

Unidad 2

CSS (Cascading Style Sheets)

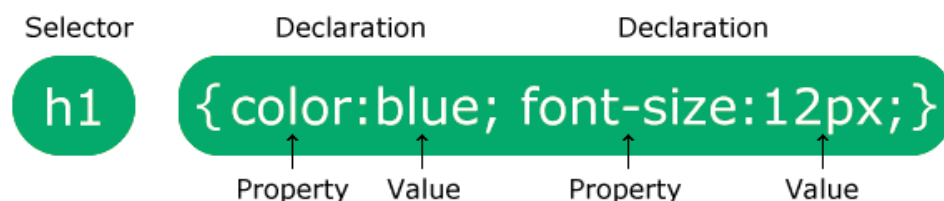
Introducción a CSS

CSS (Cascading Style Sheets) es un lenguaje de estilo utilizado para describir cómo se ve y se presenta un documento HTML. Permite controlar el diseño, el formato y la apariencia de múltiples páginas web a la vez, lo que lo convierte en una herramienta poderosa para los desarrolladores web. Al separar la presentación del contenido, CSS facilita la creación de sitios web atractivos y visualmente coherentes.

Al utilizar reglas CSS, los desarrolladores pueden especificar propiedades como el color, el tamaño y la ubicación de los elementos HTML, lo que proporciona flexibilidad y personalización. Además, CSS permite la creación de diseños responsivos que se adaptan a diferentes dispositivos y tamaños de pantalla.

CSS Syntax

La sintaxis de CSS (Hojas de Estilo en Cascada) consta de un selector, seguido por un conjunto de declaraciones encerradas entre llaves. Cada declaración incluye una propiedad y su valor correspondiente, separados por dos puntos. Múltiples declaraciones se separan por puntos y comas.



Ejemplo:

```
p {  
  color: red;  
  text-align: center;  
}
```

p es un selector en CSS (apunta al elemento HTML que deseas estilizar: <p>).

color es una propiedad, y **red** es el valor de la propiedad

text-align es una propiedad, y **center** es el valor de la propiedad

CSS Selectors

Los selectores CSS son uno de los conceptos fundamentales en la estilización de páginas web. Permiten a los desarrolladores dirigirse a elementos específicos en un documento HTML y aplicar estilos a esos elementos de manera selectiva.

Selector de Tipo: Este selector apunta a elementos HTML específicos por su nombre de etiqueta. Por ejemplo, el selector `p` se aplica a todos los elementos del párrafo `<p>` en el documento.

```
p {  
  text-align: center;  
  color: red;  
}
```

Selector de Clase: Los selectores de clase se utilizan para apuntar a elementos que tienen un atributo `class` específico. Se escriben precedidos por un punto (`.`). Por ejemplo, `.titulo` se aplicaría a todos los elementos que tienen la clase "titulo".

```
.center {  
  text-align: center;  
  color: red;  
}
```

Selector de ID: Similar al selector de clase, pero se dirige a elementos con un atributo `id` específico. Se escriben precedidos por un símbolo de almohadilla (`#`). Por ejemplo, `#encabezado` se aplicaría al elemento con el ID "encabezado".

```
#para1 {  
  text-align: center;  
  color: red;  
}
```

Selector Universal: Este selector, representado por un asterisco (`*`), selecciona todos los elementos en el documento.

```
* {  
  text-align: center;  
  color: blue;  
}
```

Selector de Atributo: Selecciona elementos que tienen un atributo específico, independientemente de su valor. Por ejemplo, `[type]` seleccionaría todos los elementos con un atributo `type`, como los elementos `<input>`.

Ejemplo:

Supongamos que tienes una página web con varios enlaces, algunos de los cuales tienen un atributo `target="_blank"` para abrir el enlace en una nueva pestaña del navegador. Quieres aplicar un estilo específico a esos enlaces que abren en una nueva pestaña.

```
<a href="https://www.ejemplo.com" target="_blank">Enlace 1</a>
<a href="https://www.otroejemplo.com">Enlace 2</a>
```

```
a[target="_blank"] {
  color: blue;
  font-weight: bold;
}
```

En el CSS, `a[target="_blank"]` es el Selector de Atributo. Selecciona todos los elementos `<a>` que tienen un atributo `target` con el valor `"_blank"`.

En este caso, se aplica un estilo de color azul y negrita a esos enlaces que tienen el atributo `target="_blank"`. Los enlaces que no tienen este atributo no se verán afectados por estas reglas de estilo.

Selector de Atributo y Valor: Este selector apunta a elementos que tienen un atributo específico con un valor específico. Por ejemplo, `[type="button"]` seleccionaría solo los elementos `<input>` con un atributo `type` igual a `"button"`.

Ejemplo:

Supongamos que tienes una lista de elementos `<input>` en tu formulario y quieres aplicar un estilo específico a los elementos que tienen el atributo `type` con el valor `"text"`.

```
<input type="text" placeholder="Nombre">
<input type="password" placeholder="Contraseña">
<input type="email" placeholder="Correo electrónico">
```

```
input[type="text"] {
  border: 2px solid green;
}
```

En el CSS, `input[type="text"]` es el Selector de Atributo y Valor. Selecciona todos los elementos `<input>` que tienen un atributo `type` con el valor `"text"`.

En este caso, se aplica un borde sólido verde a los elementos de entrada que son campos de texto. Los otros elementos de entrada que no tienen el atributo `type` con el valor `"text"` no se verán afectados por estas reglas de estilo.

Selector de Descendencia: Selecciona un elemento que es descendiente directo de otro elemento. Se utiliza con un espacio entre los selectores. Por ejemplo, `div p` seleccionaría todos los elementos de párrafo que son descendientes directos de elementos `<div>`.

Ejemplo:

Supongamos que tienes una estructura HTML con una lista ordenada () que contiene varias listas desordenadas () como elementos secundarios. Quieres aplicar un estilo diferente a los elementos de lista dentro de cada lista ordenada.

```
<ol>
  <li>Elemento de lista 1
    <ul>
      <li>Elemento de lista anidado 1</li>
      <li>Elemento de lista anidado 2</li>
    </ul>
  </li>
  <li>Elemento de lista 2</li>
</ol>
```

```
ol li {
  color: blue;
}
```

```
ol li ul li {
  color: red;
}
```

- En el CSS, ol li selecciona todos los elementos de lista () que son descendientes directos de una lista ordenada (). En este caso, se les aplica un color azul.
- ol li ul li selecciona todos los elementos de lista () que son descendientes directos de una lista desordenada () que, a su vez, es un descendiente directo de una lista ordenada (). En este caso, se les aplica un color rojo.
- Esto significa que los elementos de lista anidados dentro de una lista ordenada tendrán un color rojo, mientras que los elementos de lista que son hijos directos de la lista ordenada tendrán un color azul.

Selector de Clase Anidado: Selecciona elementos que son descendientes de un elemento con una clase específica. Se utiliza con un punto (.) antes de la clase deseada. Por ejemplo, .contenedor .titulo seleccionaría todos los elementos con la clase "titulo" que son descendientes de elementos con la clase "contenedor".

Ejemplo:

Supongamos que tienes una estructura HTML con una lista de elementos <div> que contienen elementos <p> y , y quieres aplicar estilos diferentes a los elementos <p> y que están dentro de las clases .contenedor1 y .contenedor2.

```
<div class="contenedor1">
  <p>Texto dentro de contenedor1</p>
  <span>Texto dentro de contenedor1</span>
</div>
```

```
<div class="contenedor2">
  <p>Texto dentro de contenedor2</p>
  <span>Texto dentro de contenedor2</span>
</div>
```

```
.contenedor1 p {  
  color: blue;  
}
```

```
.contenedor2 span {  
  color: red;  
}
```

- En el CSS, `.contenedor1 p` selecciona todos los elementos `<p>` que son descendientes directos de un elemento con la clase `.contenedor1`. En este caso, se les aplica un color azul.
- `.contenedor2 span` selecciona todos los elementos `` que son descendientes directos de un elemento con la clase `.contenedor2`. En este caso, se les aplica un color rojo.
- Esto significa que los párrafos dentro de `.contenedor1` serán azules, mientras que los spans dentro de `.contenedor2` serán rojos.

Formas comunes de insertar CSS en un documento HTML

Estilo en línea (Inline Style):

Esta técnica implica agregar estilos directamente a un elemento HTML utilizando el atributo `style`. Por ejemplo:

```
<!DOCTYPE html>  
<html>  
<body>  
  
<h1 style="color:blue;text-align:center;">This is a heading</h1>  
<p style="color:red;">This is a paragraph.</p>  
  
</body>  
</html>
```

Los estilos se aplican sólo al elemento específico y tienen la mayor prioridad, lo que significa que anulan cualquier estilo definido en hojas de estilo externas o internas.

Estilo interno (Internal Style):

Con esta técnica, los estilos se definen dentro del elemento `<style>` en el encabezado `<head>` del documento HTML. Por ejemplo:

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
body {  
  background-color: linen;  
}  
  
h1 {
```

```

    color: maroon;
    margin-left: 40px;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>

```

Los estilos definidos de esta manera se aplican a todos los elementos que coinciden con el selector dentro del documento HTML.

Estilo externo (External Style):

En esta técnica, los estilos se almacenan en un archivo de hoja de estilo separado (con extensión .css) y se vinculan al documento HTML utilizando la etiqueta <link>. Por ejemplo:

html

```

<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="mystyle.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>

```

mystyle.css

```

body {
    background-color: lightblue;
}

h1 {
    color: navy;
    margin-left: 20px;
}

```

El archivo de estilo externo (styles.css) contiene todas las reglas de estilo y se puede vincular a múltiples páginas HTML. Esto promueve la reutilización de estilos y facilita la actualización y mantenimiento del diseño de un sitio web.

Orden de Cascada

¿Qué estilo se utilizará cuando se especifiquen más de un estilo para un elemento HTML?

Todos los estilos en una página se "cascadearán" en una nueva hoja de estilo "virtual" según las siguientes reglas, donde el número uno tiene la mayor prioridad:

- Estilo en línea (dentro de un elemento HTML)
- Hojas de estilo externas e internas (en la sección de encabezado)
- Predeterminado del navegador

Por lo tanto, un estilo en línea tiene la prioridad más alta y anulará los estilos externos e internos y los predeterminados del navegador.

CSS Comments

Los comentarios en CSS son una forma de documentar y explicar el código para facilitar su comprensión y mantenimiento. Se pueden utilizar para dejar notas, explicaciones o descripciones dentro del código CSS sin que afecten al diseño final de la página web.

Los comentarios en CSS se pueden crear de dos maneras:

Comentarios de una sola línea: Se inician con `//` y todo lo que le sigue en esa línea se considera comentario. Por ejemplo:

```
/* Este es un comentario de una sola línea */
```

Comentarios de varias líneas: Se inician con `/*` y finalizan con `*/`. Pueden abarcar múltiples líneas de código CSS. Por ejemplo:

```
/*  
Este es un comentario  
de varias líneas  
*/
```

Es importante recordar que los comentarios en CSS no se renderizan en el navegador y no afectan el diseño o funcionamiento del sitio web, pero son útiles para los desarrolladores que trabajan en el código.

CSS Colors

Los colores en CSS son fundamentales para definir la apariencia visual de los elementos de una página web. Hay varias formas de especificar colores en CSS:

Nombres de colores: CSS proporciona un conjunto de nombres de colores predefinidos, como "red" (rojo), "blue" (azul), "green" (verde), entre otros.

Tomato	Orange	DodgerBlue	MediumSeaGreen
Gray	SlateBlue	Violet	LightGray

Ejemplo:
color: red;

Valores hexadecimales: Los colores se pueden especificar utilizando valores hexadecimales que representan la cantidad de rojo, verde y azul (RGB) en el color. Se escriben con un signo de numeral (#) seguido de seis caracteres hexadecimales.

Por ejemplo:

color: #ff0000; /* Rojo */

Valores RGB: Los colores también se pueden especificar utilizando valores RGB, que indican la cantidad de rojo, verde y azul en el color en una escala de 0 a 255. Por ejemplo:

color: rgb(255, 0, 0); /* Rojo */

Valores RGBA: Similar a RGB, pero con un cuarto parámetro que representa la opacidad (alfa) del color, que va desde 0 (completamente transparente) a 1 (completamente opaco).

Por ejemplo:

color: rgba(255, 0, 0, 0.5); /* Rojo semi-transparente */

Valores HSL y HSLA: HSL representa el tono (H), la saturación (S) y el nivel de luminosidad (L) del color. HSLA es lo mismo que HSL pero con un cuarto parámetro para la opacidad. Por ejemplo:

color: hsl(0, 100%, 50%); /* Rojo */

color: hsla(0, 100%, 50%, 0.5); /* Rojo semi-transparente */

Estos son solo algunos ejemplos de cómo se pueden especificar colores en CSS. La elección del método depende de las necesidades y preferencias del diseñador web.

Puedes consultar y armar tu paleta de colores en:

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

CSS Backgrounds

CSS (Cascading Style Sheets) Backgrounds son una parte esencial del diseño web, permitiendo a los desarrolladores definir y controlar la apariencia visual de los elementos en una página web. La propiedad background en CSS es versátil y permite una amplia gama de posibilidades estilísticas, desde colores simples hasta imágenes complejas. A continuación, se detallan algunas de las propiedades más comunes y su uso:

Propiedades Básicas de Fondo en CSS

background-color: Define el color de fondo de un elemento. Se puede especificar en varios formatos, incluyendo nombres de colores (como "red"), valores hexadecimales (como #ff0000), valores RGB (como rgb(255, 0, 0)) y valores HSL (como hsl(0, 100%, 50%)).

```
body {  
  background-color: #f0f0f0;  
}
```

background-image: Permite establecer una imagen de fondo para un elemento. La imagen puede ser una URL de una imagen externa o un data URI.

```
.hero {  
  background-image: url('hero-bg.jpg');  
}
```

background-repeat: Controla la repetición de la imagen de fondo. Los valores comunes incluyen repeat (repite la imagen tanto horizontal como verticalmente), repeat-x (repite solo horizontalmente), repeat-y (repite solo verticalmente) y no-repeat (no repite la imagen).

```
.pattern {  
  background-image: url('pattern.png');  
  background-repeat: repeat-x;  
}
```

background-position: Establece la posición inicial de la imagen de fondo. Puede usar palabras clave (como top, right, bottom, left, center), valores porcentuales o valores específicos en píxeles.

```
.banner {  
  background-image: url('banner.jpg');  
  background-position: center center;  
}
```

background-size: Define el tamaño de la imagen de fondo. Los valores pueden ser específicos (como 100px), porcentuales (como 50% 50%), cover (ajusta la imagen para cubrir el contenedor), y contain (ajusta la imagen para que quepa completamente dentro del contenedor).

```
.cover-bg {  
  background-image: url('cover.jpg');  
  background-size: cover;  
}
```

background-attachment: Controla el comportamiento de desplazamiento de la imagen de fondo. Los valores incluyen scroll (la imagen de fondo se desplaza con el contenido), fixed (la imagen de fondo se fija en su posición y no se desplaza con el contenido), y local (la imagen se desplaza con el contenido del elemento de desplazamiento).

```
.fixed-bg {  
  background-image: url('fixed-bg.jpg');  
  background-attachment: fixed;  
}
```

Opacity / Transparency

La propiedad **opacity** especifica la opacidad/transparencia de un elemento. Puede tomar un valor de 0.0 a 1.0. Cuanto más bajo sea el valor, más transparente será el elemento:



```
div {  
  background-color: green;  
  opacity: 0.3;  
}
```

Transparency using RGBA

Si no deseas aplicar opacidad a los elementos hijos, como en nuestro ejemplo anterior, usa valores de color RGBA. El siguiente ejemplo establece la opacidad para el color de fondo y no para el texto:



```
div {  
  background: rgba(0, 128, 0, 0.3) /* Green background with 30%  
  opacity */  
}
```

CSS Borders

Los bordes en CSS (Cascading Style Sheets) son una parte fundamental del diseño web, permitiendo a los desarrolladores definir el contorno de los elementos en una página. La propiedad `border` y sus propiedades asociadas proporcionan una gran flexibilidad para estilizar los bordes de los elementos de diversas maneras. A continuación, se describen las propiedades más comunes y su uso:

border: Es una propiedad compuesta que permite definir el ancho, el estilo y el color del borde en una sola línea. Por ejemplo:

```
.box {  
  border: 1px solid #000;  
}
```

border-width: Especifica el grosor del borde. Puede definirse en píxeles, ems, o palabras clave como `thin`, `medium`, y `thick`.

```
.box {  
  border-width: 2px;  
}
```

CSS Border Style

border-style: Define el estilo del borde. Los valores posibles incluyen:

```
p.dotted {border-style: dotted;}  
p.dashed {border-style: dashed;}  
p.solid {border-style: solid;}  
p.double {border-style: double;}  
p.groove {border-style: groove;}  
p.ridge {border-style: ridge;}  
p.inset {border-style: inset;}  
p.outset {border-style: outset;}  
p.none {border-style: none;}  
p.hidden {border-style: hidden;}  
p.mix {border-style: dotted dashed solid double;}
```

Result:

A dotted border.

A dashed border.

A solid border.

A double border.

A groove border. The effect depends on the border-color value.

A ridge border. The effect depends on the border-color value.

An inset border. The effect depends on the border-color value.

An outset border. The effect depends on the border-color value.

No border.

A hidden border.

A mixed border.

CSS Rounded Borders

```
p {  
  border: 2px solid red;  
  border-radius: 5px;  
}
```

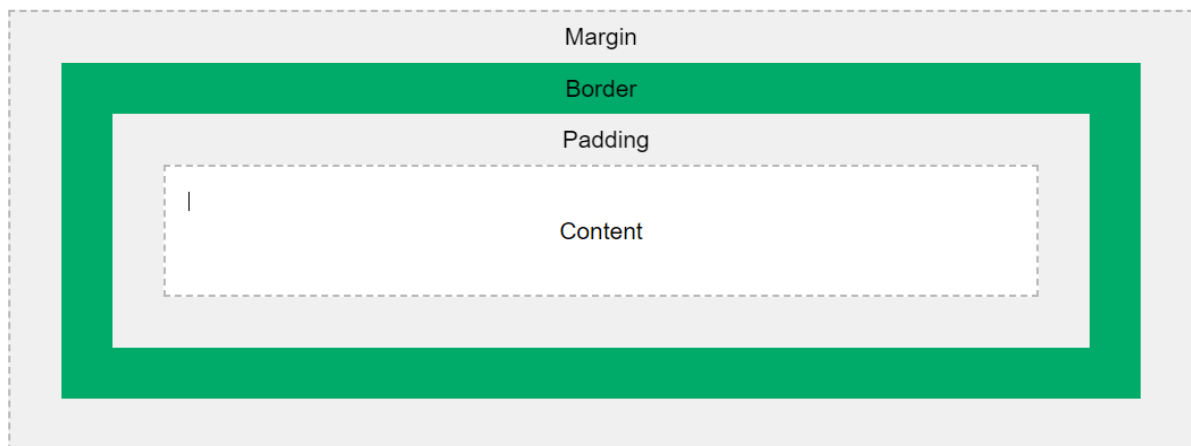
Normal border

Round border

Rounder border

Roundest border

Modelo de caja



El Modelo de Caja en CSS (CSS Box Model) es un concepto fundamental que describe cómo se componen y se disponen los elementos en una página web. Cada elemento en CSS se representa como una caja rectangular que incluye su contenido, relleno (padding), borde (border) y margen (margin). Comprender el Modelo de Caja es esencial para diseñar y maquetar correctamente las páginas web.

CSS Margins

Los márgenes en CSS (CSS Margins) permiten controlar el espacio exterior alrededor de los elementos, separándolos de otros elementos y del borde del contenedor. A continuación, se detallan las propiedades relacionadas con los márgenes y cómo se utilizan en CSS.

margin: Es una propiedad abreviada que permite definir los márgenes en los cuatro lados de un elemento (superior, derecho, inferior e izquierdo) en una sola declaración. Se pueden usar varios valores:

- Un valor: se aplica a todos los lados.
- Dos valores: el primer valor se aplica a los márgenes superior e inferior, y el segundo a los márgenes izquierdo y derecho.
- Tres valores: el primer valor se aplica al margen superior, el segundo a los márgenes izquierdo y derecho, y el tercero al margen inferior.
- Cuatro valores: se aplican en el orden superior, derecho, inferior e izquierdo.

```
.box {  
  margin: 10px; /* Un valor */  
  margin: 10px 20px; /* Dos valores */  
  margin: 10px 20px 30px; /* Tres valores */  
  margin: 10px 20px 30px 40px; /* Cuatro valores */  
}
```

margin-top, margin-right, margin-bottom, margin-left: Estas propiedades permiten especificar los márgenes de manera individual para cada lado del elemento.

```
.box {  
  margin-top: 10px;  
  margin-right: 20px;  
  margin-bottom: 30px;  
  margin-left: 40px;  
}
```

Los valores de los márgenes pueden ser especificados en diferentes unidades, incluyendo **píxeles (px)**, **porcentajes (%)**, **ems (em)**, **rem (rem)**, entre otros. También se pueden usar valores especiales como auto y valores negativos.

Valores en píxeles (px): Unidades fijas que no cambian con el tamaño de la pantalla.

```
.box {  
  margin: 20px;  
}
```

Valores negativos: Permiten reducir el espacio alrededor de un elemento, haciendo que se superponga con otros elementos.

```
.box {  
  margin: -10px;  
}
```

CSS Padding

CSS (CSS Padding) es una propiedad fundamental que se utiliza para controlar el espacio interior entre el contenido de un elemento y su borde. A diferencia de los márgenes, que afectan el espacio exterior del elemento, el relleno se aplica dentro del contenedor del elemento, influyendo en cómo se presenta el contenido en relación con su borde. A continuación, se describen las propiedades relacionadas con el relleno y su uso en CSS.

Propiedades Básicas de Padding en CSS

padding: Es una propiedad abreviada que permite definir el relleno en los cuatro lados de un elemento (superior, derecho, inferior e izquierdo) en una sola declaración. Los valores pueden especificarse de la siguiente manera:

- Un valor: se aplica a todos los lados.
- Dos valores: el primer valor se aplica a los rellenos superior e inferior, y el segundo a los rellenos izquierdo y derecho.
- Tres valores: el primer valor se aplica al relleno superior, el segundo a los rellenos izquierdo y derecho, y el tercero al relleno inferior.
- Cuatro valores: se aplican en el orden superior, derecho, inferior e izquierdo.

```
.box {
  padding: 10px; /* Un valor */
  padding: 10px 20px; /* Dos valores */
  padding: 10px 20px 30px; /* Tres valores */
  padding: 10px 20px 30px 40px; /* Cuatro valores */
}
```

padding-top, padding-right, padding-bottom, padding-left: Estas propiedades permiten especificar el relleno de manera individual para cada lado del elemento.

```
.box {
  padding-top: 10px;
  padding-right: 20px;
  padding-bottom: 30px;
  padding-left: 40px;
}
```

CSS Setting height and width

Las propiedades height y width se utilizan para establecer la altura y el ancho de un elemento.

Las propiedades height y width no incluyen el relleno (padding), los bordes (borders) ni los márgenes (margins). Estas propiedades establecen la altura/el ancho del área dentro del relleno, el borde y el margen del elemento.

```
div {
  height: 200px;
  width: 50%;
  background-color: powderblue;
}
```

Ejemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color: lightgrey;
  width: 300px;
  border: 15px solid green;
  padding: 50px;
  margin: 20px;
}
```

```
</style>
</head>
<body>
```

```
<h2>Demonstrating the Box Model</h2>
```

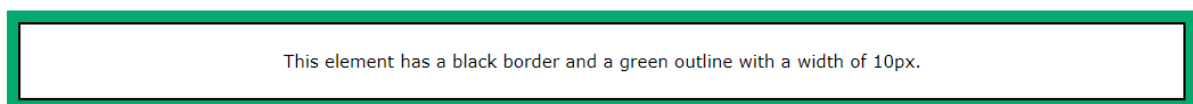
```
<p>The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.</p>
```

```
<div>This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border.</div>
```

```
</body>
</html>
```

CSS Outline

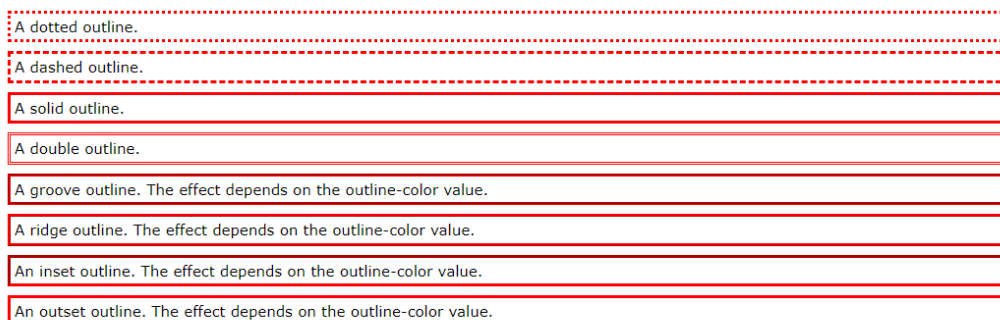
Un contorno es una línea dibujada fuera del borde del elemento.



```
p {
  border: 2px solid black;
  outline: #4CAF50 solid 10px;
  margin: auto;
  padding: 20px;
  text-align: center;
}
```

```
p.dotted {outline-style: dotted;}
p.dashed {outline-style: dashed;}
p.solid {outline-style: solid;}
p.double {outline-style: double;}
p.groove {outline-style: groove;}
p.ridge {outline-style: ridge;}
p.inset {outline-style: inset;}
p.outset {outline-style: outset;}
```

Result:



Diseño de página y posicionamiento.

Recuerda que el elemento `<div>` se utiliza frecuentemente como contenedor para agrupar secciones de una página web juntas.

Ejemplo:

<pre><!DOCTYPE html> <html> <style> div { background-color: #FFF4A3; } </style> <body> <h1>HTML DIV Example</h1> <div> <h2>London</h2> <p>London is the capital city of England.</p> <p>London has over 13 million inhabitants.</p> </div> <p>The yellow background is added to demonstrate the footprint of the DIV element.</p> </body> </html></pre>	<h3>HTML DIV Example</h3> <h4>London</h4> <p>London is the capital city of England.</p> <p>London has over 13 million inhabitants.</p> <p>The yellow background is added to demonstrate the footprint of the DIV element.</p>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Para alinear centrado un elemento `<div>` que no ocupa el ancho completo de la pantalla, establece la propiedad CSS `margin` en auto.

<pre><!DOCTYPE html> <html> <style> div { width: 300px; margin: auto; background-color: #FFF4A3; } </style> <body> <h1>Center align a DIV element</h1> <div> <h2>London</h2> <p>London is the capital city of England.</p> <p>London has over 13 million inhabitants.</p> </div> </body> </html></pre>	<h3>Center align a DIV element</h3> <h4>London</h4> <p>London is the capital city of England.</p> <p>London has over 13 million inhabitants.</p>
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------

Puedes tener varios contenedores `<div>` en la misma página.

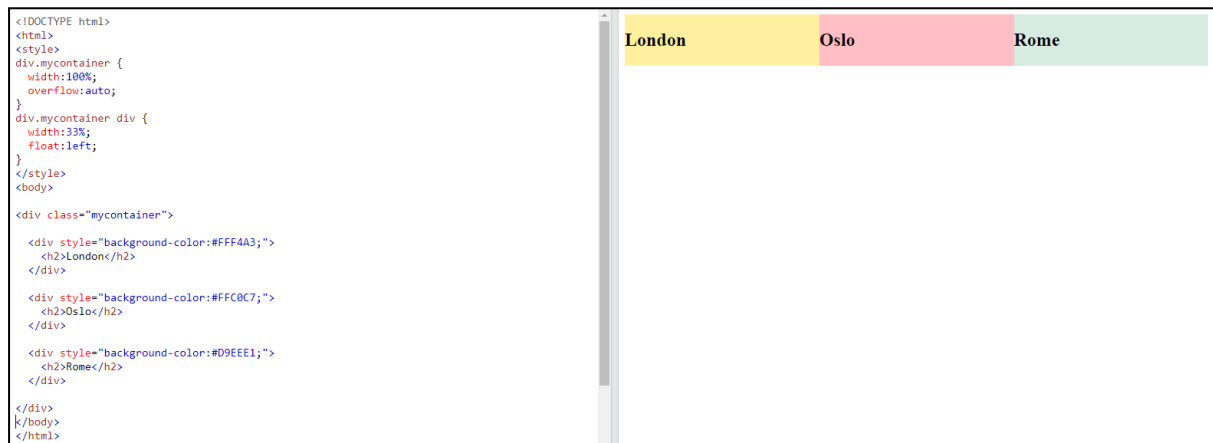
<pre><!DOCTYPE html> <html> <body> <h1>Multiple DIV Elements</h1> <div style="background-color:#FFF4A3;"> <h2>London</h2> <p>London is the capital city of England.</p> <p>London has over 13 million inhabitants.</p> </div> <div style="background-color:#FFC0C7;"> <h2>Oslo</h2> <p>Oslo is the capital city of Norway.</p> <p>Oslo has over 600.000 inhabitants.</p> </div> <div style="background-color:#D9EEE1;"> <h2>Rome</h2> <p>Rome is the capital city of Italy.</p> <p>Rome has almost 3 million inhabitants.</p> </div> <p>CSS styles are added to make it easier to separate the divs, and to make them more pretty:</p> </body> </html></pre>	<h3>Multiple DIV Elements</h3> <h4>London</h4> <p>London is the capital city of England.</p> <p>London has over 13 million inhabitants.</p> <h4>Oslo</h4> <p>Oslo is the capital city of Norway.</p> <p>Oslo has over 600.000 inhabitants.</p> <h4>Rome</h4> <p>Rome is the capital city of Italy.</p> <p>Rome has almost 3 million inhabitants.</p> <p>CSS styles are added to make it easier to separate the divs, and to make them more pretty:)</p>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Alinear elementos `<div>` uno al lado del otro es una práctica común al diseñar páginas web. Hay diferentes métodos para lograrlo, todos requieren un poco de estilizado CSS. Vamos a explorar los métodos más comunes.

Float:

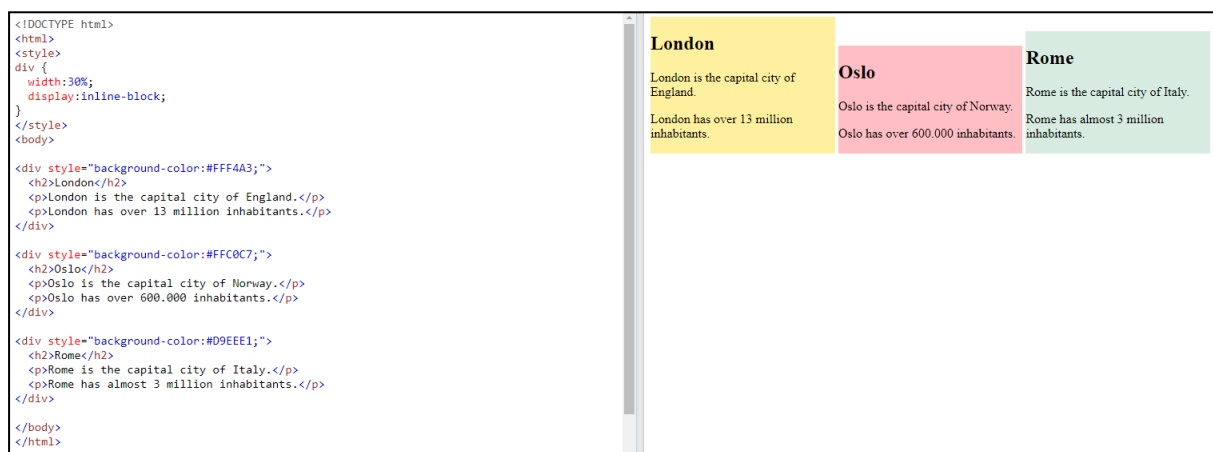
La propiedad CSS float originalmente no estaba destinada a alinear elementos <div> uno al lado del otro, pero se ha utilizado para este propósito durante muchos años.

La propiedad CSS float se utiliza para posicionar y formatear contenido y permite que los elementos floten uno al lado del otro en lugar de uno encima del otro.



Inline-block:

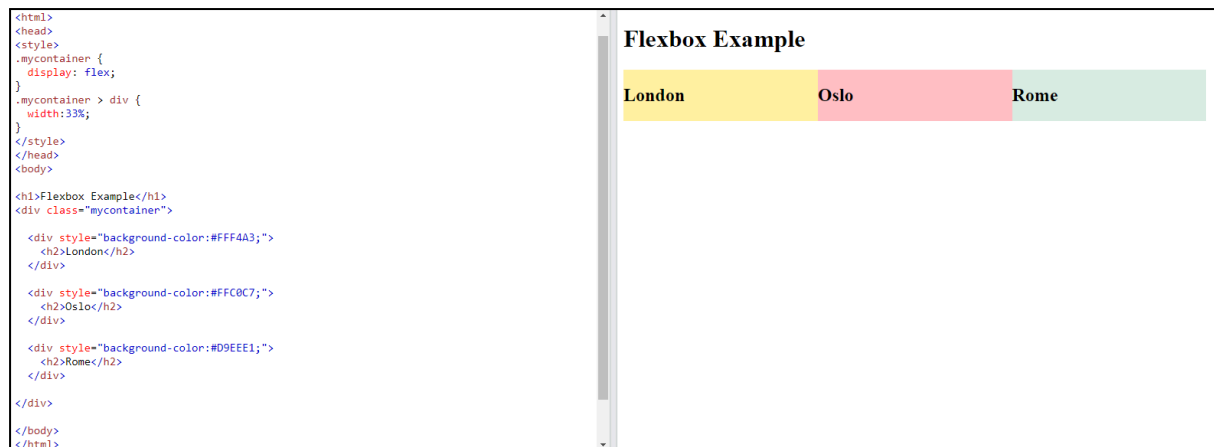
Si cambias la propiedad de visualización (display) del elemento <div> de bloque (block) a en línea-bloque (inline-block), los elementos <div> ya no añadirán un salto de línea antes y después, y se mostrarán uno al lado del otro en lugar de uno encima del otro.



Flex:

El Módulo de Diseño Flexbox en CSS fue introducido para facilitar el diseño de estructuras de diseño flexibles y responsivas sin necesidad de utilizar flotantes o posicionamiento.

Para hacer que el método flex en CSS funcione, rodea los elementos <div> con otro elemento <div> y dale el estado de un contenedor flex.

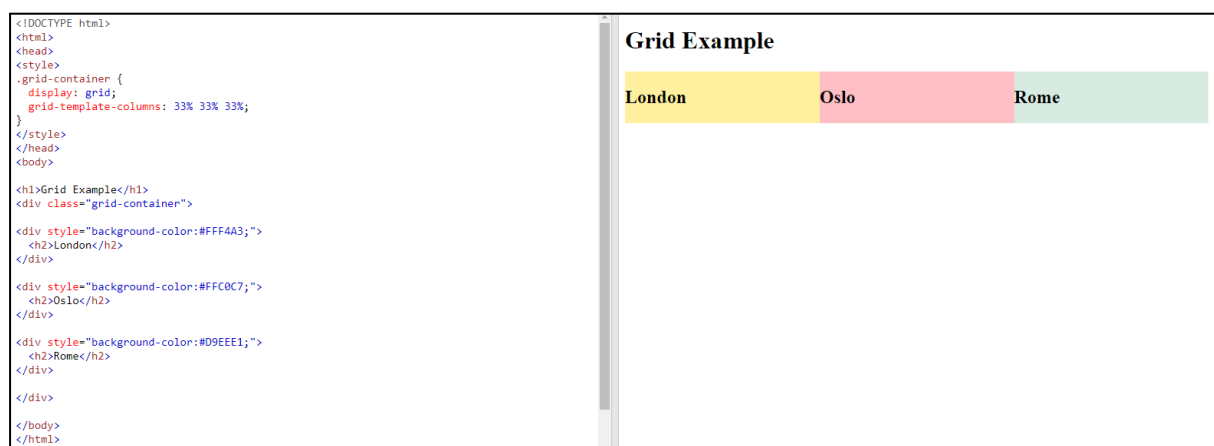


Grid:

El Módulo de Diseño de Rejilla en CSS ofrece un sistema de diseño basado en rejillas, con filas y columnas, lo que facilita el diseño de páginas web sin necesidad de utilizar flotantes y posicionamiento.

Suena casi igual que flex, pero tiene la capacidad de definir más de una fila y posicionar cada fila individualmente.

El método de rejilla en CSS requiere que rodees los elementos `<div>` con otro elemento `<div>` y le des el estado de un contenedor de rejilla, y debes especificar el ancho de cada columna.



CSS Flexbox y Grid

El Módulo de Diseño de Caja Flexible (Flexbox), facilita la creación de estructuras de diseño flexibles y responsivas sin necesidad de utilizar flotantes o posicionamiento.

Contenedor Flex (Flex Container): Para utilizar Flexbox, se debe crear un contenedor flex que envuelva a los elementos que se desean alinear y distribuir. Se logra estableciendo la propiedad `display` del contenedor como `flex` o `inline-flex`.

Elementos Flex (Flex Items): Los elementos contenidos dentro de un contenedor flex se denominan elementos flex. Estos elementos se distribuyen automáticamente dentro del contenedor flex de acuerdo con las reglas de Flexbox.

Ejemplo:



Propiedades Importantes de Flexbox

flex-direction: Controla la dirección en la que se colocan los elementos flex dentro del contenedor flex. Los valores comunes incluyen row, row-reverse, column y column-reverse.

justify-content: Alinea los elementos flex a lo largo del eje principal del contenedor flex. Esto controla el espacio entre los elementos cuando no ocupan todo el espacio disponible.

align-items: Alinea los elementos flex a lo largo del eje transversal del contenedor flex. Define cómo se distribuyen los elementos cuando no llenan todo el espacio disponible en la dirección transversal.

flex-grow, flex-shrink, flex-basis: Estas propiedades controlan cómo se distribuye el espacio entre los elementos flexibles cuando no tienen un ancho o alto fijo. flex-grow determina cuánto puede crecer un elemento, flex-shrink controla cuánto puede encogerse y flex-basis establece el tamaño base inicial.

Algunos ejemplos:

Order Property

```
<style>
.flex-container {
  display: flex;
  align-items: stretch;
  background-color: #f1f1f1;
}

.flex-container>div {
  background-color: DodgerBlue;
  color: white;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
</style>
</head>
<body>

<h1>The order Property</h1>

<p>Use the order property to sort the flex items as you like:</p>

<div class="flex-container">
  <div style="order: 3">1</div>
  <div style="order: 2">2</div>
  <div style="order: 4">3</div>
  <div style="order: 1">4</div>
</div>
```

The order Property

Use the order property to sort the flex items as you like:



Flex-grow Property

```
<style>
.flex-container {
  display: flex;
  align-items: stretch;
  background-color: #f1f1f1;
}

.flex-container > div {
  background-color: DodgerBlue;
  color: white;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
</style>
</head>
<body>

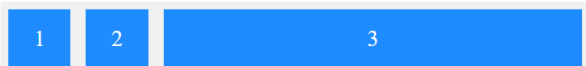
<h1>The flex-grow Property</h1>

<p>Make the third flex item grow eight times faster than the other flex items:</p>

<div class="flex-container">
  <div style="flex-grow: 1">1</div>
  <div style="flex-grow: 1">2</div>
  <div style="flex-grow: 8">3</div>
</div>
```

The flex-grow Property

Make the third flex item grow eight times faster than the other flex items:



The flex-shrink Property (La propiedad flex-shrink especifica cuánto se contraerá un elemento flex en relación con el resto de los elementos flex.)

```
<style>
.flex-container {
  display: flex;
  align-items: stretch;
  background-color: #f1f1f1;
}

.flex-container>div {
  background-color: DodgerBlue;
  color: white;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
</style>
</head>
<body>

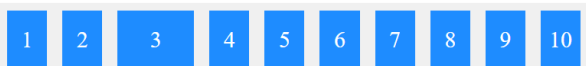
<h1>The flex-shrink Property</h1>

<p>Do not let the third flex item shrink as much as the other flex items:</p>

<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex-shrink: 0">3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
  <div>7</div>
  <div>8</div>
  <div>9</div>
  <div>10</div>
</div>
```

The flex-shrink Property

Do not let the third flex item shrink as much as the other flex items:



The flex-basis Property *(La propiedad flex-basis especifica la longitud inicial de un elemento flex.)*

```
<style>
.flex-container {
  display: flex;
  align-items: stretch;
  background-color: #f1f1f1;
}

.flex-container > div {
  background-color: DodgerBlue;
  color: white;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
</style>
</head>
<body>

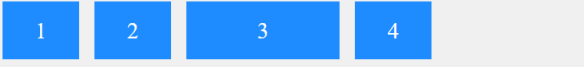
<h1>The flex-basis Property</h1>

<p>Set the initial length of the third flex item to 200 pixels:</p>

<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex-basis:200px">3</div>
  <div>4</div>
</div>
```

The flex-basis Property

Set the initial length of the third flex item to 200 pixels:



The align-self Property *(La propiedad align-self especifica la alineación para el elemento seleccionado dentro del contenedor flexible.)*

La propiedad align-self anula la alineación predeterminada establecida por la propiedad align-items del contenedor.)

```
<style>
.flex-container {
  display: flex;
  height: 200px;
  background-color: #f1f1f1;
}

.flex-container > div {
  background-color: DodgerBlue;
  color: white;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
</style>
</head>
<body>


<h1>The align-self Property</h1>

<p>The "align-self: center;" aligns the selected flex item in the middle of the container:
</p>

<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="align-self: center">3</div>
  <div>4</div>
</div>
```

The align-self Property

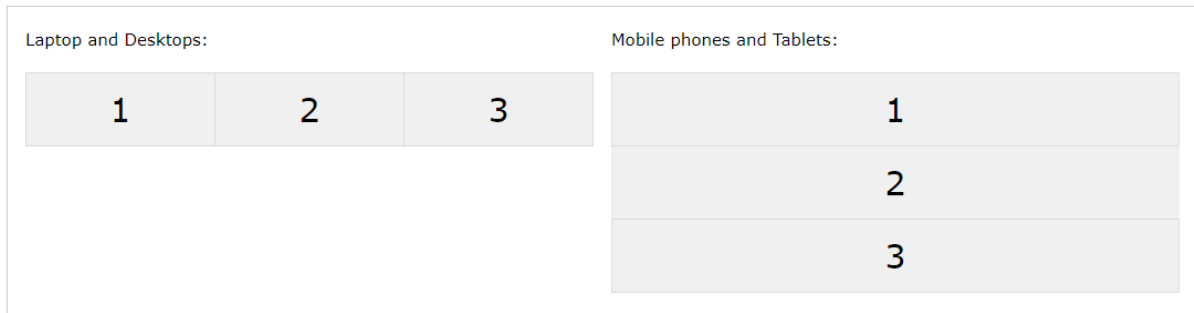
The "align-self: center;" aligns the selected flex item in the middle of the container:



The align-self property overrides the align-items property of the container.

Responsive Flexbox

Por ejemplo, si deseas crear un diseño de dos columnas para la mayoría de los tamaños de pantalla, y un diseño de una sola columna para tamaños de pantalla pequeños (como teléfonos y tabletas), puedes cambiar la dirección de flexión de fila (row) a columna (column) en un punto de interrupción específico (800px en el ejemplo siguiente):



Ejemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
* {
  box-sizing: border-box;
}

.flex-container {
  display: flex;
  flex-direction: row;
  font-size: 30px;
  text-align: center;
}

.flex-item-left {
  background-color: #f1f1f1;
  padding: 10px;
  flex: 50%;
}

.flex-item-right {
  background-color: dodgerblue;
  padding: 10px;
  flex: 50%;
}
```

```

/* Responsive layout - makes a one column-layout instead of two-column layout */
@media (max-width: 800px) {
  .flex-container {
    flex-direction: column;
  }
}
</style>
</head>
<body>

<h1>Responsive Flexbox</h1>

<p>The "flex-direction: row;" stacks the flex items horizontally (from left to right).</p>
<p>The "flex-direction: column;" stacks the flex items vertically (from top to bottom).</p>
<p><b>Resize the browser window to see that the direction changes when the
screen size is 800px wide or smaller.</b></p>

<div class="flex-container">
  <div class="flex-item-left">1</div>
  <div class="flex-item-right">2</div>
</div>

</body>
</html>

```

Recomendamos analizar el ejemplo:

https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_image_gallery

Grid Layout

El Módulo de Diseño de Rejilla en CSS ofrece un sistema de diseño basado en rejillas, con filas y columnas, lo que facilita el diseño de páginas web sin tener que utilizar flotantes y posicionamiento.

```

<style>
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto;
  background-color: #2196f3;
  padding: 10px;
}
.grid-item {
  background-color: rgba(255, 255, 255, 0.8);
  border: 1px solid rgba(0, 0, 0, 0.8);
  padding: 20px;
  font-size: 30px;
  text-align: center;
}
</style>
</head>
<body>

<h1>Grid Elements</h1>

<div class="grid-container">
  <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 class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>

```

Grid Elements

A Grid Layout must have a parent element with the *display* property set to *grid* or *inline-grid*.

Direct child element(s) of the grid container automatically becomes grid items.

1	2	3
4	5	6
7	8	9

La propiedad de visualización (Display Property)

Un elemento HTML se convierte en un contenedor de rejilla cuando su propiedad de visualización se establece en grid o inline-grid.

```
<style>
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto;
  background-color: #2196F3;
  padding: 10px;
}

.grid-item {
  background-color: rgba(255, 255, 255, 0.8);
  border: 1px solid rgba(0, 0, 0, 0.8);
  padding: 20px;
  font-size: 30px;
  text-align: center;
}
</style>
</head>
<body>

<div class="grid-container">
  <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 class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>
```

display: grid

Use display: grid; to make a block-level grid container:

1	2	3
4	5	6
7	8	9

```
<style>
.grid-container {
  display: inline-grid;
  grid-template-columns: auto auto auto;
  background-color: #2196F3;
  padding: 10px;
}

.grid-item {
  background-color: rgba(255, 255, 255, 0.8);
  border: 1px solid rgba(0, 0, 0, 0.8);
  padding: 20px;
  font-size: 30px;
  text-align: center;
}
</style>
</head>
<body>

<div class="grid-container">
  <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 class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>
```

display: inline-grid

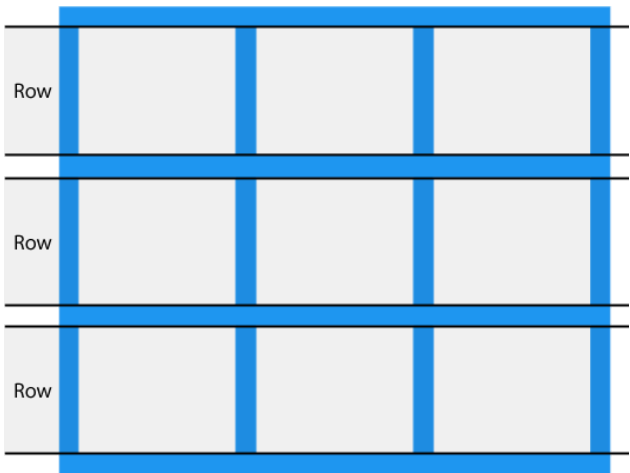
Use display: inline-grid; to make an inline grid container:

1	2	3
4	5	6
7	8	9

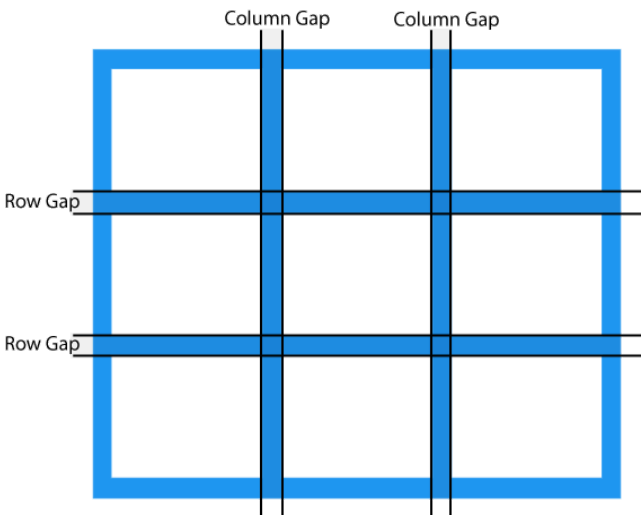
Grid Columns

Column	Column	Column

Grid Rows



Grid Gaps



Ejemplo:

```
<style>
.grid-container {
  display: grid;
  column-gap: 50px;
  grid-template-columns: auto auto auto;
  background-color: #2196F3;
  padding: 10px;
}

.grid-item {
  background-color: rgba(255, 255, 255, 0.8);
  border: 1px solid rgba(0, 0, 0, 0.8);
  padding: 20px;
  font-size: 30px;
  text-align: center;
}
</style>
</head>
<body>

<div class="grid-container">
  <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 class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>
```

The column-gap Property

Use the *column-gap* property to adjust the space between the columns:

1	2	3
4	5	6
7	8	9

Recursos Educativos:

(Aprender Flex) <https://flexboxfroggy.com/#es>

(Aprender Grid) <https://cssgridgarden.com/#es>

Media Queries y Diseño Adaptable (responsive).

¿Qué es el Diseño Web Responsivo?

- El diseño web responsivo hace que tu página web se vea bien en todos los dispositivos.
- El diseño web responsivo utiliza únicamente HTML y CSS.
- El diseño web responsivo no es un programa ni un JavaScript.



Viewport

El viewport es la ventana visible de un navegador web que muestra el contenido de una página web. En términos simples, es el área de la pantalla en la que se muestra el contenido de una página web. Esto puede variar según el dispositivo en el que se está visualizando la página web.

El tamaño del viewport puede cambiar dependiendo del dispositivo y de cómo el usuario esté interactuando con la página (por ejemplo, al hacer zoom). Para garantizar una experiencia de usuario consistente y adaptativa, especialmente en dispositivos móviles y tabletas, los desarrolladores web utilizan la etiqueta meta viewport en el HTML para controlar cómo se muestra y se escala el contenido en el viewport del dispositivo. Esto se hace estableciendo propiedades como width (ancho), initial-scale (escala inicial), minimum-scale (escala mínima), maximum-scale (escala máxima) y user-scalable (habilitar/deshabilitar el zoom del usuario).

Ejemplo:

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

Grid-View

Muchas páginas web se basan en una vista de rejilla, lo que significa que la página se divide en columnas.

Usar una vista de rejilla es muy útil al diseñar páginas web. Facilita la colocación de elementos en la página.

Construyendo una Vista de Rejilla Responsiva

Comencemos a construir una vista de rejilla responsiva.

Primero, asegúrate de que todos los elementos HTML tengan la propiedad `box-sizing` establecida en `border-box`. Esto garantiza que el relleno (`padding`) y el borde (`border`) estén incluidos en el ancho y alto total de los elementos.

Agrega el siguiente código en tu CSS:

```
* {  
  box-sizing: border-box;  
}
```

El siguiente ejemplo muestra una página web responsiva simple, con dos columnas:

25%	75%
-----	-----

Example

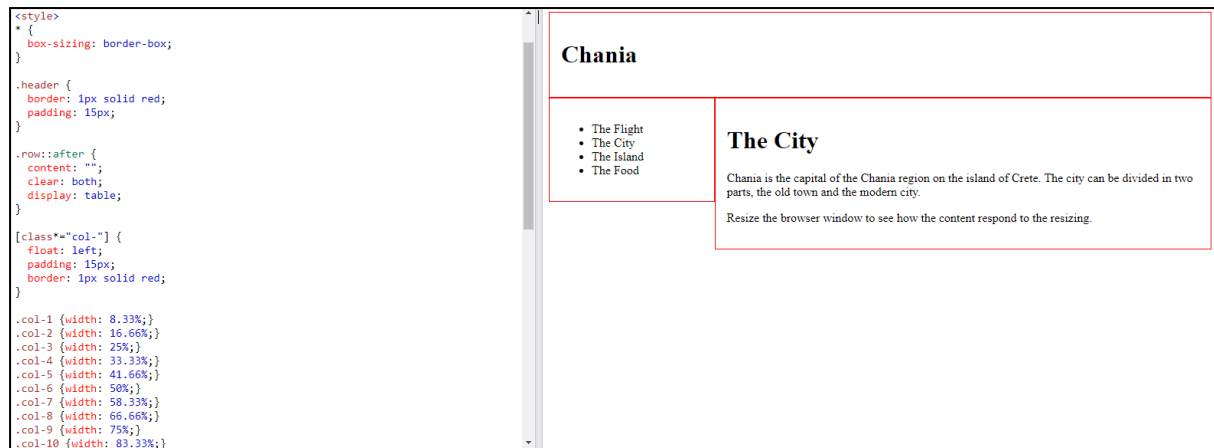
```
.menu {  
  width: 25%;  
  float: left;  
}  
.main {  
  width: 75%;  
  float: left;  
}
```

El ejemplo anterior está bien si la página web solo contiene dos columnas.

Sin embargo, queremos usar una vista de rejilla responsiva con 12 columnas, para tener más control sobre la página web.

Primero debemos calcular el porcentaje para una columna: $100\% / 12 \text{ columnas} = 8.33\%$.

Luego creamos una clase para cada una de las 12 columnas, `col-` y un número que define cuántas columnas debería abarcar la sección:



Todas estas columnas deben flotar hacia la izquierda y tener un relleno de 15px:

```
[class*="col-"] {
  float: left;
  padding: 15px;
  border: 1px solid red;
}
```

Cada fila debe estar envuelta en un <div>. El número de columnas dentro de una fila siempre debe sumar 12:

```
<div class="row">
  <div class="col-3">...</div> <!-- 25% -->
  <div class="col-9">...</div> <!-- 75% -->
</div>
```

Las columnas dentro de una fila están flotando hacia la izquierda, y por lo tanto se sacan del flujo de la página, y otros elementos se colocarán como si las columnas no existieran. Para evitar esto, agregaremos un estilo que limpie el flujo:

```
.row::after {
  content: "";
  clear: both;
  display: table;
}
```

Agregamos colores y estilos

```
html {
  font-family: "Lucida Sans", sans-serif;
}

.header {
```

```

    background-color: #9933cc;
    color: #ffffff;
    padding: 15px;
}

.menu ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
}

.menu li {
    padding: 8px;
    margin-bottom: 7px;
    background-color: #33b5e5;
    color: #ffffff;
    box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px
    rgba(0,0,0,0.24);
}

.menu li:hover {
    background-color: #0099cc;
}

```

Ejemplo completo:

```

<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
* {
    box-sizing: border-box;
}

.row::after {
    content: "";
    clear: both;
    display: table;
}

[class*="col-"] {
    float: left;
    padding: 15px;
}

.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}

```

```

.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}

html {
  font-family: "Lucida Sans", sans-serif;
}

.header {
  background-color: #9933cc;
  color: #ffffff;
  padding: 15px;
}

.menu ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}

.menu li {
  padding: 8px;
  margin-bottom: 7px;
  background-color: #33b5e5;
  color: #ffffff;
  box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
}

.menu li:hover {
  background-color: #0099cc;
}
</style>
</head>
<body>

<div class="header">
  <h1>Chania</h1>
</div>

<div class="row">
  <div class="col-3 menu">
    <ul>
      <li>The Flight</li>
      <li>The City</li>
      <li>The Island</li>
      <li>The Food</li>
    </ul>
  </div>

  <div class="col-9">
    <h1>The City</h1>
    <p>Chania is the capital of the Chania region on the island of Crete. The city can be
divided in two parts, the old town and the modern city.</p>
    <p>Resize the browser window to see how the content respond to the resizing.</p>
  </div>
</div>

</body>
</html>

```

Una parte fundamental del diseño responsive son las consultas de medios (media queries), que permiten aplicar estilos específicos según las características del dispositivo, como el ancho de la pantalla, la orientación y la resolución.

¿Qué son las Consultas de Medios?

Las consultas de medios son reglas CSS que permiten aplicar estilos basados en las características del dispositivo en el que se visualiza la página web. Se definen utilizando la sintaxis `@media`, seguida de una condición que especifica las características del dispositivo, como el ancho de la pantalla. Por ejemplo:

```
@media only screen and (max-width: 600px) {  
  body {  
    background-color: lightblue;  
  }  
}
```

Ejemplo:

Cuando la pantalla (ventana del navegador) se hace más pequeña que 768px, cada columna debería tener un ancho del 100%:

```
/* For desktop: */  
.col-1 {width: 8.33%;}  
.col-2 {width: 16.66%;}  
.col-3 {width: 25%;}  
.col-4 {width: 33.33%;}  
.col-5 {width: 41.66%;}  
.col-6 {width: 50%;}  
.col-7 {width: 58.33%;}  
.col-8 {width: 66.66%;}  
.col-9 {width: 75%;}  
.col-10 {width: 83.33%;}  
.col-11 {width: 91.66%;}  
.col-12 {width: 100%;}  
  
@media only screen and (max-width: 768px) {  
  /* For mobile phones: */  
  [class*="col-"] {  
    width: 100%;  
  }  
}
```

Puntos de Interrupción Típicos para Dispositivos

Hay una gran cantidad de pantallas y dispositivos con diferentes alturas y anchuras, por lo que es difícil crear un punto de

interrupción exacto para cada dispositivo. Para simplificar las cosas, podrías dirigirte a cinco grupos:

```
/* Extra small devices (phones, 600px and down) */
@media only screen and (max-width: 600px) {...}

/* Small devices (portrait tablets and large phones, 600px and up)
*/
@media only screen and (min-width: 600px) {...}

/* Medium devices (landscape tablets, 768px and up) */
@media only screen and (min-width: 768px) {...}

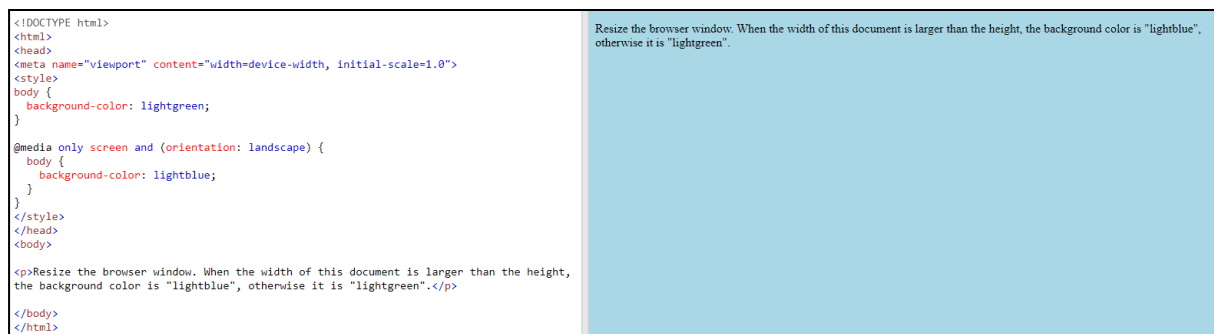
/* Large devices (laptops/desktops, 992px and up) */
@media only screen and (min-width: 992px) {...}

/* Extra large devices (large laptops and desktops, 1200px and up)
*/
@media only screen and (min-width: 1200px) {...}
```

Orientación: Vertical / Horizontal

Las consultas de medios también se pueden utilizar para cambiar el diseño de una página dependiendo de la orientación del navegador.

Puedes tener un conjunto de propiedades CSS que sólo se apliquen cuando la ventana del navegador es más ancha que alta, una orientación llamada "Horizontal" (Landscape):



Ejemplo ocultar elementos con media query

```
/* If the screen size is 600px wide or less, hide the element */
@media only screen and (max-width: 600px) {
  div.example {
    display: none;
  }
}
```

Cambiar tamaños de fuentes

```
/* If the screen size is 601px or more, set the font-size of <div>
to 80px */
@media only screen and (min-width: 601px) {
  div.example {
    font-size: 80px;
  }
}

/* If the screen size is 600px or less, set the font-size of <div>
to 30px */
@media only screen and (max-width: 600px) {
  div.example {
    font-size: 30px;
  }
}
```

Responsive Web Design - Frameworks

Hay varios marcos de CSS (CSS frameworks) gratuitos que ofrecen Diseño Responsivo. Estos marcos proporcionan conjuntos predefinidos de estilos y componentes que facilitan la creación de sitios web receptivos y adaptables a una variedad de dispositivos y tamaños de pantalla. Algunos de los frameworks de CSS más populares y ampliamente utilizados que ofrecen características de diseño responsivo son:

- **Bootstrap:** Desarrollado por Twitter, Bootstrap es uno de los frameworks de CSS más populares que ofrece una amplia gama de componentes y utilidades para la creación rápida de sitios web responsivos.
- **Foundation:** Desarrollado por ZURB, Foundation es otro framework de CSS robusto que proporciona herramientas y componentes para diseñar sitios web receptivos y escalables.
- **Materialize:** Basado en el diseño de Material Design de Google, Materialize es un framework de CSS que ofrece componentes y estilos inspirados en el diseño moderno y limpio de Material Design, con soporte para diseño responsivo.
- **Semantic UI:** Semantic UI es un framework de CSS que se centra en la semántica del HTML y ofrece una colección de componentes y estilos modernos y adaptables para la creación de sitios web receptivos y accesibles.
- **Tailwind CSS:** Tailwind CSS es un framework de CSS único que se centra en la construcción de interfaces de usuario personalizadas utilizando clases de utilidad de bajo nivel. Es altamente configurable y permite un control granular sobre el diseño y el estilo de un sitio web responsivo.

Sitio Responsive con Bootstrap

```
<html lang="en">
<head>
  <title>Bootstrap 5 Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body>

<div class="container-fluid p-5 bg-primary text-white text-center">
  <h1>My First Bootstrap Page</h1>
  <p>Resize this responsive page to see the effect!</p>
</div>

<div class="container mt-5">
  <div class="row">
    <div class="col-sm-4">
      <h3>Column 1</h3>
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit...</p>
    </div>
    <div class="col-sm-4">
      <h3>Column 2</h3>
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit...</p>
    </div>
    <div class="col-sm-4">
      <h3>Column 3</h3>
    </div>
  </div>
</div>
```

My First Bootstrap Page

Resize this responsive page to see the effect!

Column 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit...

Column 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit...

Column 3