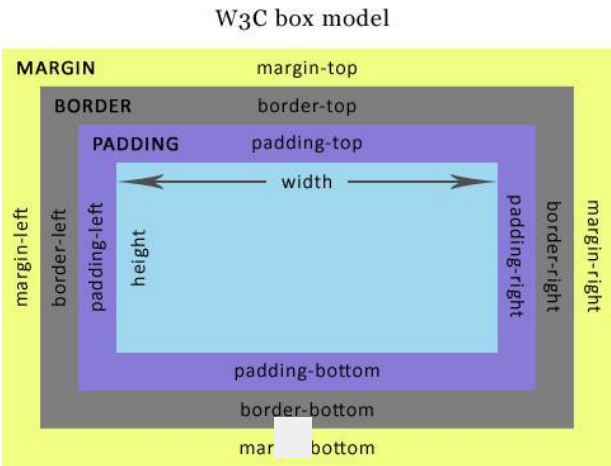


5. Modelo de cajas: márgenes, relleno y bordes.

Los navegadores, de forma automática, crean y colocan cada elemento HTML en una estructura que tiene formato de caja. Así, por defecto, la caja de los elementos HTML de tipo *block* ocupa todo el ancho de la ventana, y la de los elementos *inline* sólo toman la anchura necesaria.

En principio, la caja de cada elemento no es visible, puesto que no muestra ningún color de fondo ni borde. Sin embargo, se puede modificar con CSS para cambiar sus dimensiones, colores, borde y posición.

Se pueden modificar las propiedades de la caja de cualquier elemento HTML, pero generalmente se utiliza el elemento <div> para organizar y dar estructura al diseño de las páginas web.



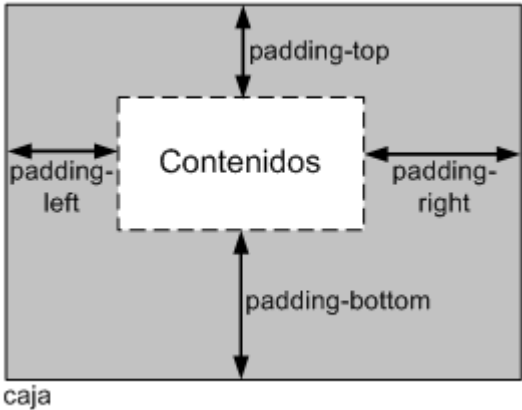
Dimensiones de la caja

Las partes que componen una caja en su orden de visualización desde el punto de vista del usuario son:

- **Contenido** , *content*: se refiere al contenido del elemento; por ejemplo, el texto, una imagen, etc.
- **Relleno**, *padding*: espacio libre opcional existente entre el contenido y el borde.
- **Borde** ,*border* :línea que envuelve el contenido y el relleno.
- **Imagen de fondo**, *background image*: imagen que se muestra por detrás del contenido y el relleno.
- **Color de fondo**, *background color*: color que se muestra detrás del contenido y el relleno.
- **Margen**, *margin*:separación opcional existente entre la caja y el resto de cajas adyacentes.

5.1. Padding o relleno

El padding es el margen interno de un elemento, también se le llama relleno y es la **cantidad de espacio entre el borde y el contenido del elemento**. Veamos una tabla con las propiedades para dar un margen interno a un elemento.



Propiedad	Descripción	Valores
Padding-top padding-right padding-bottom padding-left	Tamaño del relleno superior, derecho, inferior e izquierdo	longitud porcentaje
padding	Tamaño del relleno	longitud porcentaje {1,4}

Valores del padding (también aplicables a la propiedad margin):

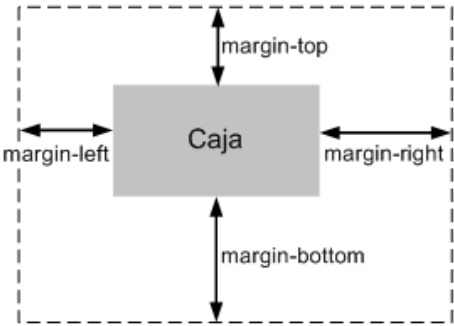
- **Un valor:** se aplica el estilo a los **4 lados**.
- **Dos valores:** el primer valor se aplica a **arriba y abajo**, el segundo valor se aplica a **derecha e izquierda**.
- **Tres valores:** el primer valor se aplica a **arriba**, el segundo valor a **derecha e izquierda** y el tercer valor se aplica a **abajo** del elemento
- **Cuatro valores:** el primer valor se aplica a **arriba**, el segundo valor se aplica a la **derecha**, el tercer valor se aplica a **abajo** y el cuarto valor se aplica a la **izquierda**.

Ejemplo Padding

HTML	CSS
<pre><html> <head> <meta charset="utf-8"> <title>Propiedad padding</title> <link rel="stylesheet" href="styles.css"> </head> <body> <h3>Propiedad padding</h3> <div class="a" border="1"> <p>Contenedor con padding-top: 30px; padding-bottom: 80px; padding: 50px; padding: 40px;</p> </div>
 <div class="b" border="1"> <p>Contenedor con padding: 30px 50px 80px 40px</p> </div> </body> </html></pre>	<pre>div.a { padding-top: 30px; padding-bottom: 80px; padding-right: 50px; padding-left: 40px; background-color: azure; } div.b { /*Propiedad corta*/ padding: 30px 50px 80px 40px; background-color: darkseagreen; }</pre> <div><div>Propiedad padding</div><div>Contenedor con padding-top: 30px; padding-bottom: 80px; padding: 50px; padding: 40px;</div><div>Contenedor con padding: 30px 50px 80px 40px</div></div>

5.2. Margin

El **margin** es el **margin externo** de un elemento, fuera de cualquier borde definido. Veamos las distintas propiedades para dar estilo a los márgenes de un elemento.



Propiedad	Descripción	Valores
Margin-top margin-right margin-bottom margin-left	Tamaño del margen superior, derecho, inferior e izquierdo	longitud porcentaje auto
margin	Ancho de los márgenes	longitud porcentaje auto {1,4}

Ejemplo Margin

HTML	CSS
<pre> <html> <head> <meta charset="utf-8"> <title>Propiedad margin</title> <link rel="stylesheet" href="styles.css"> </head> <body> <h3>Propiedad margin</h3> <div class="a" border="1"> <p>Contenedor con margin-top: 30px; margin-bottom: 80px; margin-right: 50px; margin-left: 40px;</p> </div> <div class="b" border="1"> <p>Contenedor con margin: 30px 80px 50px 40px</p> </div> </body> </html> </pre>	<pre> div.a { margin-top: 30px; margin-bottom: 80px; margin-right: 50px; margin-left: 40px; background-color: azure; } div.b { /*Propiedad corta*/ margin: 30px 80px 50px 40px; background-color: darkseagreen; } </pre> <p>La captura de pantalla muestra una ventana de navegador con la pestaña 'Propiedad margin'. La barra de direcciones indica la ruta local 'C:/Users/CSJ/D...'. El contenido de la página muestra el título 'Propiedad margin'. Hay dos párrafos: el primero, 'Contenedor con margin-top: 30px; margin-bottom: 80px; margin-right: 50px; margin-left: 40px;', está resaltado en azul claro; el segundo, 'Contenedor con margin: 30px 80px 50px 40px', está resaltado en verde oscuro.</p>

5.3. Border

La propiedad *border* en CSS permite **especificar el estilo, el ancho y el color de los bordes de un elemento**. Pueds usar diferentes valores para crear distintos tipos de bordes, como líneas lisas, de puntos, redondeados, etc. Veamos las propiedades para dar estilo a los bordes de un elemento.

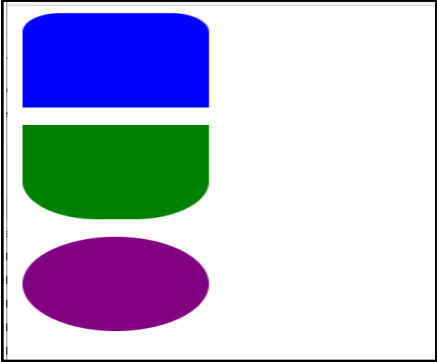
Propiedad	Descripción	Valores
Border-top-width border-right-width border-bottom-width border-left-width	Anchura del borde superior, derecho, inferior o izquierdo	thin medium thick longitud
border-width	Anchura del borde (reducida)	thin medium thick longitud {1,4}
border-top-color border-right-color border-bottom-color border-left-color	Color del borde superior, derecho, inferior e izquierdo	color transparent
border-color	Color del borde (reducida)	color transparent {1,4}
border-top-style border-right-style border-bottom-style border-left-style	Estilo del borde superior, derecho, inferior e izquierdo	none hidden dotted dashed solid double groove ridge inset outset
border-style	Estilo del borde (reducida)	none hidden dotted dashed solid double groove ridge inset outset {1,4}
Border-top border-right border-bottom border-left	Ancho, estilo y color para el borde superior, derecho, inferior e izquierdo	border-top-width border-top-style border-top-color
border	Ancho, estilo y color para los bordes (reducida)	border-width border-style border-color
border-radius	Curvatura del borde	longitud porcentaje {1,4}

***Las propiedades del borde no tendrán efecto hasta que se defina la propiedad **border-style**.*

Ejemplo border

HTML	CSS
<pre><html> <head> <meta charset="utf-8"> <title>Propiedad margin</title> <link rel="stylesheet" href="styles.css"> </head> <body> <h3>Propiedad border</h3> <p class="a">Párrafo con un borde sólido de color azul</p> <p class="b">Párrafo con borde verde punteado</p> <p class="c">Párrafo con borde violeta a rayas</p> <p class="d">Párrafo con borde mixto</p> </body> </html></pre>	<pre>.a { border: 4px solid blue; } .b { border: 6px dotted green; } .c { border: 3px dashed purple; } .d { border-top: 3px double orange; border-right: 3px double brown; border-bottom: 3px double brown; border-left: 3px double brown; }</pre> <p>Propiedad border</p> <div><div>Párrafo con un borde sólido de color azul</div><div>Párrafo con borde verde punteado</div><div>Párrafo con borde violeta a rayas</div><div>Párrafo con borde mixto</div></div>

Ejemplo border-radius

HTML	CSS
<pre><html> <head> <meta charset="utf-8"> <title>Propiedad margin</title> <link rel="stylesheet" href="styles.css"> </head> <body> <div class="a"></div>
 <div class="b"></div>
 <div class="c"></div>
 </body> </html></pre>	<pre>.a { background-color: blue; border-radius: 20px 20px 0 0; height:100px; width: 100px; } .b { background-color: green; border- radius: 0 0 40px 40px; height:100px; width: 100px; } .c { background-color: purple; border- radius: 50px; height:100px; width: 100px; }</pre> 

5.4. Propiedades width, height, max-width y min-height

Veamos una tabla resumen de las propiedades que vamos a tratar en esta sección:

Nombre propiedad	Descripción	Valores
width	Establece el ancho del área de contenido de un elemento	Unidad de longitud (px, em, %, etc.), auto, initial, inherit
height	Establece el alto del área de contenido de un elemento	Unidad de longitud (px, em, %, etc.), auto, initial, inherit
max-width	Establece el ancho máximo que puede tener un elemento	Unidad de longitud (px, em, %, etc.), none, initial, inherit
min-width	Establece el ancho mínimo que debe tener un elemento	Unidad de longitud (px, em, %, etc.), 0, initial, inherit

Las **propiedades CSS width y height** se usan para **establecer el ancho y el alto de un elemento**. Por defecto, estas propiedades se refieren al área de contenido del elemento, es decir, el espacio que ocupa el texto, las imágenes u otros elementos dentro del elemento. El área de contenido **no incluye el padding (relleno), el border (borde) ni el margin (margen) del elemento**.

Sin embargo, podemos cambiar este comportamiento con la propiedad CSS box-sizing, que nos permite definir qué partes del elemento se incluyen en el cálculo del ancho y el alto. Esto lo veremos más adelante.

Ejemplos de uso de width y height con diferentes unidades:

- Para darle un ancho y un alto fijo a un elemento, podemos usar píxeles como unidad. Por ejemplo, si queremos que un elemento tenga un ancho de 200px y un alto de 100px, podemos usar este código:

```
div {  
  width: 200px;  
  height: 100px;  
}
```

- Para darle un ancho y un alto relativo al elemento contenedor, podemos usar porcentajes como unidad. Por ejemplo, si queremos que un elemento ocupe el 50% del ancho y el 25% del alto del elemento padre, podemos usar este código:

```
div {  
  width: 50%;  
  height: 25%;  
}
```

Las propiedades CSS **max-width** y **max-height** se usan para establecer el ancho y el alto máximo de un elemento. Estas propiedades nos permiten limitar el tamaño de un elemento para que no supere cierto valor, incluso si le damos un ancho o un alto mayor con las propiedades **width** o **height**.

Esto puede ser muy útil para crear diseños responsivos que se adapten a diferentes tamaños de pantalla. Por ejemplo, podemos usar **max-width** para evitar que una imagen se salga del contenedor o que se distorsione al cambiar la resolución. También podemos usar **max-height** para controlar la altura de un elemento que tenga contenido variable, como un menú desplegable o una caja de texto.

Ejemplos de uso de max-width y max-height con diferentes unidades:

- Para darle un ancho y un alto máximo a un elemento en píxeles, podemos usar esta unidad como valor. Por ejemplo, si queremos que una imagen no supere los 300px de ancho ni los 200px de alto, podemos usar este código:

```
img {  
  max-width: 300px;  
  max-height: 200px;  
}
```

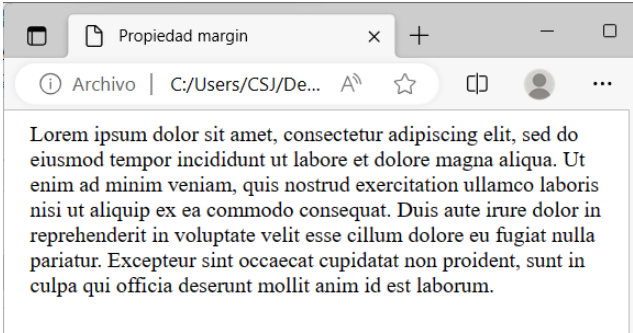
- Para darle un ancho y un alto máximo a un elemento en porcentajes, podemos usar esta unidad como valor. Por ejemplo, si queremos que una imagen no ocupe más del 50% del ancho ni del 25% del alto del elemento contenedor, podemos usar este código:

```
img {  
  max-width: 50%;  
  max-height: 25%;  
}
```

Veamos un ejemplo práctico: Para crear un contenido a pantalla completa que no se haga más grande al llegar a un ancho máximo, puedes usar la propiedad **max-width** junto con la propiedad **width**. Así, le das al elemento un ancho relativo al tamaño de la pantalla, pero también le pones un límite absoluto.

Por ejemplo, si quieres que el contenido ocupe el 95% del ancho de la pantalla, pero que no supere los 1200px, puedes usar este código:

```
.content {
width: 95%; /* El contenido ocupa el 95% del ancho de la pantalla */
max-width: 1200px; /* El contenido no ocupará más de 1200px */
margin: 0 auto; /* Centrar el contenido, primer valor para arriba y abajo,
segundo valor para izd y dcha */
}
```

HTML	CSS
<pre><html> <head> <meta charset="utf-8"> <title>Propiedad margin</title> <link rel="stylesheet" href="styles.css"> </head> <body> <div class="content">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</div> </body> </html></pre>	<pre>.content { width: 95%; /* El contenido ocupa el 95% del ancho de la pantalla */ max-width: 1200px; /* El contenido no ocupará más de 1200px */ margin: 0 auto; /* Centrar el contenido, primer valor para arriba y abajo, segundo valor para izd y dcha */ }</pre> 

5.5. Posición y comportamiento de contenedores CSS

En el proceso de creación de una web es imprescindible **organizar elementos como imágenes, textos o tablas**. Para ello, necesitaremos conocer los elementos de ordenación y las propiedades que nos ayudan a organizar todos los componentes. Las propiedades más importantes se definen en la siguiente tabla:

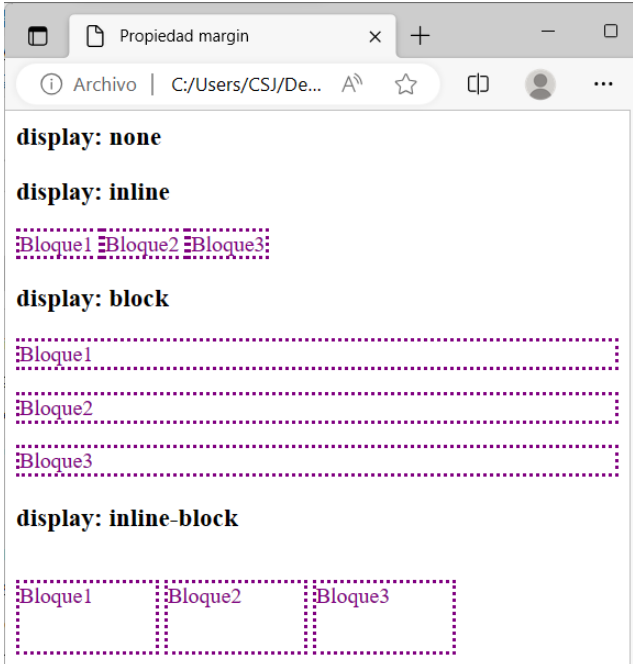
Propiedad	Descripción	Valores
display	Comportamiento del contenedor	inline block inline-block none
position	Esquema de posicionamiento	static relative absolute fixed sticky
Top right botto left	Desplazamiento de la caja respecto al borde superior, derecho, inferior o izquierdo	longitud porcentaje auto
float	Posicionamiento flotante	left right none
clear	Control de cajas adyacentes a las float	none left right both
z-index	Nivel de la capa	auto número entero
box-sizing	Control de bordes y relleno en el comportamiento del contenedor	content-box border-box
visibility	Muestra u oculta un elemento ocupando el espacio	visible hidden

5.6. Display

Valores: **none** | **inline** | **block** | **inline-block**

- **none:** los elementos se ocultan y no se muestra el espacio reservado.
- **Inline:** los elementos se muestran en la misma línea (respetando el flujo) y no se aceptan las propiedades width, height ni márgenes verticales.
- **Block:** los elementos se muestran en líneas independientes y se aceptan las propiedades width, height y márgenes verticales.
- **Inline-block:** su comportamiento es una mezcla entre los dos anteriores, los elementos se muestran en la misma línea (respetando el flujo) y se aceptan las propiedades width, height y márgenes verticales.

Ejemplo display

HTML	CSS
<pre> <html> <head> <meta charset="utf-8"> <title>Propiedad margin</title> <link rel="stylesheet" href="styles.css"> </head> <body> <h3>display: none</h3> <p class="a">Bloque1 </p> <p class="a">Bloque2 </p> <p class="a">Bloque3 </p> <h3>display: inline</h3> <p class="b">Bloque1 </p> <p class="b">Bloque2 </p> <p class="b">Bloque3</p> <h3>display: block</h3> <p class="c">Bloque1 </p> <p class="c">Bloque2 </p> <p class="c">Bloque3 </p> <h3>display: inline-block</h3> <p class="d">Bloque1 </p> <p class="d">Bloque2 </p> <p class="d">Bloque3 </p> </body> </html> </pre>	<pre> .a { display: none; } .b { display: inline; width: 100px; height: 50px;} .c { display: block; } .d { display: inline-block; width: 100px; height: 50px;} p { color: purple; border: dotted;} </pre> 

5.7. Position

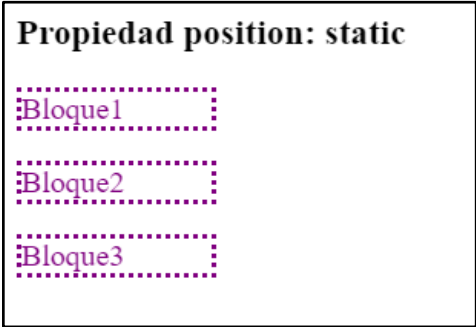
Valores: **static** | **relative** | **absolute** | **fixed** | **sticky**

- **static:** los elementos se posicionan de acuerdo al flujo normal de la página. Es la **posición natural de los elementos**. No son afectados por las propiedades top, bottom, left y right.
- **Relative:** los elementos se posicionan de forma relativa a su posición normal.
- **Fixed:** los elementos se posicionan de forma relativa a la ventana del navegador. Su posición permanece fija aunque se desplace la ventana.

- **Absolute:** los elementos se posicionan de forma relativa al primer elemento padre que tenga una posición distinta a static.
- **sticky:** los elementos son posicionados de forma relativa hasta que su bloque contenedor alcanza un límite establecido.

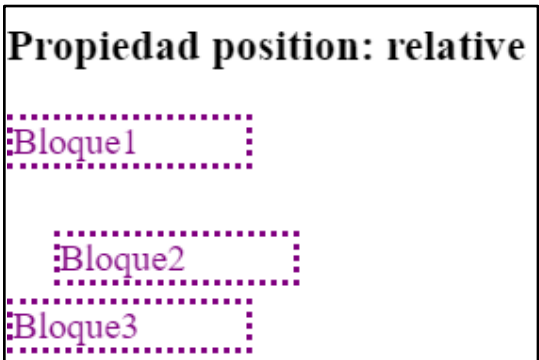
Ejemplo position: static

Las propiedades top, bottom, left y right no están permitidas con «position: static».

HTML	CSS
<pre><html> <head> <meta charset="utf-8"> <title>Propiedad position static</title> <link rel="stylesheet" href="style.css"> </head> <body> <h3>Propiedad position: static</h3> <p class="a">Bloque1 </p> <p class="b">Bloque2 </p> <p class="c">Bloque3 </p> </body> </html></pre>	<pre>.a { position: static; top: 800px; left: 400px;} /*Como puedes ver, en position static no funcionan las propiedades top, bottom, left y right*/ .b { position: static; } .c { position: static; } p { width: 100px; color: purple; border: dotted;}</pre> 

Ejemplo position: relative

Aplica el siguiente estilo sobre un elemento y comprueba que con las propiedades left: 20px y top: 10px el elemento se desplaza 20px hacia la derecha y 10 px hacia abajo desde su posición por defecto (sin eliminar el hueco de su posición por defecto).

HTML	CSS
<pre><html> <head> <meta charset="utf-8"> <title>Propiedad relative</title> <link rel="stylesheet" href="styles.css"> </head> <body> <h3>Propiedad position: relative</h3> <p>Bloque1 </p> <p class="bloque2">Bloque2 </p> <p>Bloque3 </p> </body> </html></pre>	<pre>.bloque2 { position: relative; left: 20px; top: 10px; } p { width: 100px; color: purple; border: dotted;}</pre> 

Ejemplo position: fixed

Los elementos con position:fixed toman como referencia la ventana del navegador y permanecen fijos.

HTML	CSS
<pre><html> <head> <meta charset="utf-8"> <title>Propiedad relative</title> <link rel="stylesheet" href="styles.css"> </head> <body> <h3>Propiedad position: fixed</h3> <p>Bloque1 </p> <p class="bloque2">Bloque2 </p> <p>Bloque3 </p> </body> </html></pre>	<pre>.bloque2 { position: fixed; top: 130px; left: 100px; } p { width: 100px; color: purple; border: dotted;}</pre>

Ejemplo position: absolute

Establece el siguiente estilo sobre dos contenedores y comprueba cómo el elemento «.b» deja de seguir la posición del flujo normal de la página, sin crearse espacio alguno para el elemento, y se posiciona de forma relativa al primer elemento padre que tiene una posición distinta a static, en este caso el elemento «.a». Si no hubiese ningún elemento padre con propiedad distinta a static, se ubicaría de forma relativa al contenedor inicial. Haz la prueba cambiando el valor de position a «static» en el elemento a. Como puedes ver, su posición final está definida por los valores de top, right, bottom, y left.

HTML	CSS
<pre><html> <head> <meta charset="utf-8"> <title>Propiedad relative</title> <link rel="stylesheet" href="styles.css"> </head> <body> <div class="a"> <p>Contenedor con position: relative</p> <p>Contenedor con position: relative</p> <p>Contenedor con position: relative</p> <p>Contenedor con position: relative</p> <p>Contenedor con position: relative</p> <div class="b"><p>Contenedor con position: absolute</p></div> </div> </body> </html></pre>	<pre>.a { position: relative; /*cambia este valor a static para ver la diferencia*/ width: 300px; height: 300px; border: 2px dotted purple; } .b { position: absolute; background-color: #elf4fc; top: 60px; right: 0px; width: 200px; height: 80px; border: 3px solid blue; }</pre>

Ejemplo position: sticky

La posición **sticky** se usa cuando queremos que un elemento tenga una posición relativa hasta un punto y que luego cambie a una posición fija, usando solo CSS sin necesidad de código JavaScript. Por ejemplo si tenemos un banner de publicidad flotante en el sidebar y nos interesa que una vez aparezca al hacer scroll se mantenga fijo y visible.

Otro ejemplo puede ser el que se muestra a continuación y que nos sirve para posicionar los encabezados en una lista alfabética.

HTML	CSS
<pre><html> <head> <meta charset="utf-8"> <title>Propiedad relative</title> <link rel="stylesheet" href="styles.css"> </head> <body> <p> A </p> aguacate ahuyama <p> B </p> brocoli berros banano </body> </html></pre>	<pre>p { position: -webkit-sticky; position: sticky; top: 0px; background-color: #elf4fc; padding: 20px; font-weight: bold; } ul{ margin: 20px 0; } li{ padding-left: 20px; }</pre> <div><div>Propiedad margin</div><div>Archivo C:/Users/CSJ/De...<div><div></div><div></div><div></div><div></div><div></div></div></div><div><div>A</div><div><ul style="list-style-type: none">aguacateahuyama</div></div><div><div>B</div><div><ul style="list-style-type: none">brocoliberrosbanano</div></div></div>

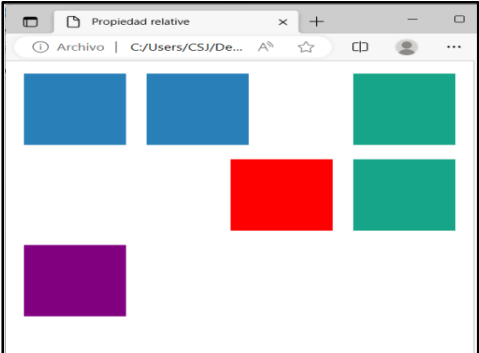
5.8. Float

Valores: **left** | **right** | **none**

Cuando a un elemento HTML se le aplica un estilo con la propiedad float, el elemento sale del flujo normal y aparece posicionado a la izquierda o a la derecha de su contenedor, donde el resto de elementos de la página se posicionarán alrededor.

Ejemplo float

Aplica los siguientes estilos sobre diferentes contenedores y comprueba el resultado.

HTML	CSS
<pre><html> <head> <meta charset="utf-8"> <title>Propiedad relative</title> <link rel="stylesheet" href="styles.css"> </head> <body> <div class="a"> <div></div> </div> <div class="a"> <div></div> </div> <div class="b"> <div></div> </div> <div class="b"> <div></div> </div> <div class="c"> <div></div> </div> <div class="d"> <div></div> </div> </body> </html></pre> 	<pre>.a { float: left; padding: 10px; } .a div{ height: 100px; width: 100px; background-color: #2980B9; } .b { float: right; padding: 10px; } .b div{ height: 100px; width: 100px; background-color: #17A589; } .c { float: right; padding: 10px; } .c div{ height: 100px; width: 100px; background-color: red; } .d { float: none; clear: both; padding: 10px; } .d div{ height: 100px; width: 100px; background-color: purple; }</pre>

5.9. Clear

Valores: **none** | **left** | **right** | **both**

La propiedad clear establece si un elemento debe estar al lado de los elementos flotantes que lo preceden o si debe situarse bajo de ellos. Se suele utilizar para restaurar el flujo normal del documento y así los elementos dejan de flotar hacia la izquierda, la derecha o ambos lados.

Ejemplo clear

Crea un contenedor y aplícale la propiedad float con el valor left. Sitúa un texto bajo del contenedor creado utilizando la propiedad clear.

HTML	CSS
<pre><html> <head> <meta charset="utf-8"> <title>Propiedad relative</title> <link rel="stylesheet" href="styles.css"> </head> <body> <div class="a"> <div></div> </div> <div class="b"> <div></div> </div> <h4>Texto sin propiedad clear</h4> <p>Texto con propiedad clear:both</p> </body> </html></pre>	<pre>.a { float: left; padding: 10px; } .a div{ height: 100px; width: 100px; background-color: #2980B9; /*Azul*/ } .b { float: right; padding: 10px; } .b div{ height: 100px; width: 100px; background-color: #17A589; } p{ clear: both; }</pre> <div><div>Texto sin propiedad clear</div><div>Texto con propiedad clear:both</div></div>

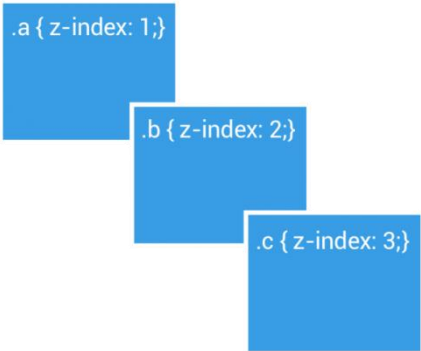
5.10. 5. Z-index

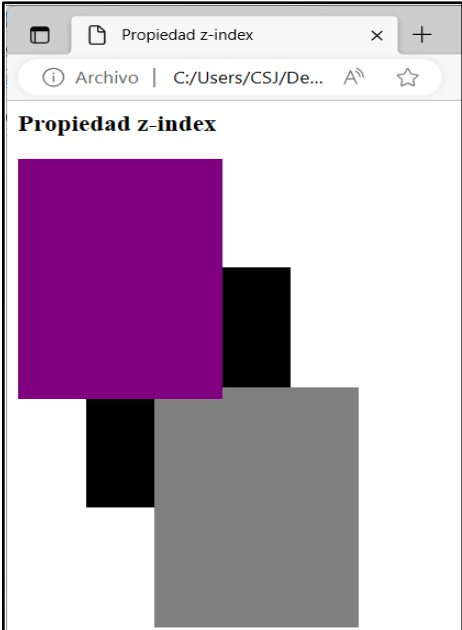
Valores: **auto** | **número entero**

Mediante el atributo z-index podemos organizar cada uno de los elementos del contenido de una página web.

Ejemplo z-index

Aplica los siguientes estilos sobre diferentes contenedores que se encuentren superpuestos y comprueba el resultado.



HTML	CSS
<pre> <html> <head> <meta charset="utf-8"> <title>Propiedad z-index</title> <link rel="stylesheet" href="style.css"> </head> <body> <h3>Propiedad z-index</h3> <div class="a"></div> <div class="b"></div> <div class="c"></div> </body> </html> </pre>	<pre> .a { width: 150px; height: 200px; background-color: purple; position: relative; z-index: 3;} .b { width: 150px; height: 200px; background-color: black; position: relative; left: 50px; top: -110px; z-index: 1;} .c { width: 150px; height: 200px; background-color: grey; position: relative; left: 100px; top: -210px; z-index: 2;} </pre> 

5.11. Box-sizing

Por defecto en el modelo de cajas de CSS, el ancho y alto asignado a un elemento es aplicado solo al contenido de la caja del elemento. Si el elemento tiene algún borde (border) o relleno (padding), este es entonces añadido al ancho y alto del tamaño de la caja o contenedor. Esto significa que **cuando se define el ancho y alto, se tiene que ajustar el valor para permitir cualquier borde o relleno que se pueda añadir.**

Valores: **content-box** | **border-box**

- **content-box** es el comportamiento CSS por defecto para el tamaño de la caja (box-sizing). Si se define el ancho de un elemento en 100 pixeles, la caja del contenido del elemento tendrá 100 pixeles de ancho, y el ancho de cualquier borde o relleno será añadido al ancho final desplegado.
- **Border-box** toma en cuenta cualquier valor que se especifique de borde o de relleno para el ancho o alto de un elemento. Es decir, si se define un elemento con un ancho de 100 pixeles. Esos 100 pixeles incluirán cualquier borde o relleno que se añada, y la caja de contenido se encogerá para absorber ese ancho extra. Esta propiedad es especialmente útil para redimensionar cualquier elemento.

Ejemplo box-sizing

Crea un contenedor que ocupe el 100% del ancho de la pantalla. Posiciona tres imágenes en línea y define que cada imagen ocupe el 33,333%. Observa que el conjunto ocupa el 100% de la pantalla.

Si a continuación dotamos a las imágenes de un padding o relleno, el conjunto ocupará más del 100%. En este punto podríamos establecer un “box-sizing: border-box” para incluir en el conjunto el relleno definido. Acuérdate de añadir los prefijos para los navegadores:

HTML	CSS
<pre><html> <head> <meta charset="utf-8"> <title>Propiedad z-index</title> <link rel="stylesheet" href="styles.css"> </head> <body> <div class="img-container"> </div> <div class="img-container"> </div> <div class="img-container"> </div> </body> </html></pre>	<pre>.img-container { box-sizing: border-box; -webkit-box-sizing: border-box; float: left; width: 33.33%; padding: 5px; } .img-container img{ width: 100%; }</pre> <div></div>

5.12. Visibility

La propiedad visibility indica si un elemento es visible o permanece oculto (ocupando el mismo espacio).

Valores: **visible** | **hidden**

Ejemplo visibility

Aplica las siguientes propiedades sobre un elemento y observa las diferencias. Como puedes ver, la diferencia principal es que display: none **no reserva el espacio del elemento mientras que** visibility: hidden **sí**.

HTML	CSS
<pre> <html> <head> <meta charset="utf-8"> <title>Propiedad visibility</title> <link rel="stylesheet" href="styles.css"> </head> <body> <div class="cajaFlotanteIzq"> Caja2 Caja7 <div class="cajaFlotanteIzq"> Caja1 Caja3 Caja4 Caja5 Caja6 Caja8 </div> </div> <div class="cajaFlotanteIzq"> Caja6 </div> <div class="cajaFlotanteIzq" style="visibility:hidden"> Caja7 </div> <div class="cajaFlotanteIzq"> Caja8 </div> </body> </html> </pre>	<pre> div { position: relative; width: 250px; height: 180px; border: 1px solid black; margin: 2px; box-sizing: border-box; padding: 2px; background-color: rgba(255, 0, 0, 0.2); } .cajaInline { height: 20px; display: inline; } .cajaBlock { width: 142px; height: 20px; margin-top: 8px; } .cajaRel { position: relative; top: 10px; left: 5px; width: 50px; height: 20px; display: inline; } .cajaAbs { position: absolute; top: 80px; left: 100px; width: 50px; height: 20px; display: inline; } .cajaFija { position: fixed; top: 250px; left: 100px; width: 50px; height: 20px; display: inline; } .cajaFlotanteDcha{ float: right; width: 50px; height: 20px; } .caixaFlotanteIzq{ float: left; width: 50px; height: 20px; } body { font-family: arial; font-size: 0.8em; } p { margin: 2px; } </pre>

NOTA PROPIEDAD POSITION

Conforme vayamos aprendiendo querremos hacer diseños más complejos y mover los elementos con respecto a dónde les correspondería según el flujo normal atendiendo a su propiedad display.

Para este posicionamiento más exacto y concreto CSS nos proporciona la propiedad *position* cuyo valores van íntimamente asociados a las propiedades CSS *top*, *bottom*, *left*, *right* y *z-index*. Las 4 primera de estas propiedades indicar desplazamientos conforme a un punto de referencia en concreto y la propiedad *z-index* nos va a permitir trabajar con capas:

Los distintos valores que puede tomar esta propiedad son:

- **static**: es el valor por defecto. El elemento sigue el flujo que lo que corresponde. Aunque use *top*, *bottom*, *left*, *right* o *z-index* al elemento no se aplica ningún desplazamiento.
- **relative**: Es como *static* pero si atiende a los desplazamientos expresados en *top*, *bottom*, *left*, *right* o *z-index*.
- **fixed**: Se le aplica *top*, *bottom*, *right* o *z-index* en relación con documento. No atiende al scroll una vez generada por lo que su posición siempre permanece fija.
- **absolute**: Se comporta como *fixed* pero en relación con la primera etiqueta padre que tenga *position: relative*.
- **sticky**: Se comporta como *relative* hasta llegar a una posición de scroll y a partir de entonces *fixed*.

La propiedad z-index

Al posicionar los elementos con las propiedades *position* puede suceder que nos encontremos que haya elementos que lleguen a solaparse por tener que ocupar la misma área.

Con *z-index* podemos establecer capas decidiendo el orden de solapamiento de estos elementos. Se mostrará arriba del todo aquel elemento de los que se solapen con el mayor valor de *z-index*.

Centrado vertical de elementos de bloque

Una vez ya sabemos usar la propiedad *position* podemos centrar verticalmente elementos de bloque. Nos encontraremos con dos casos:

- Cuando conocemos la altura del elemento.
- Cuando desconocemos la altura del elemento.

Si conocemos la altura del elemento y esta es por ejemplo 150px;

```
css {  
  .contenedor {  
    position: relative;  
  }  
  .elemento_a_centrar {  
    height: 150px;  
    margin-top: -75px; /** La mitad de la altura **/  
    position: absolute;  
    top: 50%;  
  }  
}
```

Si desconocemos la altura del elemento:

```
.contenedor {  
  position: relative;  
}  
.elemento_a_centrar {  
  position: absolute;  
  top: 50%;  
  transform: translateY(-50%);  
}
```