  
        <input type="text" placeholder="Focus me!" class="custom-input">  
      </div>  
  
      <div class="card">  
        <h2>List Items</h2>  
        <ul class="custom-list">  
          <li>First item</li>  
          <li>Second item</li>  
          <li>Third item</li>  
          <li>Fourth item</li>  
          <li>Fifth item</li>  
        </ul>  
      </div>  
  
      <div class="card">  
        <h2>Checkbox and Radio</h2>  
        <div class="checkbox-group">  
          <label class="custom-checkbox">  
            <input type="checkbox">  
            <span class="checkmark"></span>  
            Check me  
          </label>  
        </div>  
      </div>  
    </div>  
  </body>  
</html>
```

```

        </label>
    </div>
    <div class="radio-group">
        <label class="custom-radio">
            <input type="radio" name="radio" checked>
            <span class="radio-mark"></span>
            Option 1
        </label>
        <label class="custom-radio">
            <input type="radio" name="radio">
            <span class="radio-mark"></span>
            Option 2
        </label>
    </div>
</div>

<div class="card">
    <h2>Required and Optional Fields</h2>
    <form class="form-example">
        <div class="input-group">
            <input type="text" required placeholder="Required field">
        </div>
        <div class="input-group">
            <input type="text" placeholder="Optional field">
        </div>
    </form>
</div>

<div class="card">
    <h2>Enabled vs Disabled</h2>
    <button class="state-button">Enabled Button</button>
    <button class="state-button" disabled>Disabled Button</button>
</div>

<div class="card">
    <h2>Empty and Not Empty</h2>
    <div class="empty-box"></div>
    <div class="empty-box">Not empty</div>
</div>

<div class="card">
    <h2>Not Selector</h2>
    <div class="not-example">
        <p class="special">Special paragraph</p>
        <p>Normal paragraph 1</p>
        <p>Normal paragraph 2</p>
    </div>
</div>
</body>
</html>

```

Ejemplo 2

```
body {
```

```
margin: 0;
min-height: 100vh;
font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
background: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%);
color: #2d3436;
}

.container {
  max-width: 800px;
  margin: 0 auto;
  padding: 2rem;
}

h1, h2 {
  color: #2d3436;
}

.card {
  background: white;
  border-radius: 8px;
  padding: 1.5rem;
  margin: 1rem 0;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
  transition: all 0.3s ease;
}

/* Table of Contents */
.table-of-contents {
  background: white;
  padding: 1rem;
  border-radius: 8px;
  margin-bottom: 2rem;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

.table-of-contents a {
  display: block;
  color: #6c5ce7;
  text-decoration: none;
  padding: 0.5rem;
  margin: 0.25rem 0;
  border-radius: 4px;
  transition: all 0.3s ease;
}

.table-of-contents a:hover {
  background: #f0f2f5;
  padding-left: 1rem;
}

/* Target Example */
.target-example:target {
  background: #ffeaaf;
  transform: scale(1.02);
  box-shadow: 0 4px 8px rgba(0,0,0,0.2);
}
```

```
/* Language Example */
p[lang="en"] {
  border-left: 4px solid #00b894;
}

p[lang="es"] {
  border-left: 4px solid #e17055;
}

p[lang="fr"] {
  border-left: 4px solid #74b9ff;
}

/* Form styles */
.form-group {
  margin-bottom: 1.5rem;
}

.form-group label {
  display: block;
  margin-bottom: 0.5rem;
  color: #2d3436;
  font-weight: 500;
}

.form-group input {
  width: 100%;
  padding: 0.75rem;
  border: 2px solid #dfe6e9;
  border-radius: 4px;
  font-size: 1rem;
  transition: all 0.3s ease;
  box-sizing: border-box;
}

/* Focus Within Example */
.search-wrapper {
  position: relative;
  display: flex;
  align-items: center;
}

.search-icon {
  position: absolute;
  right: 1rem;
  color: #b2bec3;
  transition: all 0.3s ease;
}

.focus-within-demo:focus-within .search-icon {
  color: #6c5ce7;
  transform: scale(1.1);
}

/* Form Validation States */
```

```
input:valid {
  border-color: #00b894;
}

input:invalid {
  border-color: #ff7675;
}

input:optional {
  border-style: dashed;
}

input:required {
  border-style: solid;
}

/* Number Range */
input[type="number"]:in-range {
  background-color: #a8e6cf;
}

input[type="number"]:out-of-range {
  background-color: #ffb8b8;
}

/* Selection Example */
.selection-demo::selection {
  background-color: #6c5ce7;
  color: white;
}

/* Focus Order Example */
.focus-order {
  display: flex;
  gap: 1rem;
}

.focus-order button {
  padding: 0.75rem 1.5rem;
  border: none;
  border-radius: 4px;
  background-color: #6c5ce7;
  color: white;
  cursor: pointer;
  transition: all 0.3s ease;
}

.focus-order button:hover {
  background-color: #5b4cdb;
}

.focus-order button:focus {
  outline: none;
  box-shadow: 0 0 0 3px rgba(108, 92, 231, 0.4);
}
```

```

/* Placeholder styling */
input::placeholder {
  color: #b2bec3;
  font-style: italic;
}

/* Focus States */
input:focus {
  outline: none;
  border-color: #6c5ce7;
  box-shadow: 0 0 0 3px rgba(108, 92, 231, 0.2);
}

/* Hover States */
.card:hover {
  transform: translateY(-2px);
  box-shadow: 0 4px 8px rgba(0,0,0,0.1);
}

```

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>More CSS Pseudo-classes Demo</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <div class="container">
      <h1>More CSS Pseudo-classes</h1>

      <nav class="table-of-contents">
        <h2>Jump to Section:</h2>
        <a href="#section1">Target Example</a>
        <a href="#section2">Language Example</a>
        <a href="#section3">Form Validation</a>
        <a href="#section4">Number Range</a>
      </nav>

      <div id="section1" class="card target-example">
        <h2>:target Example</h2>
        <p>This section will highlight when targeted through the URL hash.</p>
      </div>

      <div id="section2" class="card">
        <h2>:lang Example</h2>
        <p lang="en">This is English text</p>
        <p lang="es">Este es un texto en español</p>
        <p lang="fr">C'est un texte en français</p>
      </div>

      <div id="section3" class="card">
        <h2>Form Validation States</h2>
        <form class="validation-form">
          <div class="form-group">

```

```
<label>Email (required):</label>
<input type="email" required placeholder="Enter valid email">
</div>

<div class="form-group">
  <label>Website (optional):</label>
  <input type="url" placeholder="https://example.com">
</div>

<div class="form-group focus-within-demo">
  <label>Search with icon:</label>
  <div class="search-wrapper">
    <input type="search" placeholder="Type to search...">
    <span class="search-icon">🔍</span>
  </div>
</div>

<div class="form-group">
  <label>Password Strength:</label>
  <input type="password" pattern=".{8,}" placeholder="Min 8
characters">
</div>
</form>
</div>

<div id="section4" class="card">
  <h2>Number Range Example</h2>
  <div class="form-group">
    <label>Age (between 18-100):</label>
    <input type="number" min="18" max="100" placeholder="Enter age">
  </div>

  <div class="form-group">
    <label>Score (0-10):</label>
    <input type="number" min="0" max="10" step="0.5" placeholder="Enter
score">
  </div>
</div>

<div class="card">
  <h2>Selection Example</h2>
  <p class="selection-demo">Try selecting this text to see custom selection
colors!</p>
</div>

<div class="card">
  <h2>Focus Order Example</h2>
  <div class="focus-order">
    <button>First</button>
    <button>Second</button>
    <button>Third</button>
  </div>
</div>
</div>
</body>
</html>
```

Taller

Ejercicio

Hacer una pagina donde se pueda visualizar un listado de partes para computador. En el Stock hay 15 Productos. La información que se debe mostrar es la siguiente:

1. Código de la parte
2. Nombre de la parte
3. Año de fabricación
4. Descripción

La tabla debe mostrar las filas impares de color gris claro y la pares de color gris oscuro.

Cuando el usuario posicione el cursor sobre el componente debe cambiar el color de fondo de la fila a negro y La letra debe ser blanca.

Taller 2

Los hijos 1, 3, 5 y 7 son seleccionados.

Span 1 Span 2 Span 3 Span 4 Span 5 Span 6 Span 7

Los hijos 1, 5 y 7 son seleccionados.

3 se usa en el conteo porque es un hijo, pero no se selecciona porque no es un ``.

Span1 Span2 Este es un 'em' Span4 Span5 Span6 Span7 Span8

Los hijos 1, 4, 6 y 8 son seleccionados.

3 no se usa en el conteo ni se selecciona porque es un ``, no un ``, y `nth-of-type` solo selecciona hijos de ese tipo. El `` se omite por completo y se ignora.

Span1 Span2 Este es un 'em' Span4 Span5 Span6 Span7 Span8

Pseudoelementos

Introducción a los Pseudoelementos en CSS

Los **pseudoelementos** en CSS son elementos especiales que permiten aplicar estilos a partes específicas de un elemento sin modificar el HTML. Son útiles para agregar contenido decorativo, personalizar marcadores, o dar estilo a la primera línea o letra de un texto, entre otros. Los pseudoelementos se identifican usando una doble colonia (`:::`) en CSS3, aunque muchos navegadores también soportan la notación de una sola colonia (`:`) por razones de compatibilidad.

Tabla de Pseudoelementos en CSS

Pseudoelemento	Descripción	Ejemplo de Uso
<code>::before</code>	Inserta contenido antes de un elemento. Ideal para iconos o decoraciones adicionales.	<code>button::before { content: "🔥"; }</code>

Pseudoelemento	Descripción	Ejemplo de Uso
<code>::after</code>	Inserta contenido después de un elemento, como un decorativo o un icono de flecha.	<code>button::after { content: "→"; }</code>
<code>::first-line</code>	Aplica estilos solo a la primera línea del texto dentro de un bloque, útil para textos largos.	<code>p::first-line { font-weight: bold; color: #4CAF50; }</code>
<code>::first-letter</code>	Estiliza solo la primera letra de un bloque de texto, común en libros y revistas.	<code>p::first-letter { font-size: 2em; font-weight: bold; color: #FF5733; }</code>
<code>::placeholder</code>	Aplica estilos al texto de marcador de posición (<code>placeholder</code>) en campos de entrada.	<code>input::placeholder { color: #999; font-style: italic; }</code>
<code>::selection</code>	Aplica estilos a la parte del texto seleccionada por el usuario con el cursor.	<code>::selection { background-color: #4CAF50; color: white; }</code>
<code>::marker</code>	Estiliza los marcadores de listas en elementos <code></code> y <code></code> .	<code>li::marker { color: #FF5733; font-size: 1.2em; }</code>
<code>::backdrop</code>	Aplica estilos al fondo de elementos de diálogo (<code><dialog></code>) en pantallas modales.	<code>dialog::backdrop { background-color: rgba(0, 0, 0, 0.7); }</code>
<code>::cue</code>	Personaliza las señales visuales o subtítulos en contenido multimedia.	<code>::cue { color: blue; }</code>
<code>::file-selector-button</code>	Estiliza el botón de selección de archivo en campos <code><input type="file"></code> .	<code>input[type="file"]::file-selector-button { background: #4CAF50; color: white; }</code>

Combinadores en CSS

Los **combinadores** en CSS son símbolos o palabras clave que permiten seleccionar elementos en función de su relación con otros elementos dentro del documento HTML. Existen varios tipos de combinadores que se utilizan para definir relaciones específicas entre elementos en la estructura del DOM, lo que te permite aplicar estilos más precisos y específicos.

Tipos de Combinadores en CSS

Combinador Descendiente ()

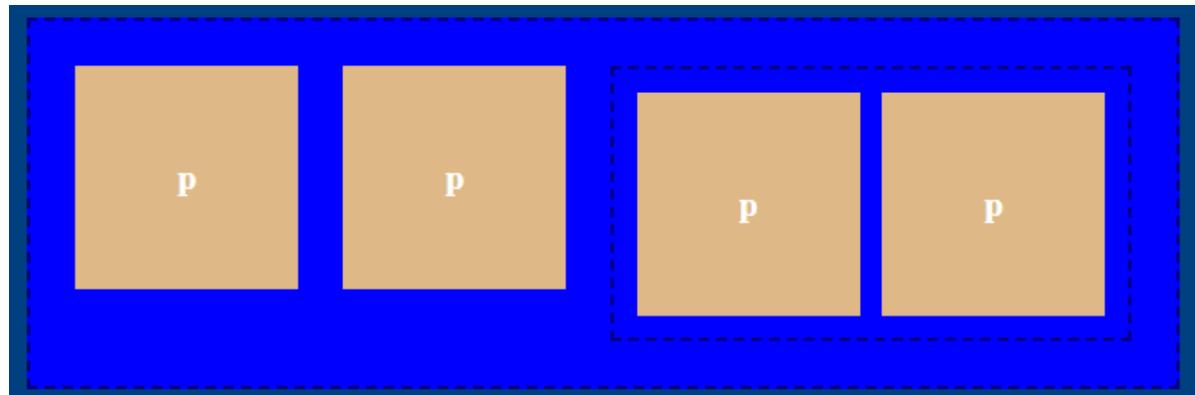
El combinador descendiente (espacio en blanco entre dos selectores) selecciona los elementos que están dentro de otro elemento, sin importar cuántos niveles de profundidad haya.

```
/* Selecciona todos los elementos `<p>` dentro de un elemento con clase
` .container` */
.container p {
    color: blue;
}
```

```
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    margin: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #004080;
}

#capa {
    background-color: blue;
    padding: 20px;
    display: flex;
    gap: 20px;
    border: 2px dashed #000080;
}
#capa p{
    background-color: burlywood;
}
```

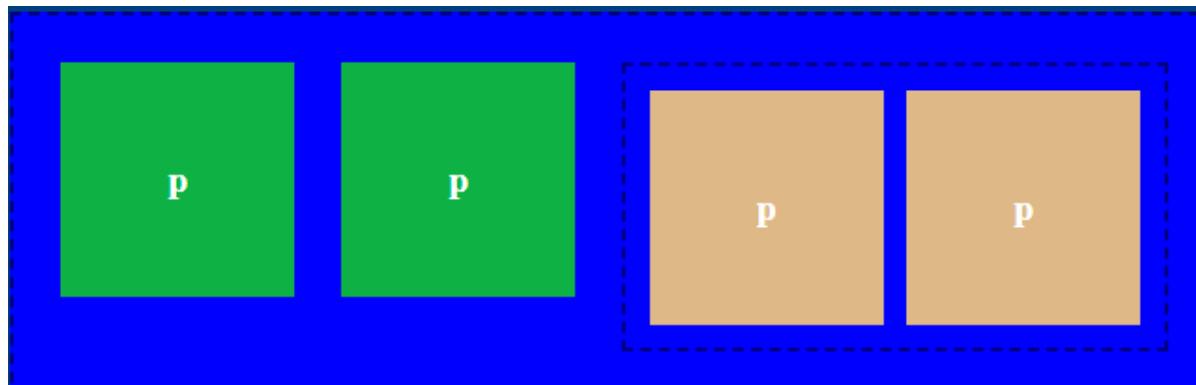


Combinador Hijo (>)

El combinador hijo selecciona los elementos que son **hijos directos** de otro elemento, es decir, que están un nivel directamente dentro de otro.

```
/* Selecciona solo los elementos `<li>` que son hijos directos de un `<ul>` con
clase `.menu` */
.menu > li {
    list-style-type: none;
}
```

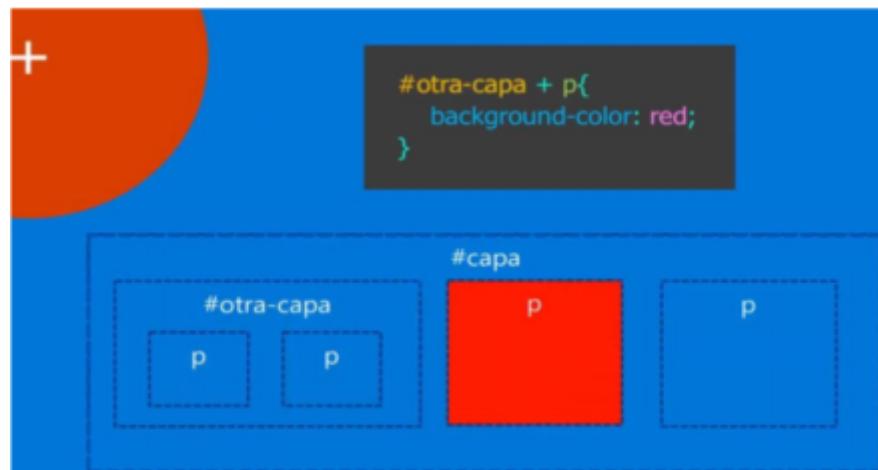
```
#capa > p{
    background-color: rgb(13, 177, 68);
}
```



Combinador de Adyacencia (+)

El combinador adyacente selecciona el **primer elemento hermano** que viene inmediatamente después de otro elemento.

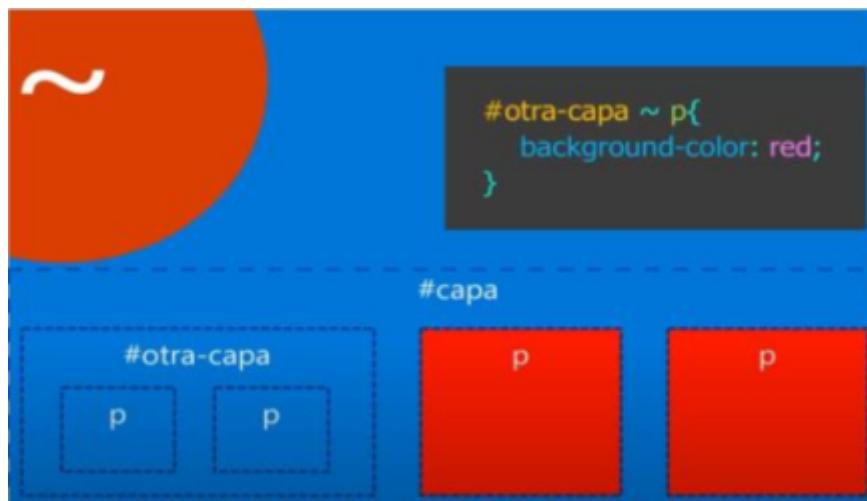
```
/* Selecciona el primer elemento `<p>` que sigue a cualquier elemento `<h2>` */
h2 + p {
    margin-top: 0;
}
```



Combinador de Hermanos Generales (~)

El combinador de hermanos generales selecciona todos los elementos hermanos que vienen después de un elemento específico, sin importar si están directamente después o si hay otros elementos entre ellos.

```
/* Selecciona todos los elementos `<p>` que son hermanos de un `<h2>` y vienen
después de él */
h2 ~ p {
  color: gray;
}
```



Variables

Las variables en CSS, conocidas como *Custom Properties*, permiten almacenar valores reutilizables en el código CSS, facilitando el mantenimiento y la personalización de estilos. Para declarar una variable, se utiliza la sintaxis `--nombre-variable`, y para usarla, la función `var()`.

```
:root {
  --primary-color: #3498db;
  --secondary-color: #2ecc71;
  --font-size: 16px;
}
```

Uso de variables:

Para aplicarlas en propiedades CSS, se usa `var()` seguido del nombre de la variable.

```
body {  
    font-size: var(--font-size);  
    color: var(--primary-color);  
}  
  
.button {  
    background-color: var(--secondary-color);  
    color: #fff;  
    padding: 10px 20px;  
}
```

Ejercicio

Cree un nuevo proyecto y agregue el siguiente código:

```
1  <!DOCTYPE html>  
2  <html lang="en">  
3  <head>  
4      <meta charset="UTF-8">  
5      <meta name="viewport" content="width=device-width, initial-scale=1.  
0">  
6      <link rel="stylesheet" href="css/variables.css">  
7      <title>Document</title>  
8  </head>  
9  <body>  
10     <article class="card">  
11         <h2 class="card__title">Soy una tarjeta</h2>  
12         <div class="container-text">  
13             <p class="card__copy">  
14                 Lorem ipsum dolor, sit amet consectetur adipisicing  
elit. In veritatis commodi quaerat minus, repellat  
voluptatem obcaecati unde alias dolore voluptate! Quia  
autem eligendi eius asperiores libero voluptates dolore  
reprehenderit voluptatem.  
15             </p>  
16             <a href="#" class="card__btn">Ver mas</a>  
17         </div>  
18     </article>  
19  </body>  
20  </html>
```

Responsive Design

Responsive Design o **Diseño Responsive** es una técnica de diseño web que permite que las páginas web se adapten automáticamente a diferentes tamaños de pantalla y dispositivos, como teléfonos móviles, tablets, laptops y monitores de escritorio. El objetivo es ofrecer una experiencia de usuario óptima en cualquier dispositivo, asegurando que el contenido se vea bien y sea fácil de usar, independientemente del tamaño o resolución de la pantalla.

Puntos de ruptura (breakpoints)



Responsive Design



¿Cómo Funciona el Responsive Design?

El diseño responsive se basa en tres componentes principales:

1. Layouts Flexibles (Grid y Flexbox)

Usando unidades relativas como porcentajes, `em` o `rem`, en lugar de unidades fijas como `px`, es posible ajustar el tamaño y la posición de los elementos en función del tamaño de la pantalla. CSS Grid y Flexbox son herramientas poderosas que permiten construir layouts adaptables.

2. Media Queries

Las media queries en CSS permiten aplicar diferentes estilos en función de las características del dispositivo, como el ancho de la pantalla. Esto permite definir breakpoints (puntos de corte) para que el diseño cambie a diferentes tamaños.

Viewport <https://whatismyviewport.com/>

El **viewport** es el área visible de una página web en un dispositivo, como la pantalla de un teléfono móvil, tablet, laptop o monitor de escritorio. En otras palabras, es la "ventana" a través de la cual el usuario ve el contenido de una página web.

Importancia del Viewport en Diseño Web

El viewport es fundamental en **Responsive Design** porque define el espacio disponible en el que se mostrará la página. Para asegurar que el contenido se ajuste correctamente a diferentes tamaños de pantalla, usamos el viewport para controlar y adaptar los elementos de la página de manera dinámica.

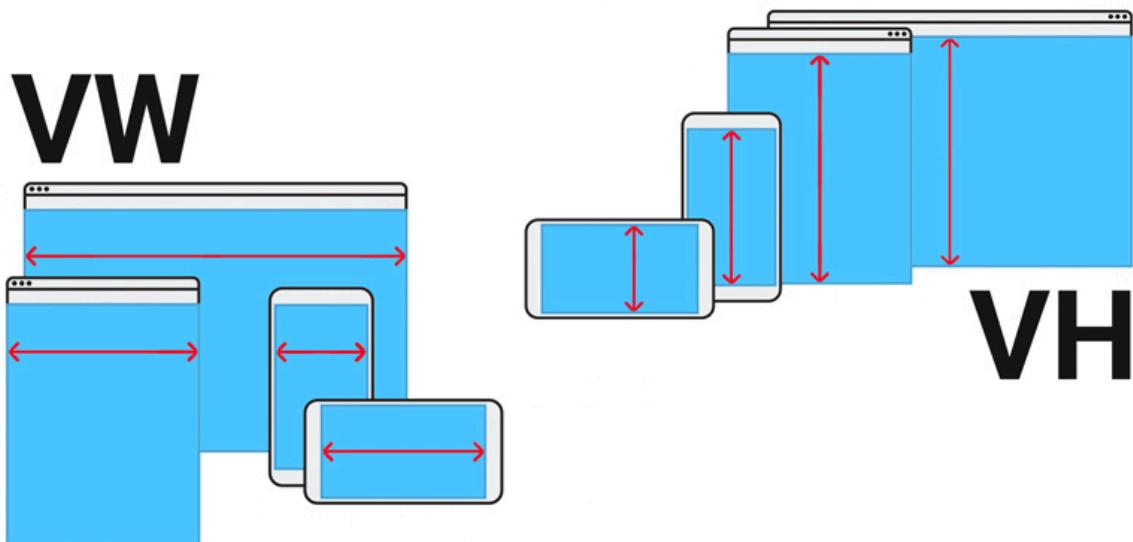
Meta Viewport en HTML

Para que el diseño de una página sea realmente adaptable, especialmente en dispositivos móviles, se utiliza la etiqueta `<meta viewport>` en el HTML. Esta etiqueta permite configurar cómo el navegador debe ajustar el contenido en función del ancho de la pantalla.

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

```
<meta name="viewport" content="width=device-width, maximum-scale=1">
```

```
<meta name="viewport" content="width=device-width, user-scalable=no">
```



Imagenes responsivas

Las **imágenes responsivas** son imágenes que se adaptan al tamaño de la pantalla y al dispositivo en el que se están viendo, para optimizar tanto la apariencia como el rendimiento. Esto es esencial para crear una buena experiencia de usuario en sitios de diseño responsive.

Técnicas para Crear Imágenes Responsivas

CSS con `max-width` y `height: auto`

Usar CSS para hacer que las imágenes se adapten al ancho del contenedor sin excederlo:

```
img {  
    max-width: 100%;  
    height: auto;  
}
```

Atributo `srcset` en HTML

El atributo `srcset` permite definir múltiples versiones de una imagen para diferentes tamaños de pantalla o resoluciones, de modo que el navegador seleccione la mejor opción según el tamaño del viewport y la densidad de píxeles del dispositivo.

```

```

`srcset`: Lista de imágenes y el ancho para el cual están optimizadas.

`sizes` : Define qué tamaño debe usar el navegador según el ancho del viewport. En el ejemplo:

- Para pantallas menores de 600px, la imagen tomará el 100% del ancho del viewport.
- Para pantallas mayores, tomará el 50% del ancho del viewport.

márgenes de fondo con `background-size: cover o contain`

Cuando se usan imágenes de fondo en CSS, `background-size` permite ajustar cómo se muestra la imagen dentro del contenedor.

```
.hero {  
    background-image: url('imagen-grande.jpg');  
    background-size: cover;  
    background-position: center;  
}
```

`cover` : Hace que la imagen cubra todo el contenedor, aunque se recorte en algunos lados.

`contain` : Escala la imagen para que esté completamente visible en el contenedor, aunque pueda dejar espacio vacío.

Formato de imagen adaptable (como SVG)

Los **SVGs** son gráficos vectoriales escalables y se adaptan a cualquier tamaño sin perder calidad, lo cual es ideal para logotipos, iconos y gráficos. Al usar SVG, la imagen puede ajustarse automáticamente al contenedor sin necesidad de varios archivos de diferentes tamaños.

```

```

Uso de `picture` para diferentes versiones de imagen

La etiqueta `<picture>` permite definir múltiples fuentes de imagen según condiciones como el tamaño de la pantalla o el tipo de dispositivo.

```
<picture>  
    <source srcset="imagen-mobile.jpg" media="(max-width: 600px)">  
    <source srcset="imagen-tablet.jpg" media="(max-width: 1024px)">  
      
</picture>
```

- Si el viewport es menor de 600px, se carga `imagen-mobile.jpg`.
- Si está entre 600px y 1024px, se carga `imagen-tablet.jpg`.
- Si es mayor a 1024px, se carga `imagen-desktop.jpg`.

Ventajas de las Imágenes Responsivas

- **Optimización de rendimiento:** Carga versiones más ligeras en dispositivos móviles, mejorando la velocidad de carga.
- **Ahorro de ancho de banda:** Evita cargar imágenes innecesariamente grandes en pantallas pequeñas.

- **Mejor experiencia de usuario:** Asegura que las imágenes se adapten adecuadamente a cualquier pantalla.

MediaQueries

Los **media queries** son una característica de CSS que permite aplicar estilos específicos a una página web en función de las características del dispositivo o el entorno en el que se visualiza, como el tamaño de la pantalla, la resolución, la orientación, entre otros. Son fundamentales en el **diseño responsive** porque ayudan a adaptar el diseño de una página a distintos dispositivos, como teléfonos, tablets, laptops y monitores de escritorio.

¿Cómo Funcionan los Media Queries?

Las media queries se definen utilizando la palabra clave `@media` seguida de una o más condiciones, y dentro de ellas se especifican los estilos que se aplicarán cuando se cumplan esas condiciones.

```
/* Estilos generales aplicables a todos los dispositivos */
body {
    font-size: 16px;
    background-color: white;
}

/* Estilos para pantallas de hasta 768px de ancho (tablets y móviles) */
@media (max-width: 768px) {
    body {
        font-size: 14px;
        background-color: lightgrey;
    }
}

/* Estilos para pantallas mayores a 1024px de ancho (escritorios) */
@media (min-width: 1024px) {
    body {
        font-size: 18px;
        background-color: lightblue;
    }
}
```

En este ejemplo:

- Para pantallas de hasta 768px de ancho, el fondo será gris claro y el tamaño de fuente será de 14px.
- Para pantallas mayores a 1024px, el fondo será azul claro y el tamaño de fuente será de 18px.

Usos Comunes de Media Queries

Diseño adaptable a diferentes tamaños de pantalla

Permiten cambiar el diseño de una página según el ancho de la pantalla. Por ejemplo, mostrar una sola columna en pantallas pequeñas y múltiples columnas en pantallas grandes.

```
@media (max-width: 600px) {  
    .container {  
        display: block;  
    }  
}  
  
@media (min-width: 601px) {  
    .container {  
        display: flex;  
    }  
}
```

Orientación de pantalla

Se pueden aplicar estilos diferentes según si el dispositivo está en **modo retrato** o **modo paisaje**.

```
@media (orientation: landscape) {  
    body {  
        font-size: 18px;  
    }  
}
```

Resolución de pantalla

Permiten ajustar el diseño según la resolución de pantalla, útil para dispositivos con alta densidad de píxeles (como pantallas Retina).

```
@media (min-resolution: 192dpi) {  
    img {  
        width: 50%;  
    }  
}
```

Tipos de dispositivo

También es posible especificar estilos solo para ciertos tipos de dispositivos, aunque hoy en día esto se usa menos ya que el enfoque se centra más en el tamaño de la pantalla.

```
@media screen and (max-width: 600px) {  
    /* Estilos específicos para pantallas de hasta 600px */  
}  
  
@media print {  
    /* Estilos específicos para impresión */  
}
```

Componentes de un Media Query

- **Tipos de medios** (`screen`, `print`, `a11`): Especifican el tipo de dispositivo o contexto donde se aplican los estilos.
- **Características**: Atributos como `width`, `height`, `orientation`, `resolution`.
- **Operadores**: Permiten combinar condiciones con palabras clave como `and`, `not`, y `only`.

```
@media only screen and (min-width: 768px) and (orientation: portrait) {  
    .container {  
        font-size: 20px;  
        padding: 10px;  
    }  
}
```

Tipos de medio

Los tipos de medios se refieren a la categoría general de un dispositivo específico. Aunque normalmente los sitios web se desarrollan pensando en pantallas, puede ser útil crear estilos adaptados a dispositivos específicos, como impresoras o lectores de pantalla de audio.

```
@media print {  
    /* ... */  
}
```

También puedes considerar múltiples dispositivos. Por ejemplo, esta regla `@media` usa dos consultas de medios tanto para dispositivos de pantalla como de impresión:

```
@media screen, print {  
    /* ... */  
}
```

[a11](#)

Aplicable a todos los dispositivos.

[print](#)

Destinado a material paginado y documentos visibles en una pantalla en modo de vista previa para impresión. Por favor, consulte la sección de [medios paginados](#), y la [sección de Media en el tutorial de Introducción](#) para más información acerca de problemas de formateo específicos para los medios paginados.

[screen](#)

Destinado a principalmente a pantallas de computadora a color.

Características de Medios (media feature)

Cada *característica de medios* verifica una característica específica del navegador o dispositivo.

Nombre	Resumen	Notas
width	Anchura del viewport	
height	Altura del viewport	
aspect-ratio	Relación de aspecto anchura-altura del viewport	
orientation	Orientación del viewport	
resolution	Densidad de pixeles del dispositivo	
scan	Proceso de escaneo del dispositivo	
grid	¿El dispositivo es un grid o un mapa de bits?	
update-frequency	Qué tan rápido (si lo hace) puede el dispositivo modificar la apariencia del contenido	Incluido en Media Queries Nivel 4
overflow-block	Cómo maneja el dispositivo el contenido que excede los límites del viewport a lo largo del eje de bloque	Incluido en Media Queries Nivel 4
overflow-inline	¿Puede desplazarse hacia el contenido que excede los límites del viewport?	Incluido en Media Queries Nivel 4
color	Componente de número de bits por color del dispositivo, o cero si el dispositivo no es a color.	
color-index	Número de entradas en la tabla de búsqueda de color del dispositivo, o cero si el dispositivo no usa una tabla.	
display-mode	Modo de visualización de la aplicación, según se especifique en la propiedad display del manifiesto de la aplicación web.	Definido en la especificación del Manifiesto de Aplicación Web .
monochrome	Bits por pixel en el buffer de marco monocromático del dispositivo, o 0 si el dispositivo no es monocromático.	
inverted-colors	¿El agente usuario o el Sistema Operativo está invirtiendo los colores?	Incluido en Media Queries Nivel 4

Nombre	Resumen	Notas
pointer	¿El mecanismo de entrada principal es un dispositivo apuntador? y de ser así, ¿qué tan preciso es?	Incluido en Media Queries Nivel 4
hover	¿El mecanismo de entrada principal permite que el usuario posicione el puntero sobre los elementos?	Incluido en Media Queries Nivel 4
any-pointer	¿Hay algún mecanismo de entrada que sea dispositivo apuntador? y de ser así, ¿qué tan preciso es éste?	Incluido en Media Queries Nivel 4
any-hover	¿Algún mecanismo de entrada disponible permite que el usuario posicione el puntero sobre los elementos?	Incluido en Media Queries Nivel 4
light-level	Nivel de luz ambiental actual	Incluido en Media Queries Nivel 4
scripting	¿Se soporta lenguaje de script (p.ej. JavaScript)?	Incluido en Media Queries Nivel 4
device-width Obsoleto	Anchura de la superficie de representación del dispositivo	Descontinuado en Media Queries Nivel 4
device-height Obsoleto	Altura de la superficie de representación del dispositivo	Descontinuado en Media Queries Nivel 4
device-aspect-ratio Obsoleto	Relación de aspecto anchura-altura del dispositivo	Descontinuado en Media Queries Nivel 4
-webkit-device-pixel-ratio Non-standard	Número de pixeles reales del dispositivo por pixel CSS	No estándar; Específica de WebKit/Blink. De ser posible, use la característica resolution en su lugar.
-webkit-transform-3d Non-standard	¿Se soportan transformaciones 3D?	No estándar; Específica de WebKit/Blink
-webkit-transform-2d Non-standard	¿Se soportan transformaciones 2D?	No estándar; Específica de WebKit
-webkit-transition Non-standard	Se soportan transiciones CSS?	No estándar; Específica de WebKit
-webkit-animation Non-standard	¿Se soportan animaciones CSS?	No estándar; Específica de WebKit

```

@media print {
  body {
    font-size: 10pt;
  }
}
@media screen {
  body {
    font-size: 13px;
  }
}
@media screen, print {
  body {
    line-height: 1.2;
  }
}
@media only screen and (min-device-width: 320px) and (max-device-width: 480px)
and (-webkit-min-device-pixel-ratio: 2) {
  body {
    line-height: 1.4;
  }
}

```

@media (hover: hover)

La media query `@media (hover: hover)` es una característica avanzada en CSS que detecta si el dispositivo del usuario **admite el uso de la acción de "hover"** (colocar el cursor sobre un elemento) o no. Esto es especialmente útil para adaptar la interfaz en dispositivos táctiles, como teléfonos móviles o tablets, que no tienen la capacidad de "hover" en el mismo sentido que un mouse en una computadora de escritorio.

¿Por Qué Usar `@media (hover: hover)`?

En dispositivos que admiten el hover (como computadoras con ratón o ciertos tablets con lápiz), puedes aplicar efectos o cambios visuales al pasar el cursor sobre un elemento. Sin embargo, en dispositivos táctiles que no admiten hover, estos estilos son innecesarios y podrían interferir con la usabilidad.

```

/* Estilos generales para todos los dispositivos */
.button {
  background-color: #4CAF50;
  color: white;
  padding: 10px 20px;
  border: none;
  cursor: pointer;
}

/* Solo aplica el efecto hover en dispositivos que soportan hover */
@media (hover: hover) {
  .button:hover {
    background-color: #45a049;
  }
}

```

Todos los dispositivos verán el botón con el color de fondo verde (#4CAF50) y el texto blanco.

Sin embargo, **solo los dispositivos que admiten hover** (como las computadoras con mouse) verán el cambio de color de fondo cuando el usuario coloque el cursor sobre el botón.

Otras Opciones con `hover`

Hay algunas variaciones que se pueden usar en media queries relacionadas con `hover`:

- `@media (hover: none)`: Se aplica en dispositivos que no soportan hover, como la mayoría de los dispositivos móviles.

```
@media (hover: none) {  
    /* Ajusta los estilos para pantallas táctiles */  
    .button {  
        padding: 12px 24px;  
    }  
}
```

`@media (max-width: 1250px)`

La media query `@media (max-width: 1250px)` se utiliza en CSS para aplicar estilos específicos cuando el ancho de la pantalla es **igual o menor a 1250 píxeles**. Esto es común en **diseño responsive** para ajustar el diseño de una página en función del tamaño de la pantalla, permitiendo una experiencia de usuario óptima en dispositivos como tablets, laptops y pantallas de escritorio más pequeñas.

```
/* Estilos generales (para pantallas mayores a 1250px) */  
.navbar {  
    display: flex;  
    flex-direction: row;  
    font-size: 18px;  
}  
  
.container {  
    max-width: 1200px;  
    margin: 0 auto;  
}  
  
/* Estilos para pantallas de 1250px o menos */  
@media (max-width: 1250px) {  
    .navbar {  
        flex-direction: column; /* Cambia a barra de navegación vertical */  
    }  
  
    .container {  
        padding: 0 20px; /* Reduce el padding para pantallas más pequeñas */  
    }  
  
    body {  
        font-size: 16px; /* Disminuye ligeramente el tamaño de la fuente */  
    }  
}
```

```
}
```

`.navbar`: Cambia la dirección de la barra de navegación a una columna en pantallas de hasta 1250px de ancho, ideal para ahorrar espacio en dispositivos más pequeños.

`.container`: Ajusta el padding para que el contenido tenga más espacio en los bordes y se vea menos comprimido en pantallas más pequeñas.

`body`: Disminuye el tamaño de fuente para que el texto sea más legible en pantallas más pequeñas.

¿Cuándo Usar `max-width` en Media Queries?

Usar `max-width` es ideal cuando se busca que un diseño específico se aplique en tamaños de pantalla más pequeños o iguales a un límite definido. Algunos ejemplos comunes incluyen:

- **1250px o menos**: Ajustes para laptops más pequeñas.
- **1024px o menos**: Ajustes para tablets en posición horizontal.
- **768px o menos**: Ajustes para dispositivos móviles y tablets en posición vertical.

@media (color)

La media query `@media (color)` en CSS se utiliza para verificar si el dispositivo en uso **admite colores** en su pantalla. Esto puede ser útil cuando se desea aplicar estilos específicos en dispositivos con capacidades de color, diferenciándolos de aquellos que solo admiten blanco y negro o escala de grises (aunque estos dispositivos son cada vez menos comunes).

```
/* Estilos básicos aplicables a todos los dispositivos */
body {
    font-family: Arial, sans-serif;
    background-color: white;
    color: black;
}

/* Solo aplica si el dispositivo admite color */
@media (color) {
    body {
        background-color: #f0f8ff;
        color: #333;
    }

    h1 {
        color: #4CAF50;
    }

    a {
        color: #FF5733;
    }
}
```

¿Cuándo Usar @media (color)?

Dado que la mayoría de los dispositivos modernos admiten color, esta media query no es muy común hoy en día. Sin embargo, puede ser útil en **entornos específicos** donde el soporte de color no esté garantizado, como algunos dispositivos e-ink (pantallas de tinta electrónica) o impresoras en blanco y negro.

Creación de *media queries* complejas

A veces, es posible que necesite crear una media query que dependa de múltiples condiciones. En estos casos, se utilizan operadores lógicos como *not*, *and* y *only*. También es posible combinar varias media queries en una lista separada por comas, lo que permite aplicar los mismos estilos en distintas circunstancias.

Combinación de múltiples tipos o características

La media query `@media (min-width: 30em) and (orientation: landscape)` se utiliza para aplicar estilos específicos cuando el ancho mínimo de la pantalla es de **30em** (o aproximadamente 480 píxeles en pantallas estándar) y la **orientación** del dispositivo es **horizontal (landscape)**. Esto es útil para ajustar el diseño en dispositivos que se giran a modo horizontal, como tablets y teléfonos móviles.

```
/* Estilos generales para todos los dispositivos y orientaciones */
.container {
    display: flex;
    flex-direction: column;
    padding: 10px;
    font-size: 16px;
}

/* Estilos para dispositivos en orientación horizontal con un ancho mínimo de 30em */
@media (min-width: 30em) and (orientation: landscape) {
    .container {
        flex-direction: row; /* Cambia la dirección de los elementos a horizontal */
    }
    padding: 20px;
}

.sidebar {
    width: 30%; /* Aumenta el ancho de la barra lateral */
}

.content {
    width: 70%; /* Aumenta el ancho de la sección de contenido */
}

.footer {
    font-size: 14px; /* Ajusta el tamaño de fuente en modo horizontal */
}
```

Explicación

- `.container`: Cambia la dirección de los elementos dentro del contenedor a fila (horizontal) cuando el dispositivo se gira a modo landscape y tiene un ancho mínimo de 30em.
- `.sidebar` y `.content`: Ajustan sus anchos para aprovechar el espacio horizontal adicional.
- `.footer`: Ajusta el tamaño de la fuente para una mejor legibilidad en orientación horizontal.

Limitar los estilos a los dispositivos con una pantalla

La media query `@media screen and (min-width: 30em) and (orientation: landscape)` se utiliza para aplicar estilos únicamente en **pantallas** (excluyendo otros medios como impresión), donde el ancho mínimo es de **30em** (aproximadamente 480 píxeles en pantallas estándar) y la orientación del dispositivo es **horizontal (landscape)**. Este enfoque permite adaptar el diseño específicamente en dispositivos como tablets o teléfonos móviles que se giran a modo horizontal y cumplen con el ancho mínimo.

```
/* Estilos generales aplicables en todas las orientaciones y tamaños */
.header {
    font-size: 16px;
    padding: 10px;
}

.nav {
    display: none;
}

.main-content {
    padding: 15px;
}

/* Estilos para pantallas con ancho mínimo de 30em y orientación horizontal */
@media screen and (min-width: 30em) and (orientation: landscape) {
    .header {
        font-size: 18px; /* Aumenta el tamaño de fuente de la cabecera */
    }

    .nav {
        display: block; /* Muestra el menú de navegación en modo horizontal */
    }

    .main-content {
        padding: 20px; /* Aumenta el padding de la sección de contenido */
        display: flex;
        flex-direction: row; /* Cambia el diseño a una disposición en fila */
    }

    .sidebar {
        width: 30%; /* Define el ancho de la barra lateral */
    }

    .content {
        width: 70%; /* Define el ancho de la sección de contenido */
    }
}
```

```
}
```

Explicación

- `.header`: Aumenta el tamaño de la fuente para mejor visibilidad en modo horizontal.
- `.nav`: Muestra el menú de navegación solo cuando el dispositivo está en landscape y tiene al menos 30em de ancho.
- `.main-content`: Cambia el diseño a disposición en fila para aprovechar el ancho horizontal, con una barra lateral (`.sidebar`) y una sección de contenido (`.content`) que se distribuyen el espacio.

¿Cuándo Usar `screen and (min-width: 30em)` and `(orientation: landscape)`?

Esta media query es útil en **diseño responsive** para optimizar el diseño específicamente en pantallas y dispositivos móviles que se giran en orientación horizontal, mejorando la usabilidad y aprovechando el espacio extra.

Múltiples consultas

La media query `@media (min-height: 680px), screen and (orientation: portrait)` se usa para aplicar estilos en dos casos específicos:

1. **Cuando la altura de la pantalla es de al menos 680 píxeles**, independientemente de otros factores, lo cual es útil para dispositivos con pantallas altas, como ciertos teléfonos móviles o monitores en orientación vertical.
2. **Cuando la pantalla está en orientación vertical (portrait)**, sin importar el tamaño de la pantalla, aplicándose a todos los dispositivos en modo retrato.

La coma (,) entre las condiciones actúa como un **operador "OR"**, lo que significa que si se cumple cualquiera de las dos condiciones, los estilos dentro de la media query se aplicarán.

```
/* Estilos generales aplicables en todas las orientaciones y tamaños */
.header {
    font-size: 16px;
    padding: 10px;
}

.main-content {
    display: flex;
    flex-direction: row;
    padding: 20px;
}

/* Estilos específicos para pantallas de altura mínima de 680px o en orientación vertical */
@media (min-height: 680px), screen and (orientation: portrait) {
    .header {
        font-size: 20px; /* Aumenta el tamaño de la fuente en pantallas altas o en vertical */
        padding: 15px;
    }
}
```

```
.main-content {  
    flex-direction: column; /* Cambia a disposición en columna en pantallas  
    verticales */  
}  
  
.sidebar {  
    display: none; /* Oculta la barra lateral para pantallas verticales o con  
    suficiente altura */  
}  
}
```

Explicación

- `.header`: Aumenta el tamaño de la fuente y el padding cuando la altura de la pantalla es al menos 680px o cuando el dispositivo está en orientación vertical.
- `.main-content`: Cambia la dirección de los elementos a columna en dispositivos en orientación vertical o en pantallas altas, facilitando la visualización en dispositivos móviles.
- `.sidebar`: Oculta la barra lateral en estas condiciones para ahorrar espacio y mejorar la usabilidad.

¿Cuándo Usar Esta Media Query?

Esta combinación es útil en **diseño responsive** para:

- Asegurar que el contenido sea más legible en dispositivos con pantallas altas o en orientación vertical.
- Reorganizar el layout para mejorar la experiencia en pantallas donde la altura es predominante o en dispositivos móviles en modo retrato.
- Optimizar la interfaz para pantallas móviles, tablets y pantallas más grandes en orientación vertical, proporcionando una experiencia de usuario coherente en distintas condiciones de visualización.

Invertir el significado de una consulta

La palabra clave `not` invierte el significado de una *media query* completa. Solo negará la *media query* específica a la que se aplica. (Por lo tanto, no se aplicará a todas las *media queries* en una lista de *media queries* separadas por comas).

La media query `@media not all and (monochrome)` se usa para aplicar estilos únicamente en dispositivos que **no están en modo monocromático** (es decir, que no muestran solo en blanco y negro o en escala de grises). Esto significa que los estilos dentro de esta media query solo se aplicarán a dispositivos que admiten color.

¿Cómo Funciona `not all and (monochrome)`?

- `not`: Invierte el resultado de la consulta. En este caso, `not` asegura que la media query se aplique a dispositivos que no son monocromáticos.
- `all`: Indica que la media query puede aplicarse a todos los tipos de dispositivos (pantallas, impresoras, etc.), pero aquí se combina con `not`.

- `(monochrome)` : Verifica si el dispositivo muestra contenido en escala de grises. Usado en combinación con `not`, esta consulta selecciona solo dispositivos que admiten color.

```

/* Estilos generales para todos los dispositivos */
body {
    font-family: Arial, sans-serif;
    font-size: 16px;
}

/* Estilos aplicables solo en dispositivos que no son monocromáticos */
@media not all and (monochrome) {
    body {
        background-color: #f0f8ff;
        color: #333;
    }

    h1 {
        color: #4CAF50;
    }

    a {
        color: #FF5733;
    }
}

```

¿Cuándo Usar `not all and (monochrome)`?

Esta media query es útil cuando se desea **evitar estilos de color** en dispositivos que solo admiten monocromo, como algunas **impresoras en blanco y negro** o ciertos dispositivos con pantallas de tinta electrónica. En estos casos:

- **Evita aplicar colores innecesarios** que no se verán en dispositivos monocromáticos.
- **Mejora la legibilidad** al no forzar colores que podrían no mostrarse correctamente.

Media queries Nivel 4

En el **Nivel 4 de Media Queries (Media Queries Level 4)** de CSS, la especificación introduce varias mejoras y ampliaciones para crear consultas de medios más potentes y precisas. Aquí están las principales novedades en la sintaxis y características:

Nuevos Ranges (Rangos de Comparación)

En el Nivel 4, se introdujo una nueva sintaxis para hacer comparaciones con operadores de rango, que permite simplificar y hacer más legibles las consultas. Ahora puedes escribir comparaciones como `<`, `<=`, `>`, y `>=` para los valores de media queries.

```

/* Ancho mínimo de 600px */
@media (width >= 600px) {
    /* Estilos para pantallas de al menos 600px de ancho */
}

/* Ancho de pantalla entre 600px y 1200px */
@media (600px <= width <= 1200px) {
    /* Estilos para pantallas entre 600px y 1200px de ancho */
}

```

Media Features como Rangos

Además de los nuevos operadores de rango, se pueden escribir expresiones con media features en ambas direcciones, lo cual hace que la lectura sea más flexible.

```

@media (width <= 800px) {
    /* Estilos para pantallas de hasta 800px de ancho */
}

@media (800px <= width) {
    /* Estilos para pantallas de al menos 800px de ancho */
}

```

Uso de Palabras Clave: `or`, `and`, y `not`

En el Nivel 4, las palabras clave (`or`, `and`, `not`) se usan en lugar de comas o palabras específicas, lo que hace la sintaxis más clara y semántica. Estas palabras clave permiten combinar varias consultas de una manera lógica.

```

/* Pantallas de al menos 800px de ancho o en modo landscape */
@media (min-width: 800px) or (orientation: landscape) {
    /* Estilos para pantallas grandes o en orientación horizontal */
}

```

Media Queries de `scripting` y `pointer`

Media Queries Level 4 también introduce algunas nuevas características para detectar la capacidad de los dispositivos, como:

`scripting`: Detecta si el navegador tiene JavaScript habilitado.

```

@media (scripting: enabled) {
    /* Estilos para navegadores con JavaScript habilitado */
}

```

`pointer`: Detecta la precisión del dispositivo de puntero (como ratón o pantalla táctil).

```

@media (pointer: coarse) {
    /* Estilos para dispositivos con punteros de baja precisión, como pantallas táctiles */
}

```

`hover`: Detecta si el dispositivo admite hover.

```
@media (hover: hover) {  
    /* Estilos para dispositivos que permiten hover (como ratones) */  
}
```

Uso de `any-hover` y `any-pointer`

`any-hover`: Detecta si *alguno* de los dispositivos de entrada admite hover (por ejemplo, en un dispositivo híbrido como un laptop con pantalla táctil y mouse).

```
@media (any-hover: hover) {  
    /* Estilos para dispositivos que tienen al menos un dispositivo que permite  
    hover */  
}
```

`any-pointer`: Detecta si *algún* dispositivo de entrada tiene cierta precisión de puntero.

```
@media (any-pointer: fine) {  
    /* Estilos para dispositivos con al menos un dispositivo de entrada de alta  
    precisión */  
}
```

Soporte de Media Queries `update`

El nivel 4 también permite detectar la frecuencia de actualización de la pantalla, como si es continua o intermitente.

- `update`: Define si el dispositivo puede actualizar visualmente de manera continua (`update: fast`), de manera limitada (`update: slow`), o si no se actualiza (`update: none`).

```
@media (update: fast) {  
    /* Estilos para pantallas con actualización rápida, como monitores  
    tradicionales */  
}
```

```
@media (width >= 600px) and (hover: hover) and (pointer: fine) {  
    /* Estilos para pantallas de al menos 600px, que admiten hover y tienen  
    punteros de alta precisión */  
}  
  
@media (600px <= width <= 1200px) or (orientation: landscape) and (update: fast)  
{  
    /* Estilos para pantallas entre 600px y 1200px de ancho, o en modo horizontal  
    con actualización rápida */  
}
```

Diseño Fluido con Unidades Relativas

En lugar de usar unidades fijas como píxeles, se utilizan unidades relativas como `%`, `em`, `rem`, o `vw` y `vh`, que permiten que los elementos se ajusten de forma proporcional al tamaño del contenedor o de la pantalla.

Diseño con Flexbox y Grid

CSS Flexbox y Grid son sistemas de diseño que facilitan la creación de layouts flexibles y adaptativos sin necesidad de calcular tamaños manualmente.

- **Flexbox** es ideal para disposiciones en una dimensión (horizontal o vertical).
- **Grid** permite organizar elementos en ambas dimensiones y es excelente para layouts más complejos.

```
.container {  
    display: flex;  
    flex-direction: row;  
    flex-wrap: wrap; /* Permite que los elementos salten a la siguiente fila si  
no caben */  
}  
  
.grid {  
    display: grid;  
    grid-template-columns: repeat(auto-fill, minmax(200px, 1fr)); /* crea  
columnas de al menos 200px que se expanden automáticamente */  
}
```

Diseño Móvil-Primero (Mobile-First)

En esta técnica, el diseño se estructura primero para dispositivos móviles (pantallas pequeñas) y luego se expande para pantallas más grandes. Este enfoque es más eficiente porque se añaden características y estilos solo para pantallas grandes, reduciendo el peso en dispositivos móviles.

```
/* Estilos base para dispositivos móviles */  
.container {  
    padding: 10px;  
    font-size: 1em;  
}  
  
/* Estilos adicionales para pantallas más grandes */  
@media (min-width: 768px) {  
    .container {  
        padding: 20px;  
        font-size: 1.2em;  
    }  
}
```

Contenedores Flexibles

Usar contenedores con propiedades flexibles permite que el contenido interno se ajuste automáticamente en función de la pantalla disponible.

Uso de Picture Element para Imágenes Adaptables

La etiqueta `<picture>` permite definir múltiples fuentes de imagen que se cargarán dependiendo del tamaño de la pantalla o de la densidad de píxeles.

```
<picture>
  <source srcset="img-grande.jpg" media="(min-width: 800px)">
  <source srcset="img-mediana.jpg" media="(min-width: 400px)">
  
</picture>
```

Componentes Condicionales

Es posible mostrar u ocultar componentes específicos en función del tamaño de la pantalla. Esto optimiza el diseño y mejora la experiencia del usuario en dispositivos pequeños.

```
@media (max-width: 768px) {
  .desktop-only {
    display: none; /* Oculta en pantallas pequeñas */
  }
}

@media (min-width: 769px) {
  .mobile-only {
    display: none; /* Oculta en pantallas grandes */
  }
}
```

URL de Interés

<https://davidwalsh.name/animate-media-queries>

https://developer.mozilla.org/es/docs/Web/CSS/CSSContainment/Container_queries

https://developer.mozilla.org/en-US/docs/Web/CSS/CSSMediaQueries/Testing_media_queries

https://developer.mozilla.org/en-US/docs/Web/CSS/WebKit_Extensions

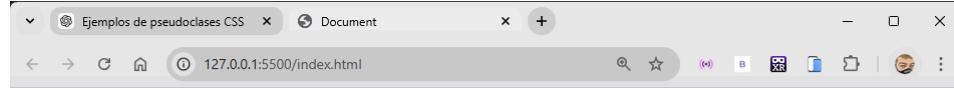
Practica Media Queries

1. Cree estructura basica de proyecto , y archivo index.html
2. En el archivo css ingrese el siguiente codigo

```
*{  
    margin: 0;  
    padding: 0;  
    box-sizing: border-box;  
}  
.title{  
    font-size: 3rem;  
    color: blueviolet;  
    margin: 40px 0;  
    text-align: center;  
    font-family: monospace;  
}
```

3. En el index.html agregue el siguiente codigo

```
<!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="css/styles.css">  
    <title>Document</title>  
</head>  
<body>  
    <h1 class="title">Hey campers</h1>  
</body>  
</html>
```



Hey campers

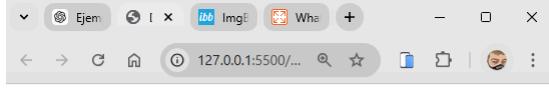
4. Aplicando media queries

```
@media print{  
    .title{  
        color: brown;  
        font-size: 5rem;  
        text-align: right;  
        font-family: cursive;  
    }  
}
```

Hey campers

```
127.0.0.1:5500/index.html 1/1

@media screen and (max-width:600px){
    .title{
        background-color: rgb(0, 0, 0);
        color: white;
    }
}
```



The screenshot shows a browser window with the URL '127.0.0.1:5500/index.html'. The page content consists of a single line of text: 'Hey campers', which is displayed on a solid black background. The browser interface includes standard navigation buttons (back, forward, search, etc.) and a status bar at the bottom.

FlexBox

Flexbox (Flexible Box Layout) es un modelo de diseño en CSS que permite distribuir y alinear elementos dentro de un contenedor de forma eficiente y flexible. Es ideal para crear layouts adaptativos y responsivos, ya que simplifica la alineación de elementos tanto en el eje horizontal como en el vertical, y permite distribuir el espacio entre ellos.

Conceptos Básicos de Flexbox

1. **Contenedor Flex (`flex container`)**: Es el elemento al que se le aplica `display: flex`. Este contenedor actúa como un "contenedor flexible" que gestiona el diseño y alineación de sus elementos hijos (los elementos flexibles).
2. **Elementos Flex (`flex items`)**: Son los elementos hijos directos dentro del contenedor flex. Estos elementos se distribuyen y alinean de acuerdo con las propiedades de Flexbox.

Propiedades del Contenedor Flex

Estas propiedades se aplican al contenedor (el elemento al que le das `display: flex`):

display

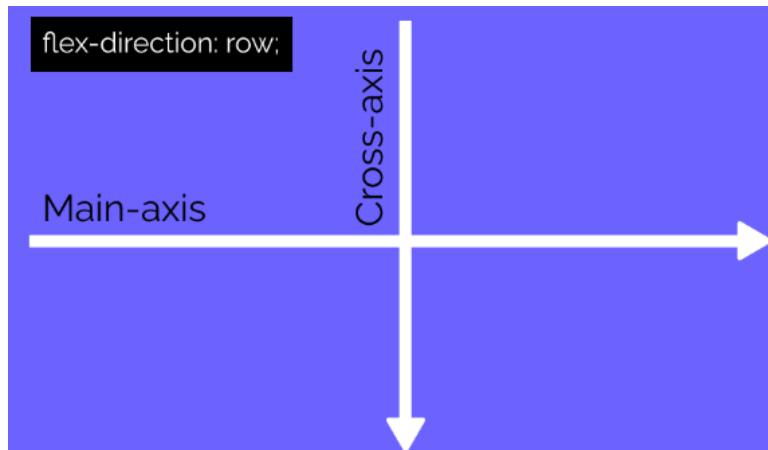
Define el contenedor como un contenedor flex. Puedes usar `flex` para diseño en una sola dimensión (horizontal o vertical) o `inline-flex` si deseas que el contenedor siga siendo un elemento en línea.

```
.container {  
    display: flex;  
}
```

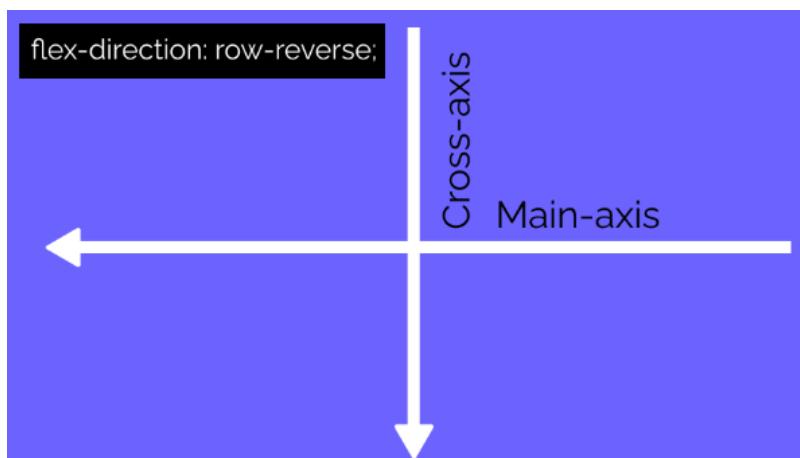
flex-direction

Define la dirección de los elementos dentro del contenedor:

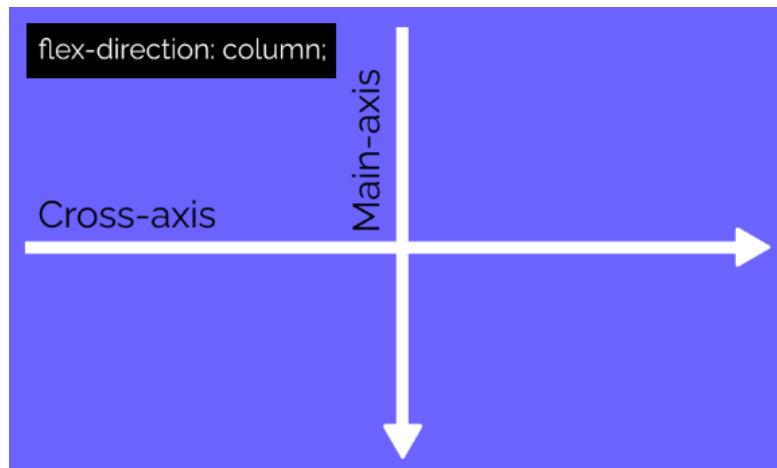
- `row`: Los elementos se alinean en una fila (predeterminado).



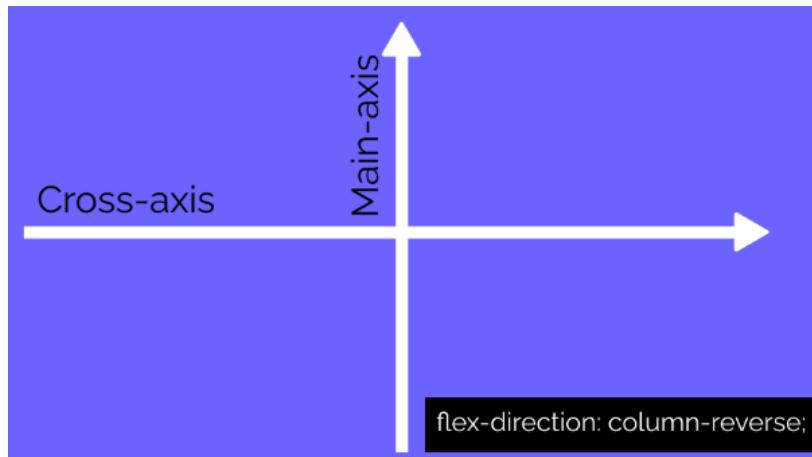
- `row-reverse`: Los elementos se alinean en una fila en orden inverso.



- `column`: Los elementos se alinean en una columna.



- `column-reverse`: Los elementos se alinean en una columna en orden inverso.



justify-content

Alinea los elementos flexibles a lo largo del **eje principal** (horizontal si `flex-direction` es `row`).

Los valores comunes son:

- `flex-start`: Alinea los elementos al inicio del contenedor.
- `flex-end`: Alinea los elementos al final.
- `center`: Centra los elementos.
- `space-between`: Deja espacio igual entre los elementos.
- `space-around`: Deja espacio igual entre los elementos, incluyendo espacio al inicio y al final.

align-items

Alinea los elementos a lo largo del **eje transversal** (vertical si `flex-direction` es `row`). Los valores comunes son:

- `flex-start`: Alinea los elementos al inicio.
- `flex-end`: Alinea los elementos al final.
- `center`: Centra los elementos.
- `stretch`: Estira los elementos para llenar el contenedor.
- `baseline`: Alinea los elementos según su línea de base de texto.

flex-wrap

Permite que los elementos flexibles se muevan a una nueva línea cuando el contenedor es demasiado pequeño:

- `nowrap`: Los elementos permanecen en una sola línea (predeterminado).
- `wrap`: Los elementos se envuelven en múltiples líneas si es necesario.
- `wrap-reverse`: Los elementos se envuelven en líneas múltiples en orden inverso.

Propiedades de los Elementos Flex

Estas propiedades se aplican a los elementos dentro del contenedor flex:

flex

Es una propiedad abreviada para `flex-grow`, `flex-shrink`, y `flex-basis`:

- `flex-grow`: Define cuánto crecerá un elemento en proporción a otros.
- `flex-shrink`: Define cuánto encogerá un elemento en proporción a otros.
- `flex-basis`: Define el tamaño inicial antes de que el elemento crezca o se encoja.

```
.item {  
    flex: 1; /* Ocupa el mismo espacio que otros elementos con flex: 1 */  
}
```

align-self

Permite alinear un solo elemento a lo largo del eje transversal, sobrescribiendo `align-items` del contenedor:

- `flex-start`, `flex-end`, `center`, `stretch`, `baseline`.

Taller

html

```
<!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="css/flex.css">  
    <title>Document</title>  
  </head>  
  <body>  
    <section class="parent">  
      <div class="item">1</div>  
      <div class="item">2</div>  
      <div class="item">3</div>  
    </section>  
  </body>  
</html>
```

```

.parent {
    display: block;
}

.item {
    border: 1px solid;
    opacity: 0.9;
    width: 100px;
    height: 100px;
    background: rgb(108, 151, 210);
    justify-content: center;
    align-items: center;
    font-weight: bold;
    color: #000000;
}

/* Estilos específicos para el primer y último elemento */
.item:first-child {
    background: yellow;
}

.item:last-child {
    background: rgb(210, 12, 65);
}

```

Aplicando Flex en el CSS

```

.parent {
    display: flex; /* Flexbox para alinear los elementos en línea */
}

.item {
    border: 1px solid;
    opacity: 0.9;
    width: 100px;
    height: 100px;
    background: rgb(108, 151, 210);
    display: flex;
    justify-content: center;
    align-items: center;
    font-weight: bold;
    color: #ffff;
}

/* Estilos específicos para el primer y último elemento */
.item:first-child {
    background: rgb(145, 145, 12);
}

.item:last-child {
    background: rgb(210, 12, 65);
}

```

Aplicando Flex-Direction

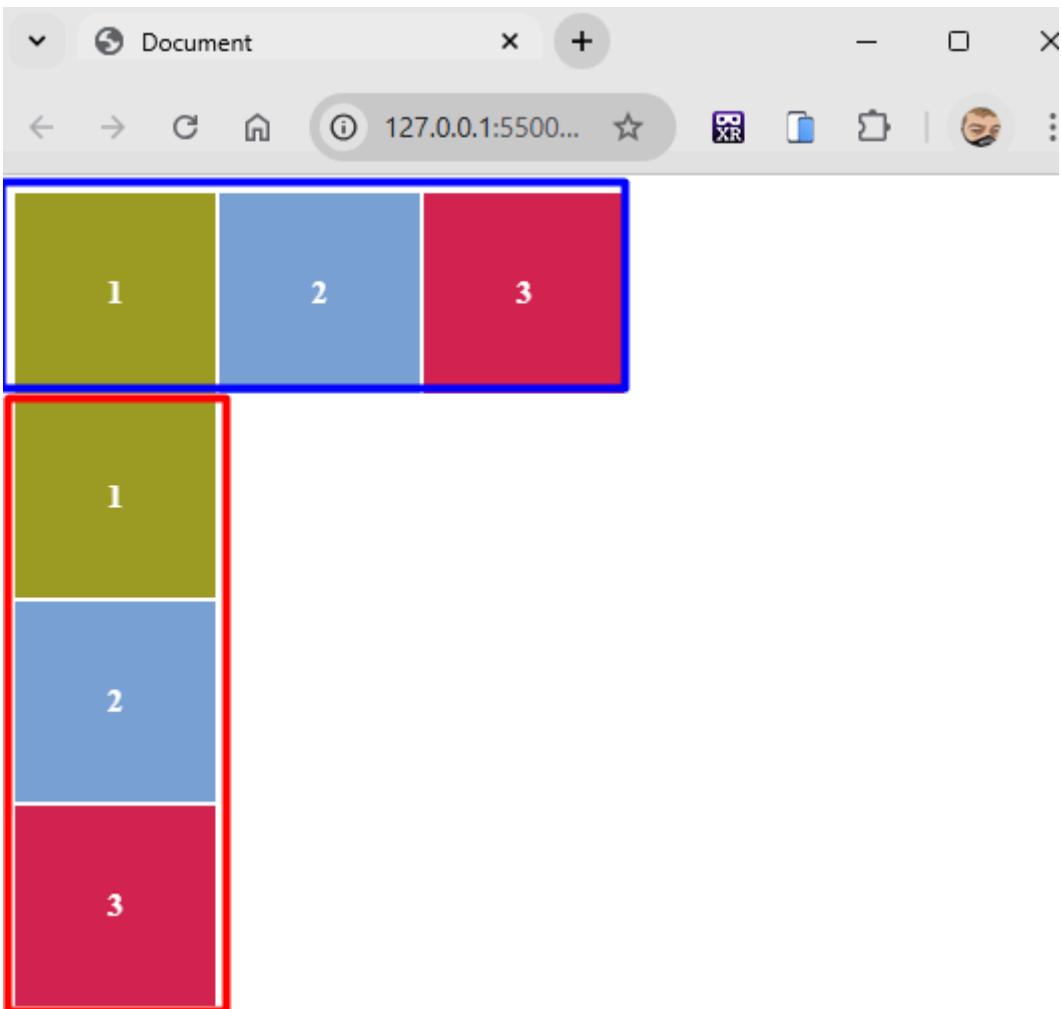
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="css/flex.css">
    <title>Document</title>
</head>
<body>
    <section class="parent">
        <div class="item">1</div>
        <div class="item">2</div>
        <div class="item">3</div>
    </section>
    <section class="parent2">
        <div class="item">1</div>
        <div class="item">2</div>
        <div class="item">3</div>
    </section>
</body>
</html>
```

```
.parent {
    display: flex; /* Flexbox para alinear los elementos en línea */
    flex-direction: row;
}
.parent2 {
    display: flex; /* Flexbox para alinear los elementos en línea */
    flex-direction: column;
}

.item {
    border: 1px solid;
    opacity: 0.9;
    width: 100px;
    height: 100px;
    background: rgb(108, 151, 210);
    display: flex;
    justify-content: center;
    align-items: center;
    font-weight: bold;
    color: #fff;
}

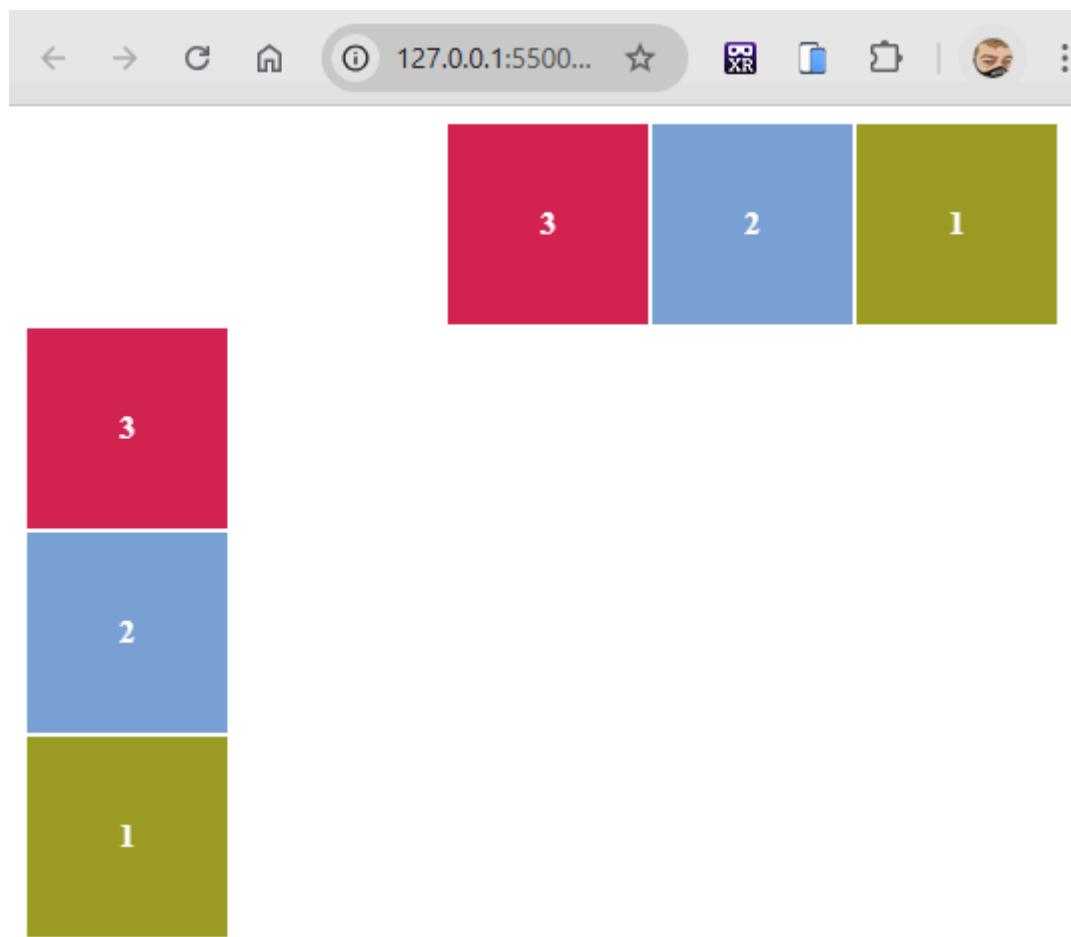
/* Estilos específicos para el primer y último elemento */
.item:first-child {
    background: rgb(145, 145, 12);
}

.item:last-child {
    background: rgb(210, 12, 65);
```



```
.parent {  
    display: flex; /* Flexbox para alinear los elementos en línea */  
    /* flex-direction: row; */  
    flex-direction: row-reverse;  
}  
.parent2 {  
    display: flex; /* Flexbox para alinear los elementos en línea */  
    flex-direction: column-reverse;  
}  
  
.item {  
    border: 1px solid;  
    opacity: 0.9;  
    width: 100px;  
    height: 100px;  
    background: rgb(108, 151, 210);  
    display: flex;  
    justify-content: center;  
    align-items: center;  
    font-weight: bold;  
    color: #fff;  
}  
  
/* Estilos específicos para el primer y último elemento */  
.item:first-child {  
    background: rgb(145, 145, 12);  
}
```

```
.item:last-child {  
    background: rgb(210, 12, 65);  
}
```



```
.parent {  
    display: flex; /* Flexbox para alinear los elementos en línea */  
    flex-direction: row;  
    direction: rtl;  
}  
.parent2 {  
    display: flex; /* Flexbox para alinear los elementos en línea */  
    flex-direction: column-reverse;  
    direction: ltr;  
}  
  
.item {  
    border: 1px solid;  
    opacity: 0.9;  
    width: 100px;  
    height: 100px;  
    background: rgb(108, 151, 210);  
    display: flex;  
    justify-content: center;  
    align-items: center;  
    font-weight: bold;  
    color: #fff;  
}
```

```
/* Estilos específicos para el primer y último elemento */
.item:first-child {
    background: rgb(145, 145, 12);
}

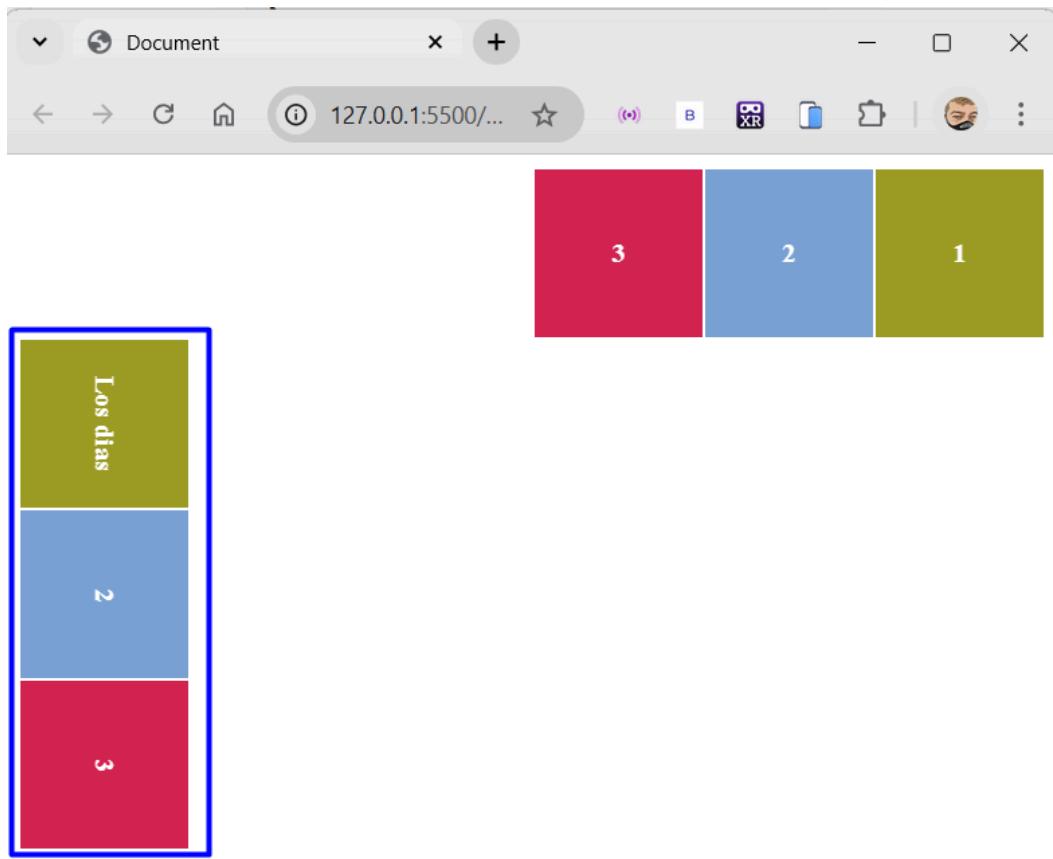
.item:last-child {
    background: rgb(210, 12, 65);
}
```

```
.parent {
    display: flex; /* Flexbox para alinear los elementos en línea */
    flex-direction: row;
    direction: rtl;
}
.parent2 {
    display: flex; /* Flexbox para alinear los elementos en línea */
    flex-direction: row;
    writing-mode: vertical-rl;
}

.item {
    border: 1px solid;
    opacity: 0.9;
    width: 100px;
    height: 100px;
    background: rgb(108, 151, 210);
    display: flex;
    justify-content: center;
    align-items: center;
    font-weight: bold;
    color: #fff;
}

/* Estilos específicos para el primer y último elemento */
.item:first-child {
    background: rgb(145, 145, 12);
}

.item:last-child {
    background: rgb(210, 12, 65);
}
```

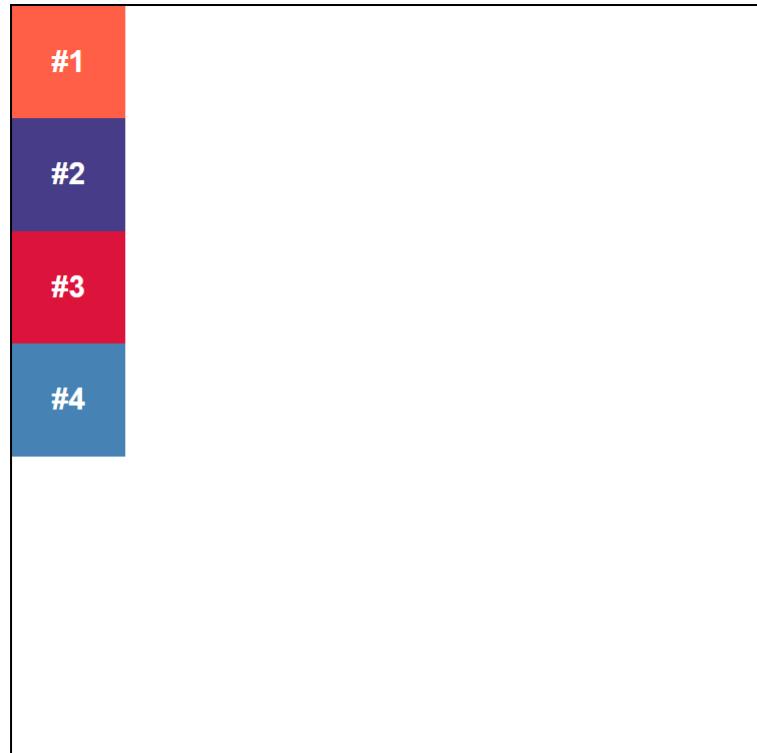


Taller Nro 2

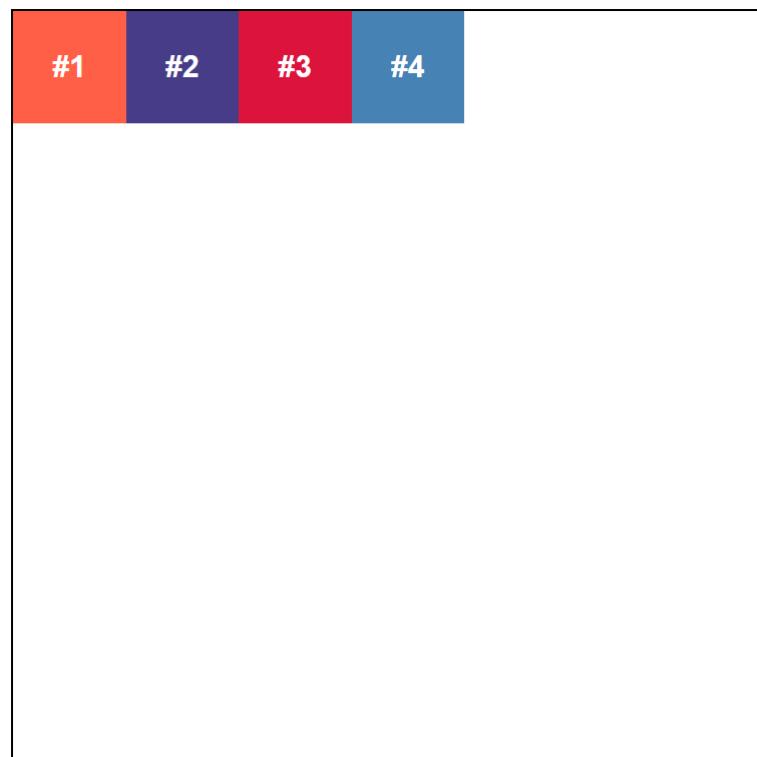
1. Cree el siguiente HTML

```
<!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="css/flexcustom.css">
    <title>Document</title>
</head>
<body>
    <div class="container">
        <div class="item item1">#1</div>
        <div class="item item2">#2</div>
        <div class="item item3">#3</div>
        <div class="item item4">#4</div>
    </div>
</body>
</html>
```

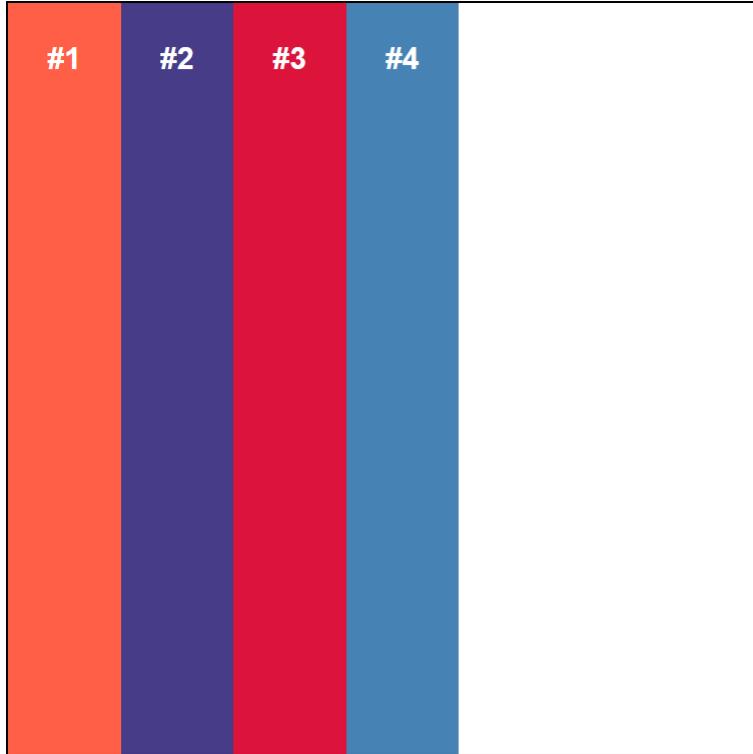
2. Genere el CSS para lograr el siguiente aspecto



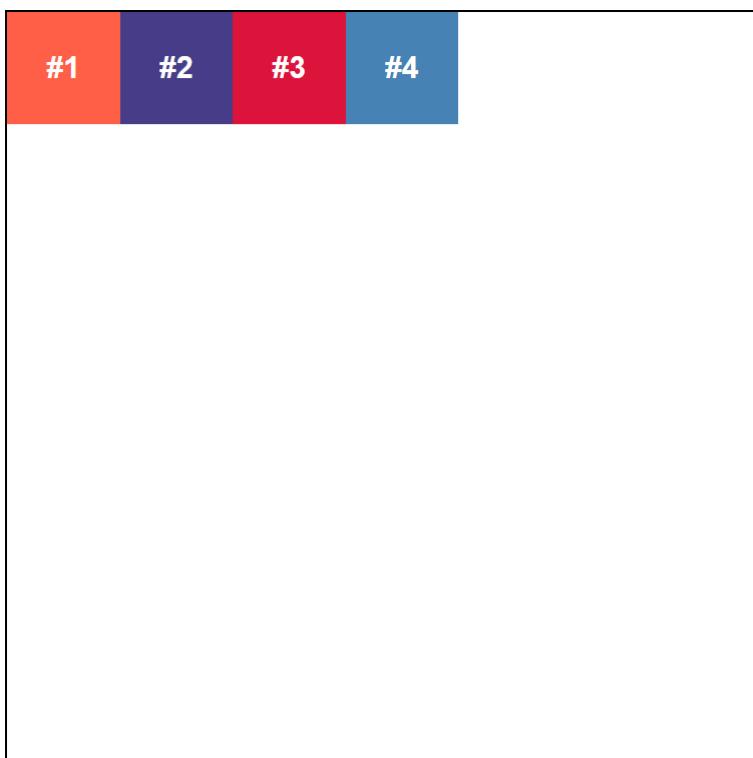
3. Modifique el CSS para lograr la siguiente disposición de los elementos



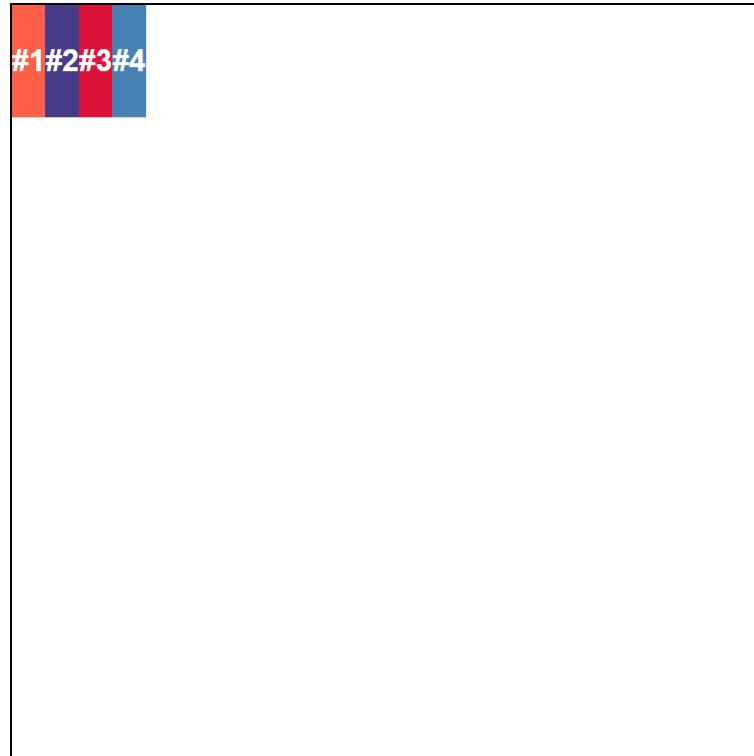
4. Modifique el CSS para lograr la siguiente disposición de los elementos



5. Modifique el CSS para lograr la siguiente disposicion de elementos usando align items



6. Modifique el CSS para lograr la siguiente disposicion de elementos



Taller 3 - <https://flexboxfroggy.com/#es>

Taller 4 - <https://mastery.games/flexboxzombies/>

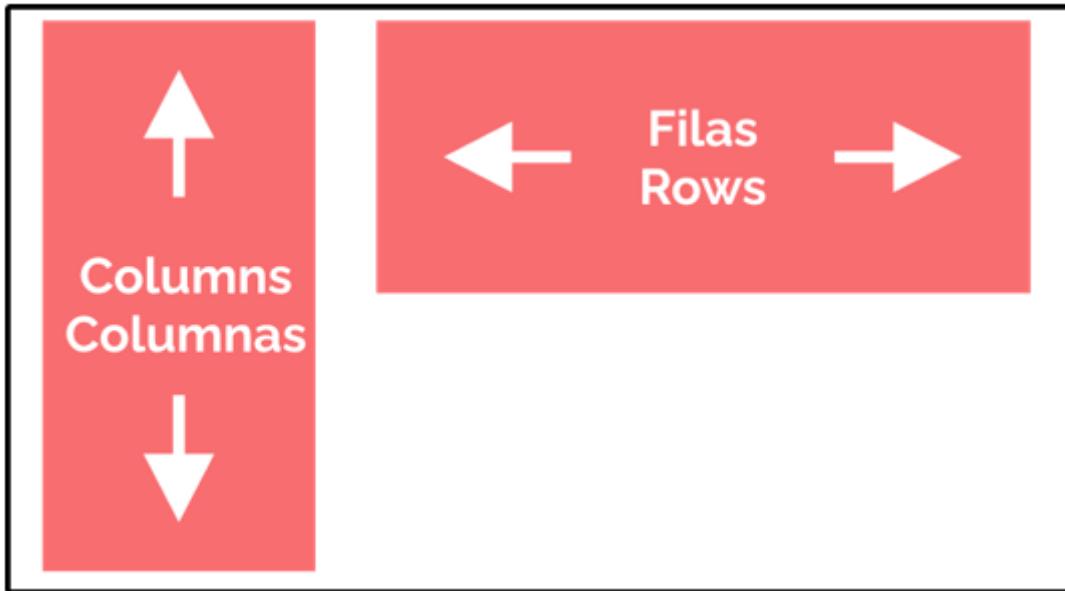
Grid <https://css-tricks.com/snippets/css/complete-guide-grid/>

<https://cssgridgenerator.io/>

CSS Grid Layout es una herramienta de diseño en CSS extremadamente potente. Su sistema bidimensional permite organizar tanto filas como columnas, a diferencia de Flexbox, que está más enfocado en un diseño unidimensional.

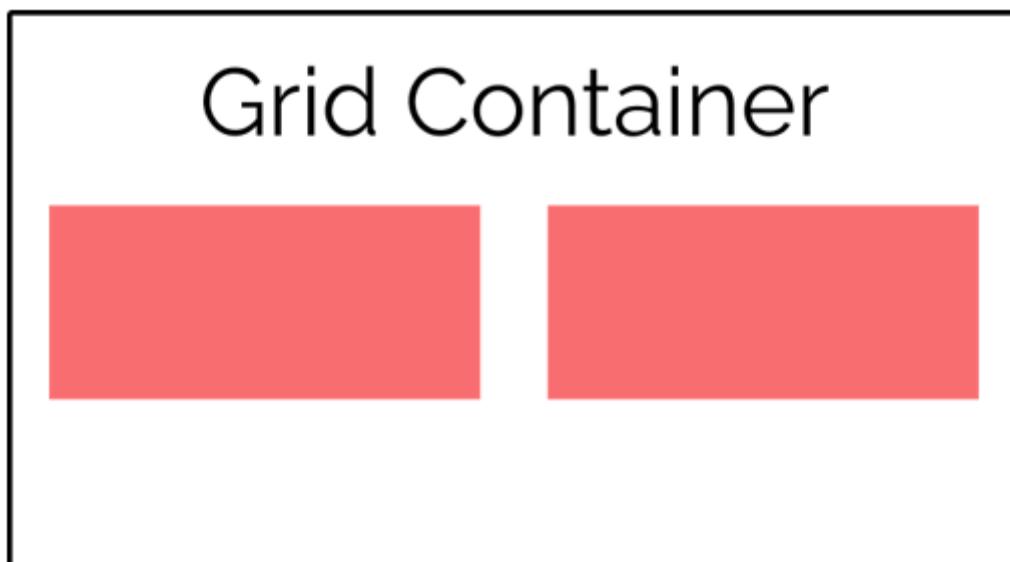
Filas y Columnas

CSS Grid utiliza un sistema bidimensional que permite organizar elementos en filas y columnas. Las filas (rows) se disponen horizontalmente, mientras que las columnas (columns) se organizan de forma vertical. Un ejemplo de este tipo de disposición es una hoja de cálculo en Excel.



Grid container

De la misma manera que en flexbox, para poder utilizar css grid, necesitamos un contenedor, a este contenedor se le denominará grid-container.



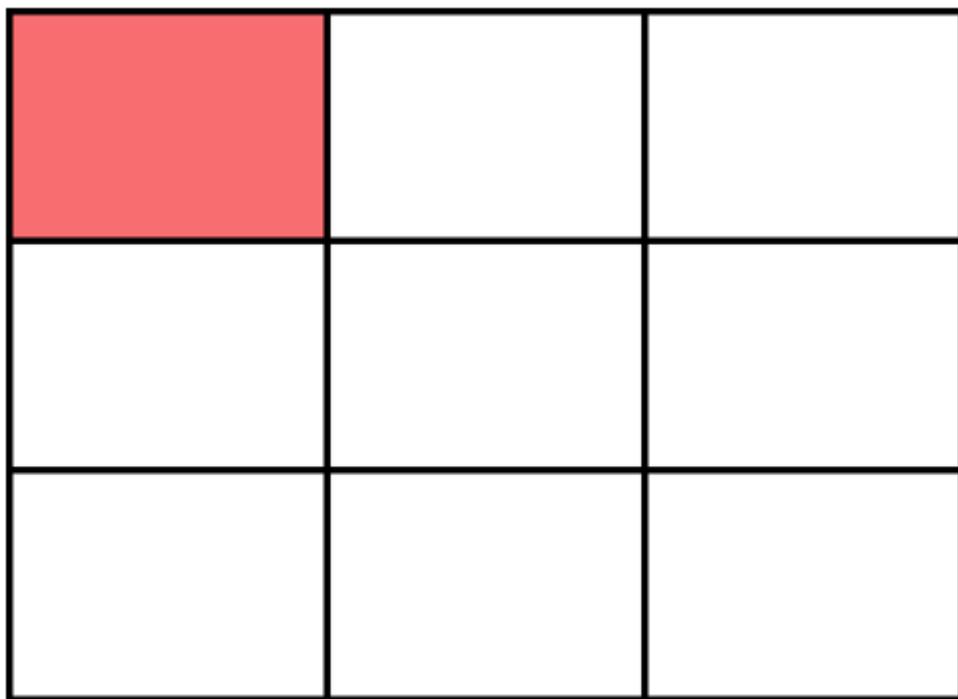
Grid Item

Teniendo creado un grid-container, es muy importantes saber, que los hijos directos del grid-container se les conoce como Grid-items.



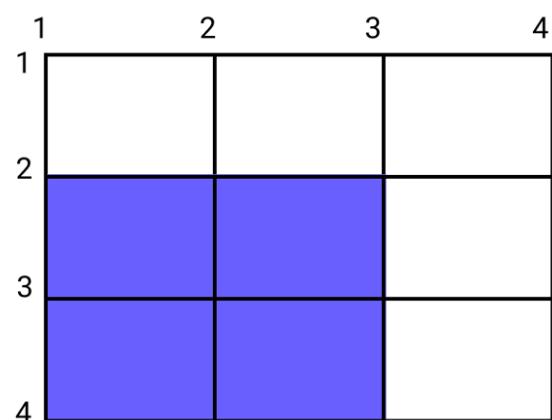
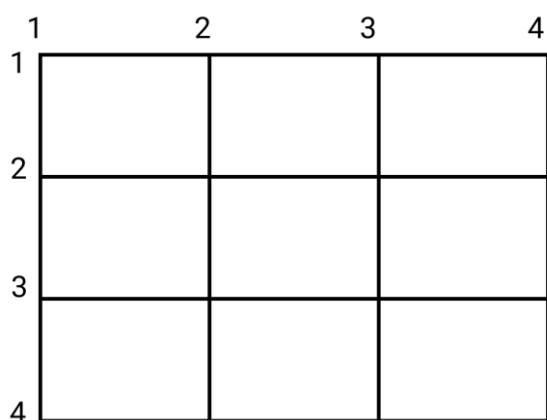
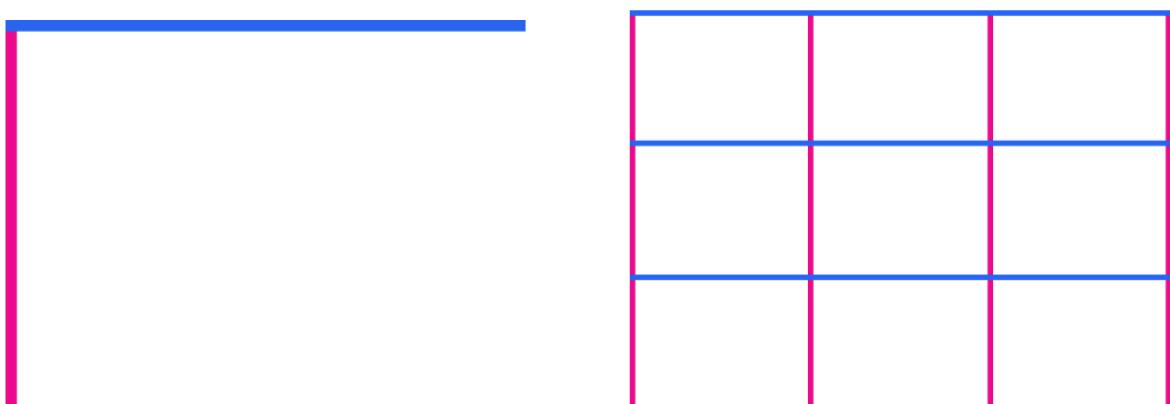
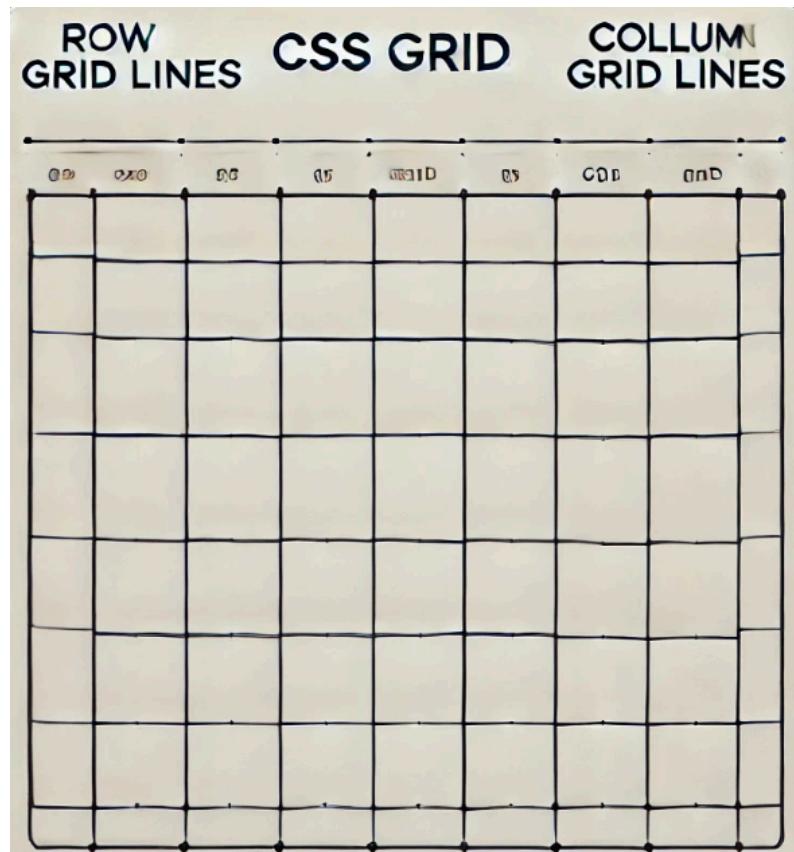
Grid Cell (Celda)

Para comprender el sistema de GRID, es útil visualizarlo como una cuadrícula formada por múltiples celdas. En su configuración inicial, cada elemento de la cuadrícula, conocido como grid-item, ocupará una sola celda de la estructura. Esta cuadrícula permite organizar el contenido de manera estructurada, proporcionando flexibilidad para expandir o contraer los elementos según las necesidades del diseño. Cada celda actúa como una unidad básica en la que los grid-items pueden ubicarse, permitiendo que los desarrolladores controlen con precisión el posicionamiento y la alineación de los elementos dentro de un contenedor bidimensional.



Grid Lines (Líneas)

El sistema CSS GRID funciona como una cuadrícula, compuesta por una serie de líneas que definen su estructura. Estas líneas incluyen tanto líneas verticales, llamadas líneas de columna (column grid lines), como líneas horizontales, conocidas como líneas de fila (row grid lines). Juntas, estas líneas forman el esqueleto sobre el cual se distribuyen los elementos de la cuadrícula, facilitando un control preciso sobre la organización y disposición del contenido en la página.



Ejercicio

```
<!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>Document</title>
</head>
<body>
<div class="grid">

    <div class="grid__item">1</div>

    <div class="grid__item">2</div>

    <div class="grid__item">3</div>

    <div class="grid__item">4</div>

    <div class="grid__item">5</div>

    <div class="grid__item">6</div>

</div>
</body>
</html>
```

```
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body{
    font-family: 'Arial';
}

.grid{
    max-width: 700px;
    width: 90%;
    height: 500px;
    margin: 20px auto;
    outline: 3px solid #000;
}

.grid__item{
    color: #fff;
    font-size: 3rem;
    text-align: center;
    background: crimson;
}

.grid :nth-child(1){
    background: crimson;
}

.grid :nth-child(2){
    background: darkmagenta;
}
```

```

.grid :nth-child(3){
    background: steelblue;
}
.grid :nth-child(4){
    background: brown;
}
.grid :nth-child(5){
    background: chocolate;
}
.grid :nth-child(6){
    background: midnightblue;
}

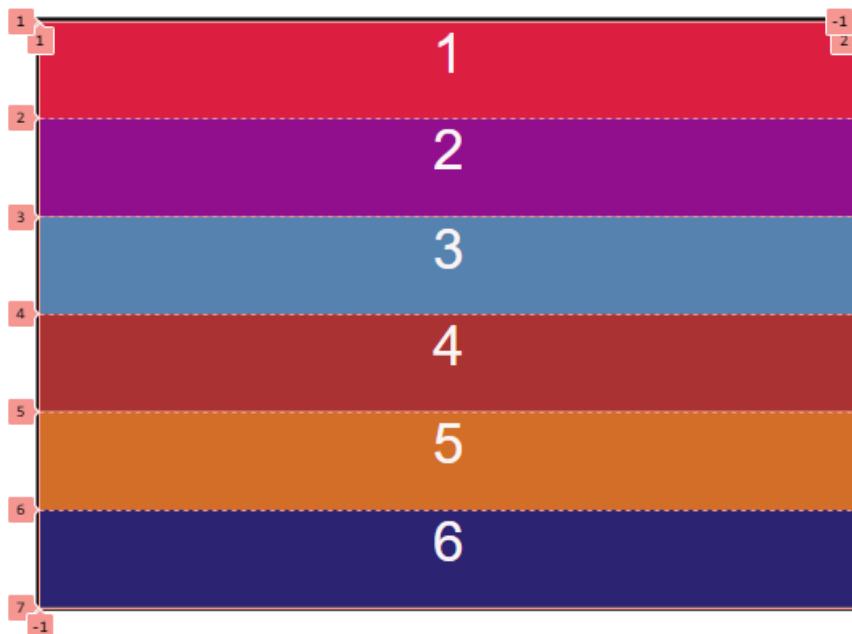
```

a la regla .grid agreguele

```
display: grid;
```

Visualizar el esquema grid en Chrome

Abra el inspector, seleccione el elemento definido como grid y seleccione la pestaña layout y marque la opción .grid o haga clic en el botón rápido grid.



Propiedades GRID

grid-template-columns

La propiedad `grid-template-columns` en CSS Grid se utiliza para definir el número y el tamaño de las columnas en una cuadícula. Es fundamental para crear la estructura del contenedor grid y establecer cómo se distribuirá el espacio en el eje horizontal. Con esta propiedad, puedes especificar tanto el número de columnas como su ancho, ya sea con valores específicos, proporcionales, o flexibles.

```
.container {  
    display: grid;  
    grid-template-columns: 100px 100px 100px;  
}
```

```
.container {  
    display: grid;  
    grid-template-columns: repeat(3, 100px);  
}
```

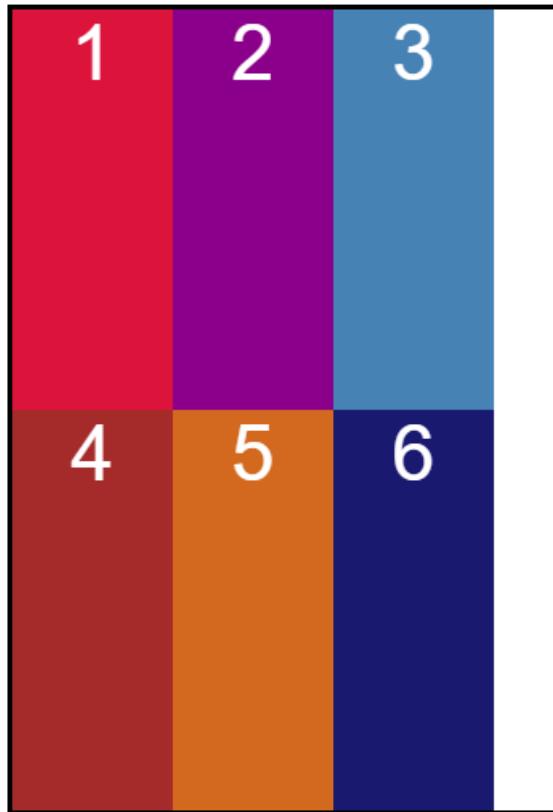
```
.container {  
    display: grid;  
    grid-template-columns: 1fr 2fr 1fr;  
}
```

```
.container {  
    display: grid;  
    grid-template-columns: 200px 1fr 2fr;  
}
```

```
.container {  
    display: grid;  
    grid-template-columns: auto auto auto;  
}
```

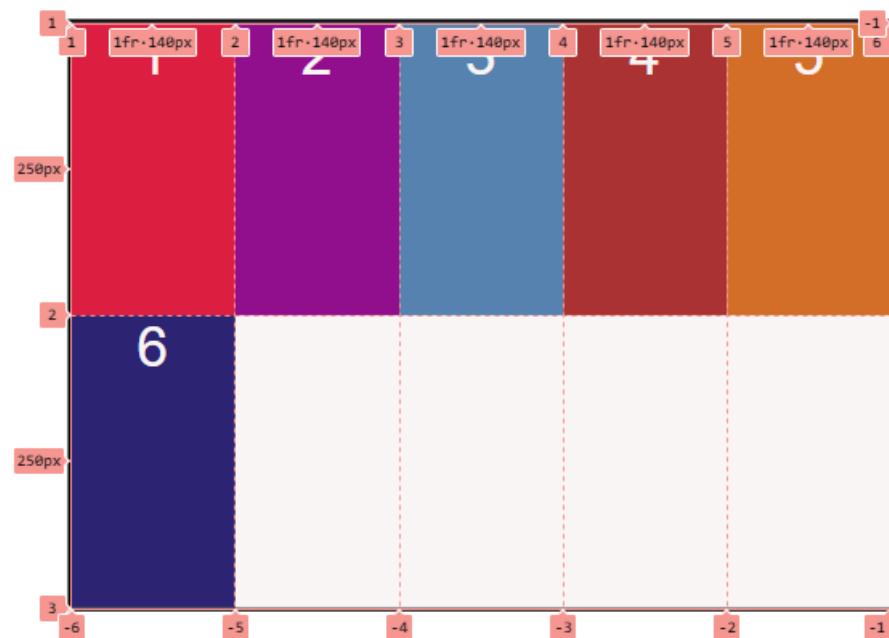
Agregue en el css la propiedad

```
grid-template-columns: 100px 100px 100px;  
  
grid-template-columns: repeat(5,10%);
```



Creando columnas por fracción

```
grid-template-columns: 1fr 1fr 1fr;
```



DevTools is now available in Spanish!

Always match Chrome's language Switch DevTools to Spanish

Don't show again

Elements Console Sources Network >> 3 :

```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <div class="grid">...</div> grid == $0
    <!-- Code injected by live-server -->
    <script>...</script>
  </body>
</html>
```

html body div.grid Layout Event Listeners DOM Breakpoints Properties >>

▼ Grid

Overlay display settings

Show line numbers ▾

Show track sizes ←

Show area names

Extend grid lines

grid-template-rows

La propiedad `grid-template-rows` en CSS Grid define el número y el tamaño de las filas en una cuadrícula, similar a cómo `grid-template-columns` funciona para las columnas. Esta propiedad te permite controlar la altura de cada fila en el contenedor grid, facilitando la organización de los elementos en el eje vertical.

```
.container {
  display: grid;
  grid-template-rows: valor1 valor2 valor3 ...;
```

tamaño fijo

```
.container {
  display: grid;
  grid-template-rows: 100px 200px 150px;
```

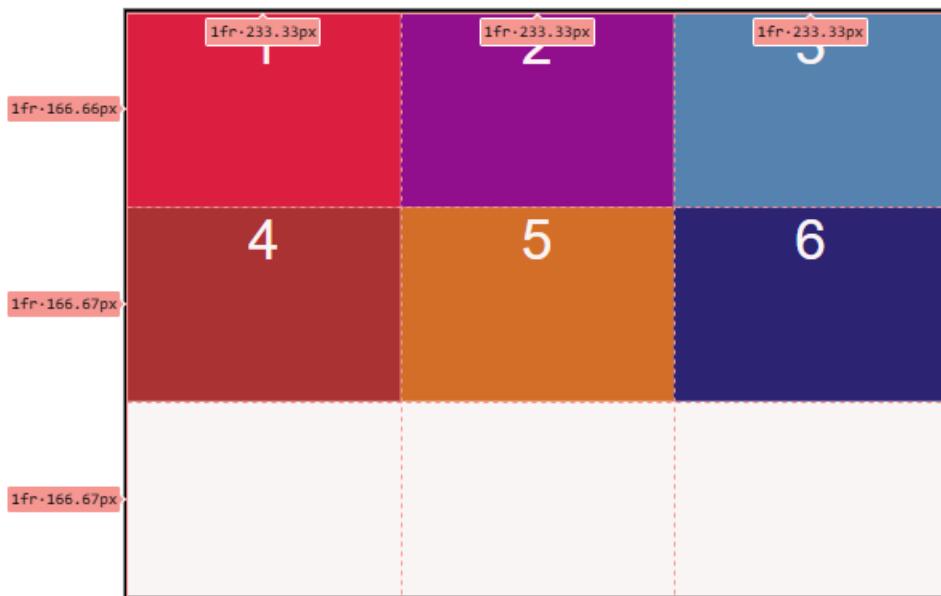
```
.container {  
    display: grid;  
    grid-template-rows: repeat(3, 100px);  
}
```

```
.container {  
    display: grid;  
    grid-template-rows: 1fr 2fr 1fr;  
}
```

```
.container {  
    display: grid;  
    grid-template-rows: 150px 1fr 2fr;  
}
```

en el css del proyecto en la regla .grid modifique o agregue estas propiedades

```
grid-template-columns: repeat(3,1fr);  
grid-template-rows: repeat(3,1fr);
```



Posicionamiento grid items por líneas

grid-column-start

La propiedad `grid-column-start` en CSS Grid se utiliza para definir el inicio de una columna específica en una cuadrícula, es decir, en qué línea de la cuadrícula comienza un elemento. Esta propiedad es útil para posicionar elementos de manera precisa dentro de una estructura grid, especialmente cuando necesitas que un elemento ocupe varias columnas o comience en un lugar específico de la cuadrícula.

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="style.css">
7   <title>Document</title>
8 </head>
9 <body>
10  <div class="grid">
11    <div class="grid__item">1</div>
12    <div class="grid__item">2</div>
13    <div class="grid__item">3</div>
14    <div class="grid__item">4</div>
15    <div class="grid__item">5</div>
16    <div class="grid__item">6</div>
17  </div>
18 </body>
19 </html>

```

```

1 .grid__item {
2   margin: 0;
3   padding: 0;
4   box-sizing: border-box;
5 }
6 body {
7   font-family: 'Arial';
8 }
9 .grid {
10  max-width: 700px;
11  width: 90%;
12  height: 500px;
13  margin: 20px auto;
14  outline: 3px solid #000;
15  display: grid;
16  grid-template-columns: repeat(4,1fr);
17  grid-template-rows: repeat(2,1fr);
18 }
19 .grid__item {
20  color: #fff;
21  font-size: 3rem;
22  text-align: center;
23  background: #crimson;
24 }
25 .grid :nth-child(1){
26   background: #crimson;
27   grid-column-start: 3;
28   grid-column-end: 5;
29 }
30 }
31 }
32 }
33 }

```

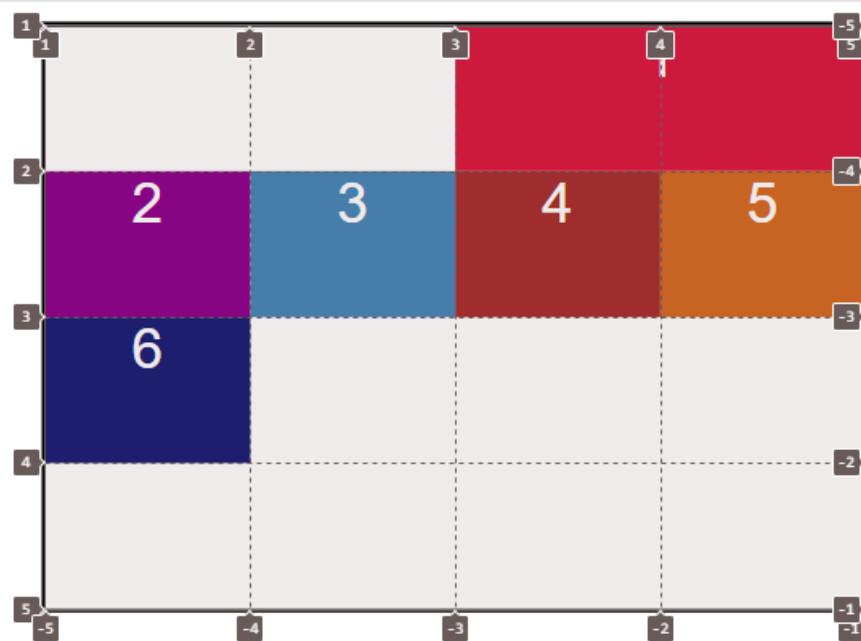
grid-column-end

La propiedad `grid-column-end` en CSS Grid se utiliza para especificar en qué línea de la cuadrícula (grid line) debe terminar un elemento en el eje horizontal (de izquierda a derecha). Con esta propiedad, puedes controlar cuántas columnas abarca un elemento, permitiéndote definir la posición final y el ancho del mismo en la cuadrícula.

```

28
29 .grid :nth-child(1){
30   background: #crimson;
31   grid-column-start: 3;
32   grid-column-end: 5;
33 }

```

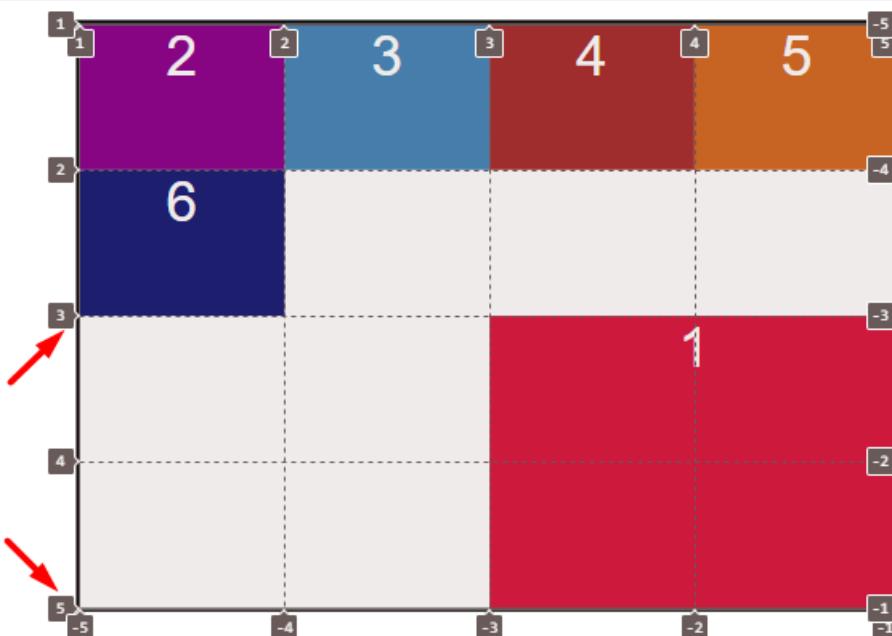


grid-row-start y grid-row-end

La propiedad `grid-row-start` en CSS Grid especifica en qué línea de la cuadrícula (grid line) comenzará a colocarse un elemento en el eje vertical (de arriba hacia abajo). Esto te permite controlar la posición de inicio de un elemento en una cuadrícula sin tener que modificar el orden del HTML o usar márgenes.

La propiedad `grid-row-end` en CSS Grid especifica en qué línea de la cuadrícula (grid line) debe terminar un elemento en el eje vertical (de arriba hacia abajo). Esta propiedad es útil para controlar cuántas filas ocupa un elemento, definiendo su altura y posición en la cuadrícula.

```
29 .grid :nth-child(1){  
30     background: #crimson;  
31     grid-column-start: 3;  
32     grid-column-end: 5;  
33     grid-row-start: 3;  
34     grid-row-end: 5;  
35 }
```



Modo Abreviado

```
28  
29 <.grid :nth-child(1)> Inicio  
30     background: #crimson;  
31     grid-column: 3 / 5; ← Termina  
32     grid-row: 3 / 5;  
33 }
```

span

En CSS Grid, la palabra clave `span` se utiliza en combinación con propiedades como `grid-column-start`, `grid-column-end`, `grid-row-start` y `grid-row-end` para hacer que un elemento abarque varias columnas o filas sin necesidad de especificar una línea de finalización exacta. Esto es útil cuando deseas que un elemento ocupe un número específico de columnas o filas sin preocuparte por el número exacto de las líneas de la cuadrícula.

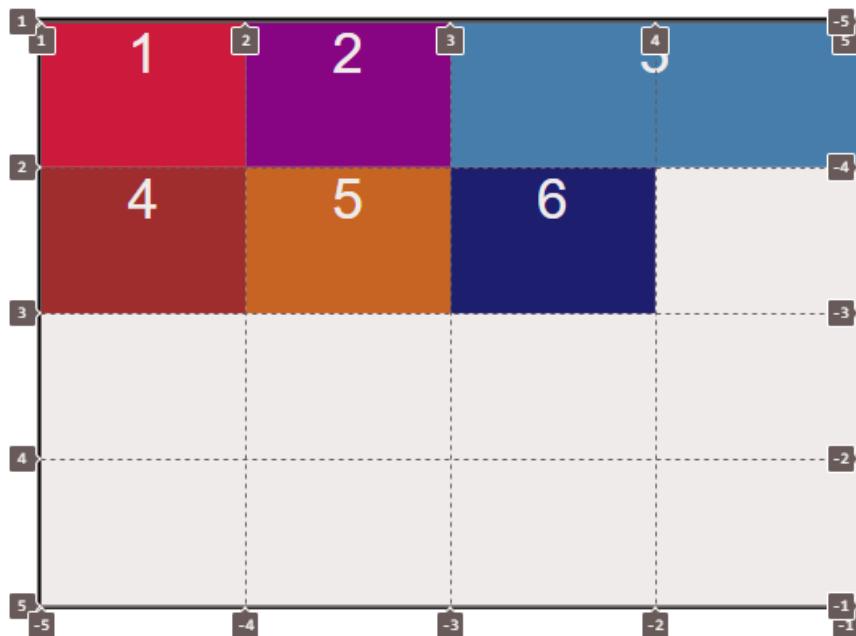
Uso de `span` en CSS Grid

1. Para abarcar varias columnas (grid-column)

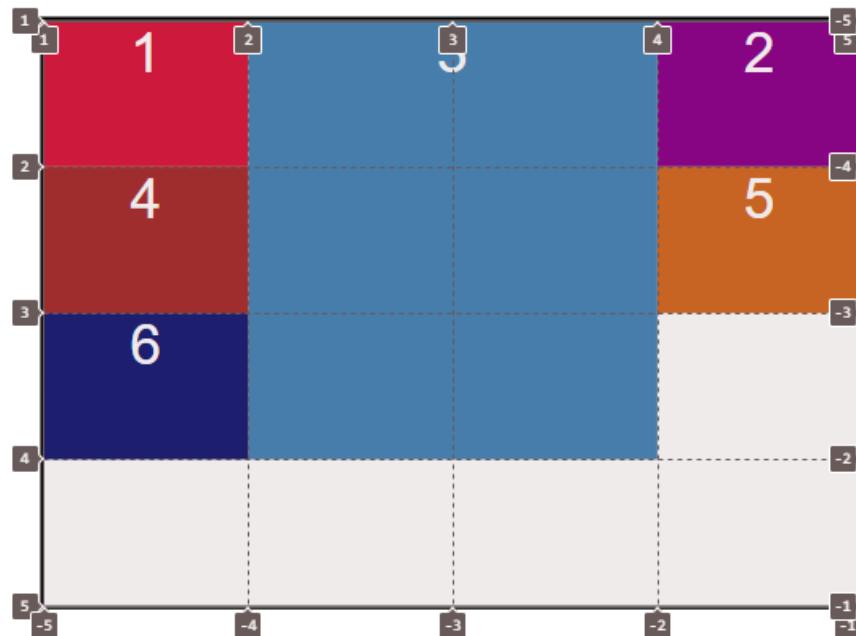
Puedes usar `grid-column` para que un elemento abarque múltiples columnas en el contenedor de cuadrícula.

```
.elemento {  
    grid-column: span 2; /* El elemento abarcará dos columnas */  
}
```

```
38 .grid :nth-child(3){  
39     background: steelblue;  
40     grid-column: span 2; →  
41 }
```



```
38 .grid :nth-child(3){  
39     background: steelblue;  
40     grid-column: 2 / span 2;  
41     grid-row: 1 / span 3;  
42 }
```



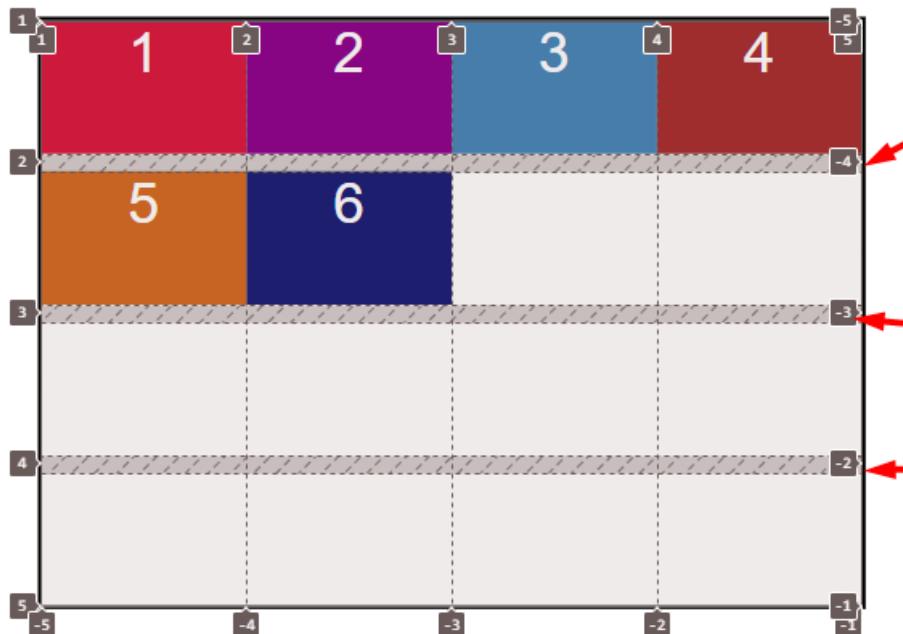
Espaciadores o Gutters (Se definen en el container)

row-gap

La propiedad `row-gap` en CSS Grid define el espacio (o "gap") entre las filas de una cuadrícula. Esto permite controlar el espacio vertical entre los elementos de las filas sin necesidad de aplicar márgenes individuales a cada elemento.

```
.container {  
    display: grid;  
    row-gap: valor;  
}
```

```
.1   .grid{  
.2     max-width: 700px;  
.3     width: 90%;  
.4     height: 500px;  
.5     margin: 20px auto;  
.6     outline: 3px solid #000;  
.7     display: grid;  
.8     grid-template-columns: repeat(4,1fr);  
.9     grid-template-rows: repeat(4,1fr);  
.10    row-gap: 15px;  
.11}  
.12
```

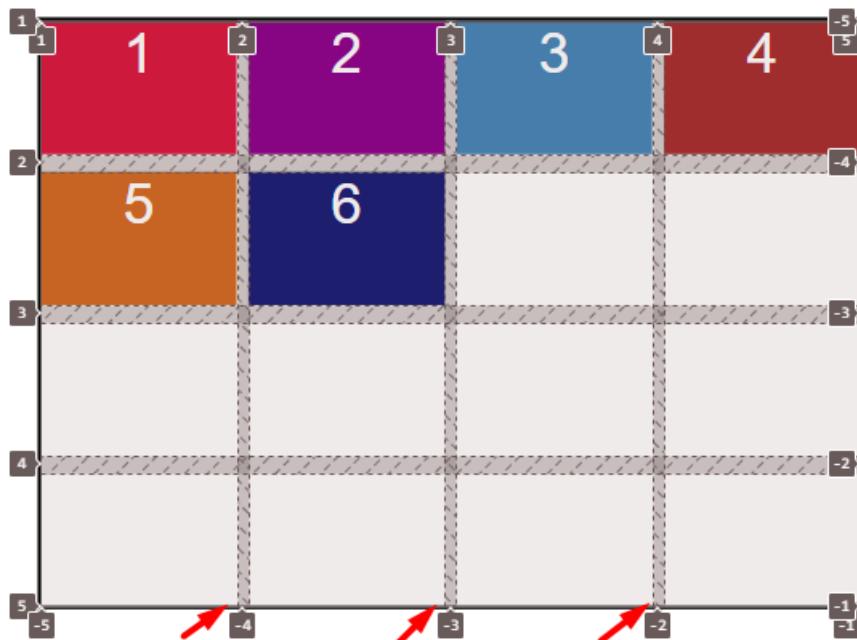


column-gap

La propiedad `column-gap` en CSS Grid define el espacio entre las columnas de una cuadrícula. Esto permite controlar el espacio horizontal entre los elementos de las columnas sin necesidad de aplicar márgenes individuales a cada elemento.

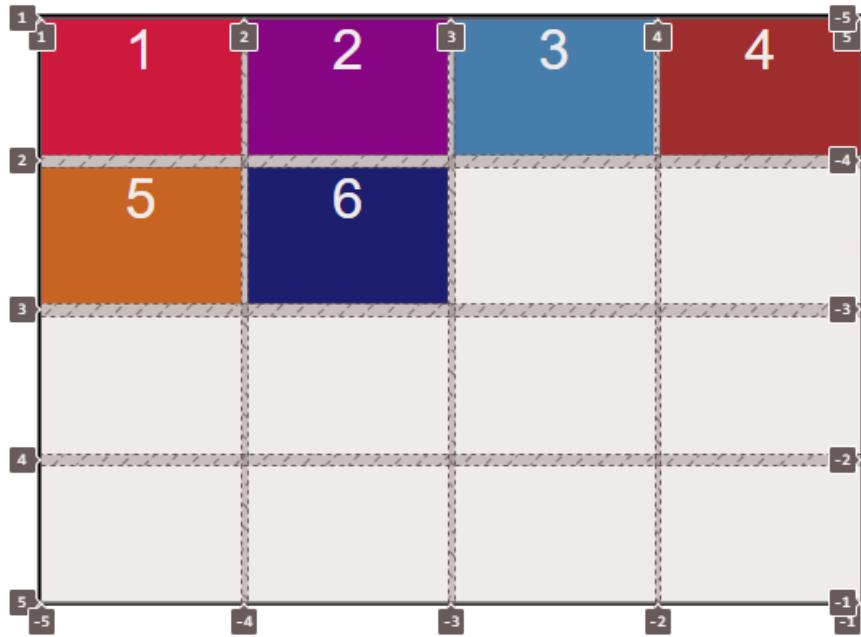
```
.container {  
    display: grid;  
    column-gap: valor;  
}
```

```
1 .grid{  
2     max-width: 700px;  
3     width: 90%;  
4     height: 500px;  
5     margin: 20px auto;  
6     outline: 3px solid #000;  
7     display: grid;  
8     grid-template-columns: repeat(4,1fr);  
9     grid-template-rows: repeat(4,1fr);  
10    row-gap: 15px;  
11    column-gap: 10px; ←  
12 }
```



```
1 .grid{  
2     max-width: 700px;  
3     width: 90%;  
4     height: 500px;  
5     margin: 20px auto;  
6     outline: 3px solid #000;  
7     display: grid;  
8     grid-template-columns: repeat(4,1fr);  
9     grid-template-rows: repeat(4,1fr);  
10    /* row-gap: 15px;  
11    column-gap: 10px; */  
12    gap: 10px 5px; ←  
13 }
```

Sep Filas **Sep Columnas**

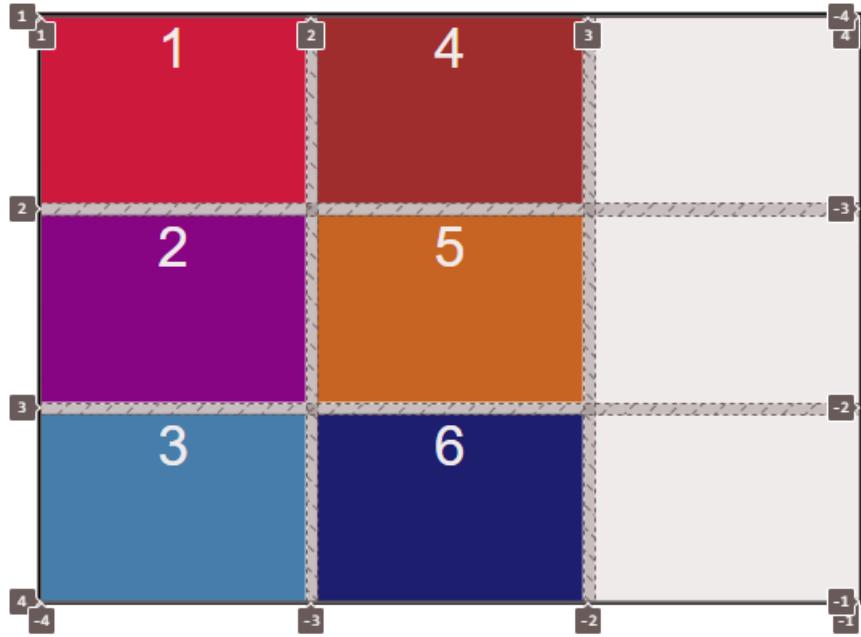


grid-auto-flow

La propiedad `grid-auto-flow` en CSS Grid define cómo se colocan automáticamente los elementos dentro de una cuadrícula cuando no se especifican posiciones exactas para ellos mediante `grid-area`, `grid-column`, o `grid-row`. Esto permite determinar si los elementos se ubicarán en filas o columnas y si se completarán en sentido horizontal o vertical.

```

11   .grid{
12     max-width: 700px;
13     width: 90%;
14     height: 500px;
15     margin: 20px auto;
16     outline: 3px solid □#000;
17     display: grid;
18     grid-template-columns: repeat(3,1fr);
19     grid-template-rows: repeat(3,1fr);
20     /* row-gap: 15px;
21     column-gap: 10px; */
22     gap: 10px;
23     grid-auto-flow: column; ↗
24 }
```



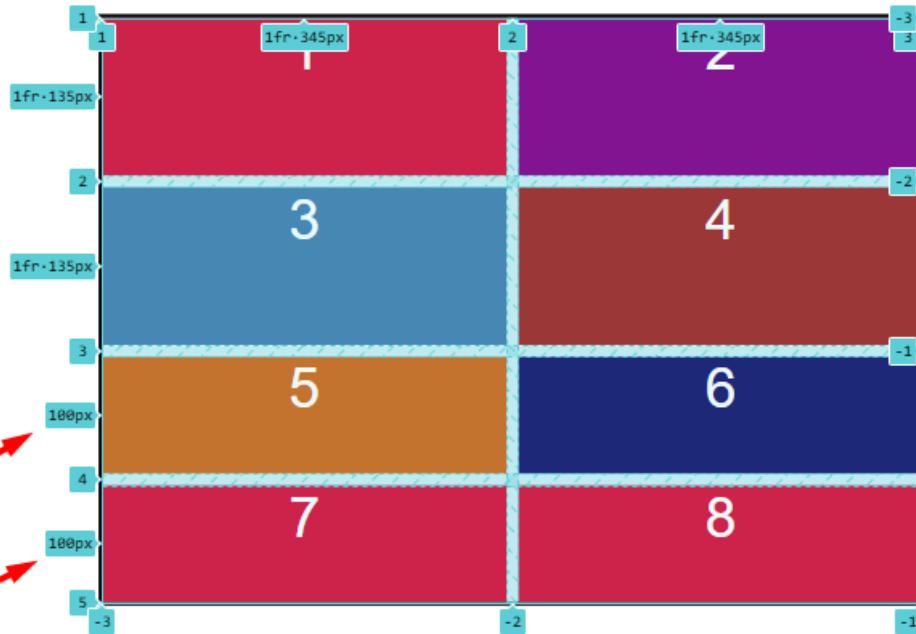
grid-auto-rows

La propiedad `grid-auto-rows` en CSS Grid define la altura de las filas generadas automáticamente en una cuadricula. Esto es útil cuando tienes más elementos que filas definidas, ya que cualquier fila adicional creada automáticamente usará la altura especificada por `grid-auto-rows`.

```
.container {
  display: grid;
  grid-auto-rows: valor;
}

.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-auto-rows: 150px; /* Cada fila automática tendrá una altura de 150px */
}
```

```
11   .grid{
12     max-width: 700px;
13     width: 90%;
14     height: 500px;
15     margin: 20px auto;
16     outline: 3px solid □#000;
17     display: grid;
18     grid-template-columns: repeat(2,1fr);
19     grid-template-rows: repeat(2,1fr);
20     /* row-gap: 15px;
21     column-gap: 10px; */
22     gap: 10px;
23     grid-auto-flow: row;
24     grid-auto-rows: 100px; ←
25 }
```



dense en CSS GRID

En CSS Grid, el valor `dense` se usa con la propiedad `grid-auto-flow` para organizar automáticamente los elementos de una cuadrícula, llenando los huecos que puedan quedar vacíos en el flujo natural de la cuadrícula. Este valor es útil cuando tienes elementos de diferentes tamaños y quieras que la cuadrícula los reorganice automáticamente para que ocupen el menor espacio posible, evitando espacios vacíos o huecos.

```
.container {
    display: grid;
    grid-auto-flow: row dense; /* o column dense */
}
```

```
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body{
    font-family: 'Arial';
}

.grid{
    max-width: 700px;
    width: 90%;
    height: 500px;
    margin: 20px auto;
    outline: 3px solid #000;
    display: grid;
    grid-template-columns: repeat(3,1fr);
    grid-template-rows: repeat(4,1fr);
    /* row-gap: 15px;
    column-gap: 10px; */
    gap: 10px;
```

```

        grid-auto-rows: 100px;
    }

.grid__item{
    color: #fff;
    font-size: 3rem;
    text-align: center;
    background: crimson;
}

.grid :nth-child(1){
    background: crimson;
}

.grid :nth-child(2){
    background: darkmagenta;
    grid-row: span 2;
}

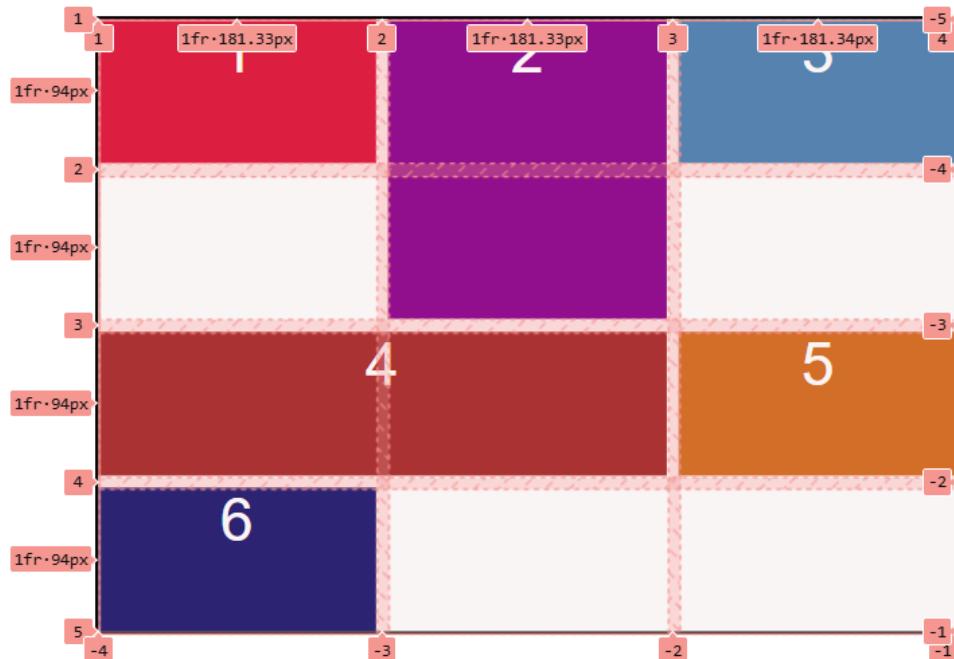
.grid :nth-child(3){
    background: steelblue;
    /* grid-column: 2 / span 2;
    grid-row: 1 / span 3; */
}

.grid :nth-child(4){
    background: brown;
    grid-column: span 2;
}

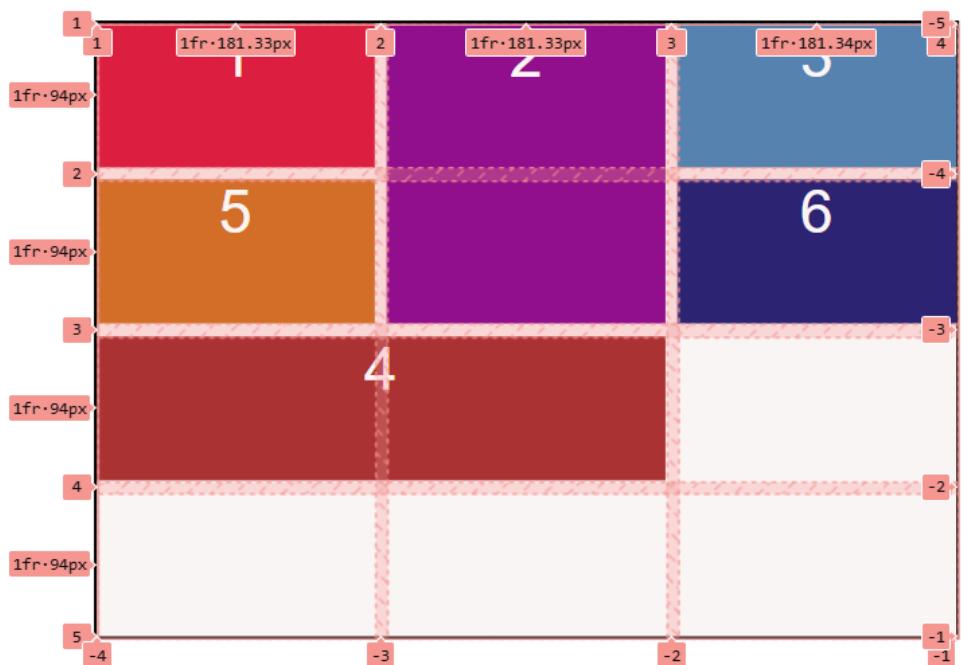
.grid :nth-child(5){
    background: chocolate;
}

.grid :nth-child(6){
    background: midnightblue;
}

```



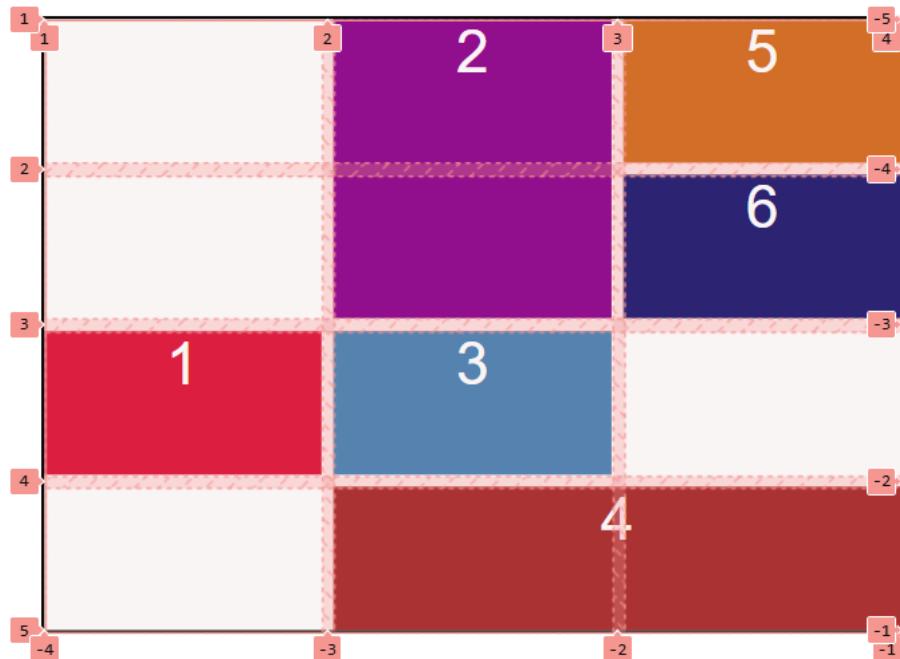
```
11 .grid{  
12     max-width: 700px;  
13     width: 90%;  
14     height: 500px;  
15     margin: 20px auto;  
16     outline: 3px solid #000;  
17     display: grid;  
18     grid-template-columns: repeat(3,1fr);  
19     grid-template-rows: repeat(4,1fr);  
20     /* row-gap: 15px;  
21     column-gap: 10px; */  
22     gap: 10px;  
23     grid-auto-rows: 100px;  
24     grid-auto-flow: dense;  
25 }
```



```

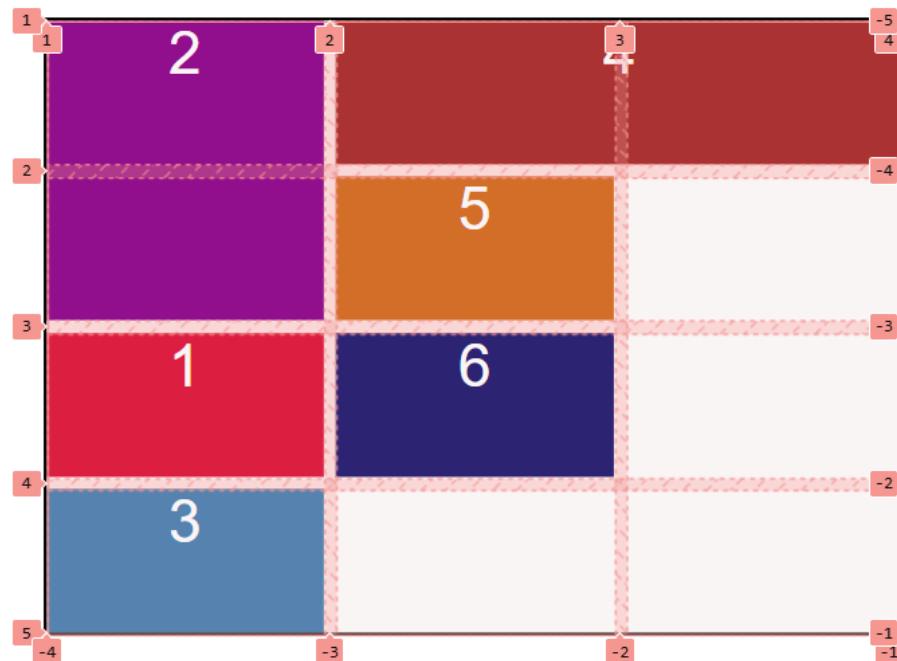
11 .grid{
12   max-width: 700px;
13   width: 90%;
14   height: 500px;
15   margin: 20px auto;
16   outline: 3px solid #000;
17   display: grid;
18   grid-template-columns: repeat(3,1fr);
19   grid-template-rows: repeat(4,1fr);
20   /* row-gap: 15px;
21   column-gap: 10px; */
22   gap: 10px;
23   grid-auto-rows: 100px;
24   grid-auto-flow: column; ←
25 }
26
27 .grid__item{
28   color: #fff;
29   font-size: 3rem;
30   text-align: center;
31   background: #crimson;
32 }
33
34 .grid :nth-child(1){
35   background: #crimson;
36   grid-row: 3; ←
37 }

```



```

11   .grid{
12     max-width: 700px;
13     width: 90%;
14     height: 500px;
15     margin: 20px auto;
16     outline: 3px solid #000;
17     display: grid;
18     grid-template-columns: repeat(3,1fr);
19     grid-template-rows: repeat(4,1fr);
20     /* row-gap: 15px;
21     column-gap: 10px; */
22     gap: 10px;
23     grid-auto-rows: 100px;
24     grid-auto-flow: column dense; ↗
25 }
```



grid-template-areas

La propiedad `grid-template-areas` en CSS Grid permite crear una cuadrícula nombrando áreas específicas dentro de la misma, lo que facilita la organización visual y el posicionamiento de los elementos. Con esta propiedad, puedes definir el diseño de la cuadrícula de forma visual usando nombres de área en lugar de especificar filas y columnas para cada elemento individualmente.

```
.container {
  display: grid;
  grid-template-areas:
    "header header"
    "sidebar content"
    "footer footer";
}
```

Cómo Funciona `grid-template-areas`

- La cuadrícula se divide en áreas que se definen como cadenas de texto.
- Cada nombre de área representa una región de la cuadrícula y puede estar formada por una o varias celdas.
- Los nombres se asignan a los elementos hijos mediante la propiedad `grid-area`, ubicándolos en la cuadrícula según el diseño especificado en `grid-template-areas`.

Ejercicio

Cree html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Grid Areas</title>
    <link rel="stylesheet" href="estilos.css">
</head>
<body class="grid">

    <div class="grid__item header">Header</div>
    <div class="grid__item main">Main</div>
    <div class="grid__item sidebar">Sidebar</div>
    <div class="grid__item footer">Footer</div>

</body>
</html>
```

cree css llamado estilos

```
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body{
    font-family: 'Arial';
}

.grid{
    height: 100vh;
    display: grid;
}

.grid__item{
    font-size: 2rem;
    color: #fff;
```

```

        display: flex;
        align-items: center;
        justify-content: center;
    }

.header{
    background: crimson;
}

.main{
    background: steelblue;
}

.sidebar{
    background: purple;
}

.footer{
    background: chocolate;
}

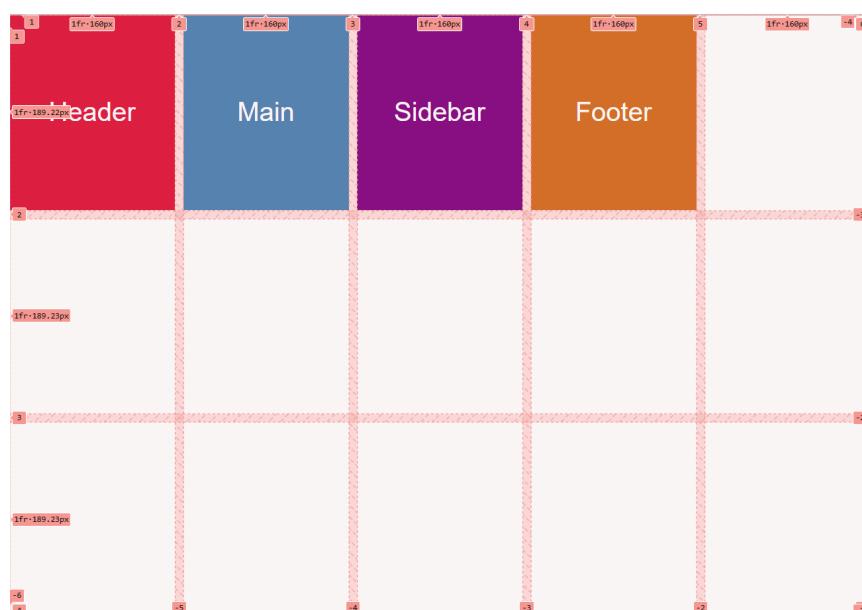
```

Modifique el css agregue las siguientes reglas

```

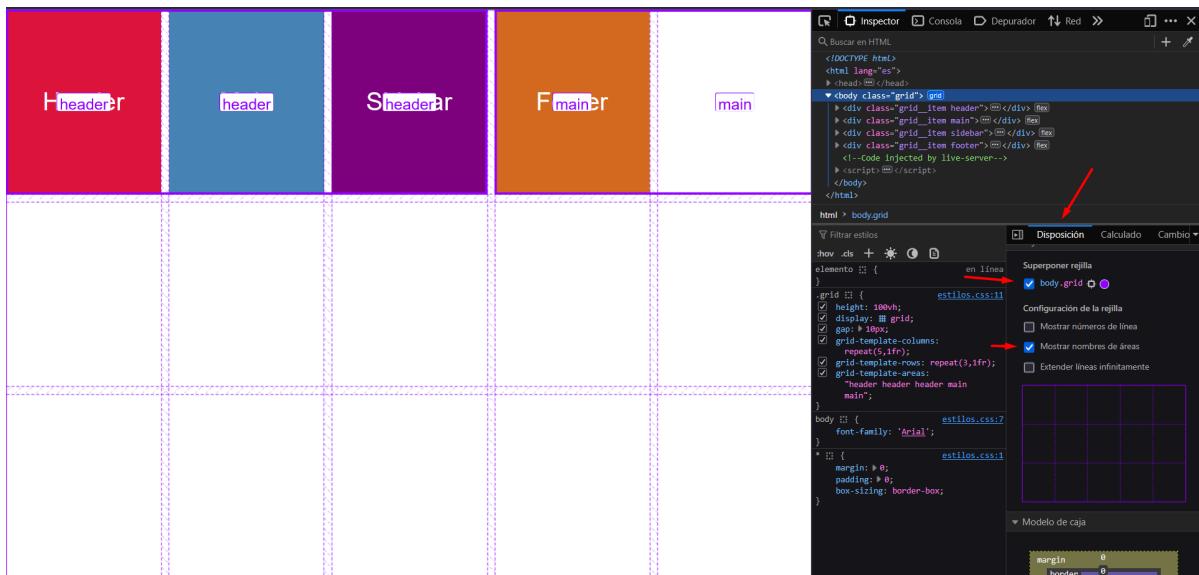
.grid{
    height: 100vh;
    display: grid;
    gap: 10px;
    grid-template-columns:repeat(5,1fr) ;
    grid-template-rows: repeat(5,1fr);
}

```



Definiendo areas

```
11   .grid{  
12  
13     height: 100vh;  
14     display: grid;  
15     gap: 10px;  
16     grid-template-columns:repeat(5,1fr) ;  
17     grid-template-rows: repeat(3,1fr);  
18     grid-template-areas:  
19       "header header header main main"  
20  
21 }
```



grid-area

La propiedad `grid-area` en CSS Grid tiene dos usos principales:

1. **Asignar nombres a áreas específicas dentro de la cuadrícula** (cuando se utiliza junto con `grid-template-areas`).
2. **Definir simultáneamente las líneas de inicio y fin de un elemento en la cuadrícula**, especificando su ubicación y tamaño en términos de filas y columnas.

Asignar nombres a áreas específicas de una cuadrícula

En este caso, `grid-area` se usa junto con `grid-template-areas` para ubicar elementos en áreas de la cuadrícula que se han nombrado previamente. Esto facilita la organización visual de los elementos.

modifique el css

```
*{  
  margin: 0;  
  padding: 0;  
  box-sizing: border-box;  
}
```

```
body{
    font-family: 'Arial';
}

.grid{
    height: 100vh;
    display: grid;
    gap: 5px;
    grid-template-columns:repeat(5,1fr) ;
    grid-template-rows: repeat(5,1fr);
    grid-template-areas:
        "header header header header header"
        "main main main main sidebar"
        "main main main main sidebar"
        "main main main main sidebar"
        "footer footer footer footer footer ";
}

.grid__item{
    font-size: 2rem;
    color: #fff;
    display: flex;
    align-items: center;
    justify-content: center;
}

.header{
    background: crimson;
    grid-area: header;
}

.main{
    background: steelblue;
    grid-area: main;
}

.sidebar{
    background: purple;
    grid-area: sidebar;
}

.footer{
    background: chocolate;
    grid-area: footer;
}

@media screen and (max-width:800px){
    .grid{
        grid-template-areas:
        "header header header header header"
        "main main main sidebar sidebar"
        "main main main sidebar sidebar"
        "main main main sidebar sidebar"
        "footer footer footer footer footer ";
    }
}
```

```
}

@media screen and (max-width:500px){
  .grid{
    grid-template-columns: 1fr;
    grid-template-rows: repeat(4,1fr);
    grid-template-areas:
      "header"
      "main"
      "sidebar"
      "footer";
  }
}
```