

A graphic on the left side of the slide. It features a 3D effect with four stacked rectangular blocks in purple, orange, yellow, and blue. The text 'Agencia de Aprendizaje a lo largo de la vida' is written across these blocks in white. An orange arrow points to the right from the orange block.

Agencia de
Aprendizaje
a lo largo
de la vida

FULL STACK FRONTEND

Clase 7

CSS 3

Modelo de caja y posicionamiento



Les damos la bienvenida

Vamos a comenzar a grabar la clase

Clase 06

CSS 2 - Medidas, colores, fondos, fuentes e íconos

- Unidades de medida.
- Colores CSS.
- Fondos en CSS.
- Fuentes y tipografías.
- Estilos para textos y listas.
- Íconos.

Clase 07

CSS 3 - Modelo de caja y posicionamiento

- Modelo de caja y propiedades.
- Posicionamiento y visualización.

Clase 08

CSS 4 - Selectores avanzados y Animaciones

- Selectores avanzados.
- Animaciones con CSS.
- Incorporación de transformaciones y transiciones a elementos.
- Introducción Responsive Web Design.

Modelo de caja

El **modelo de caja** o “*box model*” es seguramente la característica más importante del lenguaje de hojas de estilos CSS, ya que condiciona el diseño de todas las páginas web. El Modelo de caja es un mecanismo mediante el cual CSS hace que todos los elementos de las páginas se representen mediante cajas rectangulares.

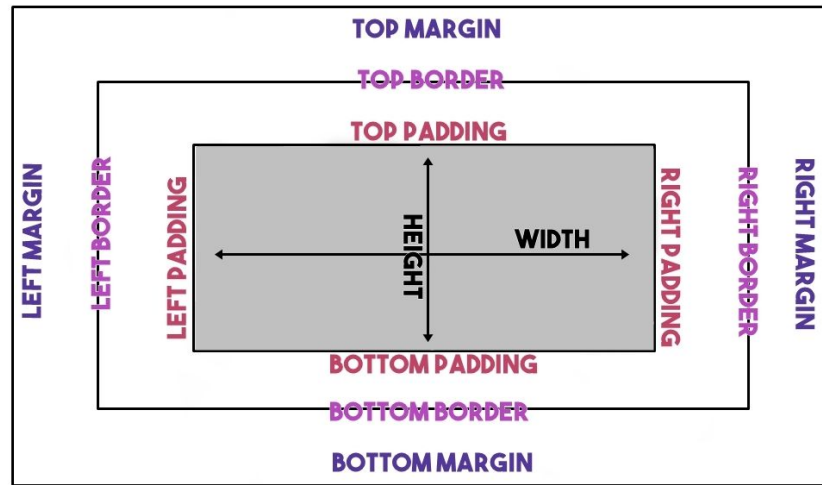
Las cajas de una página se crean automáticamente. Cada vez que se inserta una etiqueta HTML, se crea una nueva caja rectangular que encierra los contenidos de ese elemento.

Los navegadores crean y colocan las cajas de forma automática, pero CSS permite modificar todas sus características.

Modelo de caja

Las cajas de las páginas no son visibles a simple vista porque inicialmente no muestran ningún color de fondo ni ningún borde.

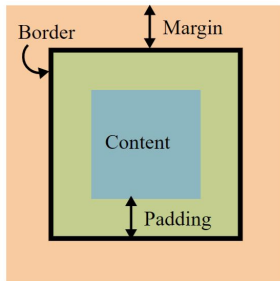
Cada elemento incluido en el documento HTML genera una caja que tiene varios atributos modificables. El comportamiento de esa caja depende de su clasificación, es decir, si se trata de un elemento de línea o de bloque.



Modelo de caja

La representación básica del **Modelo de caja** es la siguiente, donde podemos observar varios conceptos importantes a diferenciar:

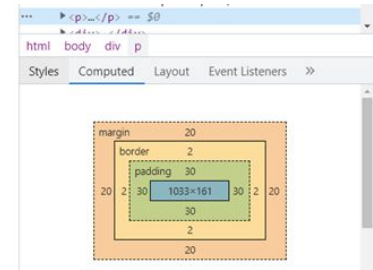
- El **borde** (*border*). En negro, es el límite que separa el interior del exterior del elemento.
- El **margen** (*margin*). En naranja, es la parte exterior del elemento, por fuera del borde.
- El **relleno** (*padding*). En verde, es la parte interior del elemento, entre el contenido y el borde.
- El **contenido** (*content*). En azul, es la parte interior del elemento, excluyendo el relleno.



Contenedor principal

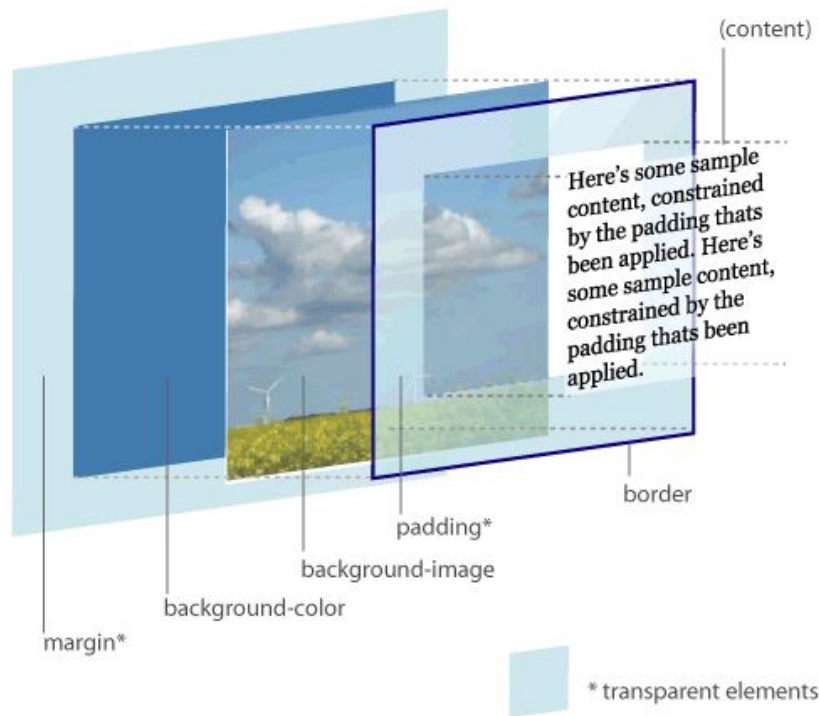
Lorem ipsum dolor, sit amet consectetur adipisicing elit. Necessitatibus maxime, suscipit corporis quasi delectus, officiis voluptate laudantium sint fugiat eius voluptatem iste? Illo a dolor sit fugit laudantium perferendis natus tempora error. Eaue dolor natus facere, nulla adipisci facilis, ratione, placeat molestias nemo debitis aliquam officiis reiciendis corrupti. Non dolorem obcaecati et, suscipit ducimus explicabo porro autem consectetur dicta alias quidem ut magnam ipsa cupiditate nisi officiis rem pariatur doloribus a totam ratione veniam? Iste libero unde saepe rem, laudantium quaerat, eligendi itaque aperiam natus deleniti enim suscipit, fuga autem voluptas sapiente fugit doloremque ducimus temporibus reiciendis non soluta iure.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Perspiciatis voluptatibus laborum sequi soluta neque laboriosam amet provident, est sapiente velit.



Modelo de caja



El relleno y el margen son transparentes, por lo que en el espacio ocupado por el relleno se muestra el color o imagen de fondo (si están definidos) y en el espacio ocupado por el margen se muestra el color o imagen de fondo de su elemento padre (si están definidos). Si ningún elemento padre tiene definido un color o imagen de fondo, se muestra el color o imagen de fondo de la propia página (si están definidos).



Dimensiones y desbordamiento

Dimensiones (ancho y alto)

Proporcionamos tamaños específicos a los diferentes elementos de un documento HTML asignando valores a las propiedades **width** (ancho) y **height** (alto).

Propiedad	Valor	Significado
<code>width</code>	<code>auto</code> 	Tamaño de ancho de un elemento.
<code>height</code>	<code>auto</code> 	Tamaño de alto de un elemento.

En el caso de utilizar el valor **auto** en las propiedades anteriores (valor por defecto), el navegador se encarga de calcular el ancho o alto necesario, dependiendo del contenido del elemento. El tamaño automático dado a un elemento depende del tipo de elemento que se trate (en bloque o en línea).

Dimensiones en modelo de caja

Si en lugar de utilizar la opción **auto**, o simplemente no indicamos valores para **ancho** y **alto**, el tamaño de la caja suele acomodarse al contenido sin problemas. Pero cuando asignamos valores a estos atributos, forzamos al elemento a tener un aspecto concreto, obteniendo resultados inesperados si su contenido es más grande que el tamaño que hemos definido.



min-width, max-width, min-height y max-height

Una forma de mitigar el problema mencionado consiste en utilizar las propiedades hermanas de **width**: **min-width** y **max-width**; y las propiedades hermanas de **height**: **min-height** y **max-height**. Con estas propiedades, en lugar de establecer un tamaño fijo, establecemos límites máximos y mínimos, donde el ancho o alto de la caja queda comprendido entre esos valores.

```
div{  
  width: 800px;  
  height: 800px;  
  background: red;  
  max-width: 500px;  
}
```

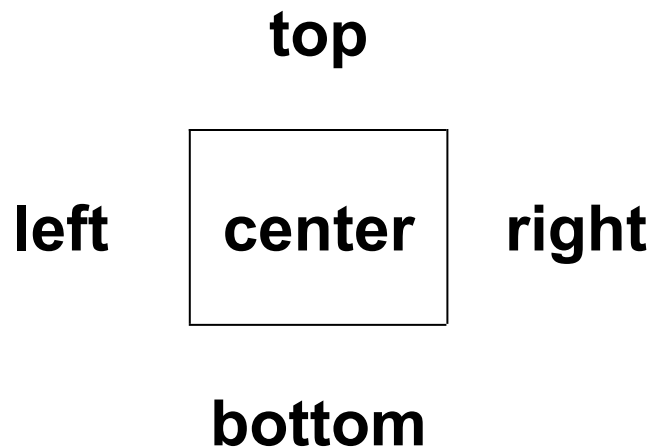
En este caso a pesar de estar indicando un tamaño de **800px**, le aplicamos un **max-width** de **500px**, por lo que estamos limitando el elemento a un tamaño de ancho de 500 píxeles como máximo y nunca superará ese tamaño.

Las propiedades de mínimos (**min-width** y **min-height**) por defecto tienen valor **0**, mientras que las propiedades de máximos (**max-width** y **max-height**) tienen por defecto valor **none**.

Zonas de un elemento

En CSS existen ciertas palabras clave para hacer referencia a una zona u orientación concreta sobre un elemento. Son conceptos muy sencillos y lógicos:

- **Top:** Parte superior
- **Left:** Parte izquierda
- **Right:** Parte derecha
- **Bottom:** Parte inferior
- **Center:** Se refiere a la posición central entre los extremos horizontales y verticales



Desbordamiento

Puede ocurrir que apliquemos un tamaño de alto y ancho a un elemento HTML, pero su contenido de texto sea tan grande que no quepa dentro de la caja.

En ese caso lo que ocurre es que el contenido se desborda. Podemos modificar este comportamiento con la propiedad de CSS **overflow** o con alguna de sus propiedades específicas **overflow-x** u **overflow-y** [+info](#)

Propiedad	Valor	Significado
overflow	visible hidden scroll auto	Establece el comportamiento de desbordamiento.
overflow-x	visible hidden scroll auto	Establece el desbordamiento sólo para el eje X (<i>horizontal</i>).
overflow-y	visible hidden scroll auto	Establece el desbordamiento sólo para el eje Y (<i>vertical</i>).

Desbordamiento

Dichas propiedades pueden tomar varios valores, donde **visible** es el valor que tiene por defecto, permitiendo el desbordamiento. Otras opciones son las siguientes:

Valor	¿Qué ocurre si se desborda el contenedor?	¿Desbordamiento?
visible	Se muestra el contenido que sobresale (<i>comportamiento por defecto</i>)	Sí
hidden	Se oculta el contenido que sobresale.	No
scroll	Se colocan barras de desplazamiento (horizontales y verticales).	No
auto	Se colocan barras de desplazamiento (sólo las necesarias).	No

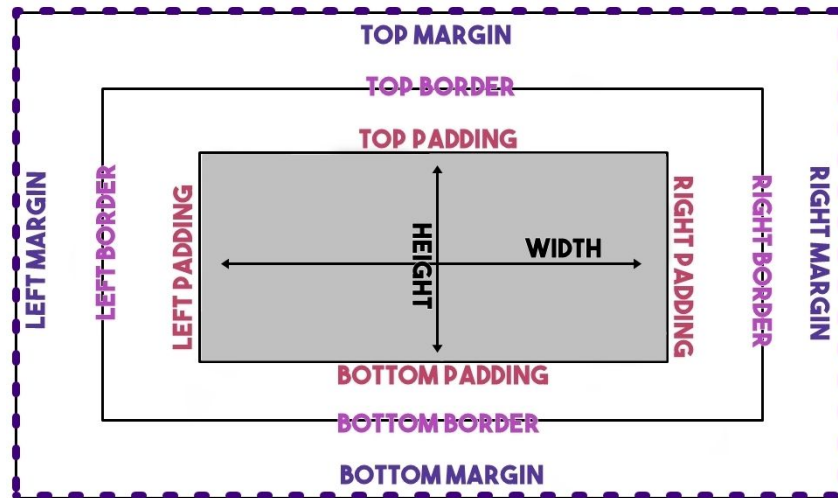
overflow-x y **overflow-y** permiten ocultar alguna barra de desplazamiento (habitualmente la barra de desplazamiento horizontal).

Márgenes y relleno

Márgenes (margin)

Se utilizan para crear espacio alrededor de los elementos, fuera de los bordes definidos. **Margin** especifica el espacio que existe entre el borde de un elemento y el borde de otros elementos adyacentes. Las opciones son:

- margin-top
- margin-right
- margin-bottom
- margin-left



Márgenes (margin)

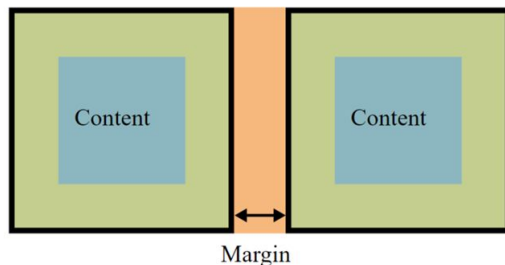
Se puede aplicar en conjunto o de forma concreta a cada una de las zonas del elemento. Estas son las propiedades específicas de cada zona:

Propiedad	Valor	Significado
<code>margin-top</code>	<code>auto</code> <small>SIZE</small>	Establece un tamaño de margen superior.
<code>margin-left</code>	<code>auto</code> <small>SIZE</small>	Establece un tamaño de margen a la izquierda.
<code>margin-right</code>	<code>auto</code> <small>SIZE</small>	Establece un tamaño de margen a la derecha.
<code>margin-bottom</code>	<code>auto</code> <small>SIZE</small>	Establece un tamaño de margen inferior.

Podemos aplicar diferentes márgenes a cada zona de un elemento utilizando cada una de estas propiedades, o dejando al navegador que lo haga de forma automática indicando el valor **auto**.

Márgenes (margin)

Para centrar un elemento podemos aplicar un ancho fijo al contenedor. Por ejemplo: **width: 500px** con **margin: auto**. El navegador, al conocer su tamaño, se encarga de repartirlo equitativamente entre los márgenes, dado que conoce el resto del tamaño de la ventana.



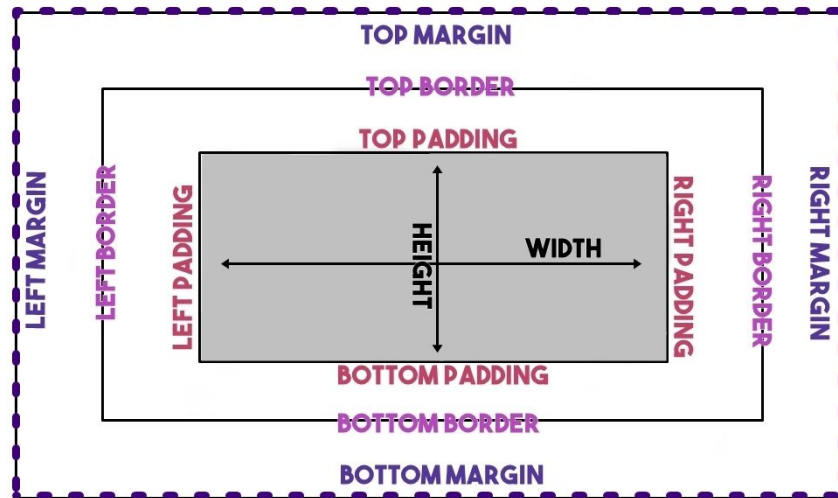
En el siguiente ejemplo nos encontramos con un **solapamiento de márgenes**. Por defecto, si tenemos dos elementos adyacentes con, por ejemplo, **margin: 20px** cada uno, ese espacio de margen se solapará y tendremos **20px** en total, y no **40px** (la suma de cada uno) como podríamos pensar en un principio. [+info](#)

Relleno (padding)

Los rellenos (**padding**) son los espacios que hay entre los bordes del elemento en cuestión y el contenido (por la parte interior).





Las opciones son:

- padding-top
- padding-right
- padding-bottom
- padding-left



Relleno (padding)

Al igual que con los márgenes, los rellenos tienen varias propiedades para indicar cada zona:

Propiedad	Valor	Significado
<code>padding-top</code>	0 	Aplica un relleno interior en el espacio superior de un elemento.
<code>padding-left</code>	0 	Aplica un relleno interior en el espacio izquierdo de un elemento.
<code>padding-right</code>	0 	Aplica un relleno interior en el espacio derecho de un elemento.
<code>padding-bottom</code>	0 	Aplica un relleno interior en el espacio inferior de un elemento.

Como se puede ver en la tabla, por defecto no hay relleno (*el relleno está en cero*), aunque puede modificarse tanto con las propiedades anteriores como la propiedad de atajo: Modelo de caja (en la siguiente página) [+info](#)

Modelo de caja - Atajos

CSS proporciona atajos para los márgenes y los rellenos:

Propiedad	Valores	Significado
<code>margin</code> o <code>padding</code>	<code>SIZE</code>	1 parámetro. Aplica el mismo margen a todos los lados.
	<code>SIZE</code> <code>SIZE</code>	2 parámetros. Aplica margen top/bottom y left/right .
	<code>SIZE</code> <code>SIZE</code> <code>SIZE</code>	3 parámetros. Aplica margen top , left/right y bottom .
	<code>SIZE</code> <code>SIZE</code> <code>SIZE</code> <code>SIZE</code>	4 parámetros. Aplica margen top , right , bottom e left .

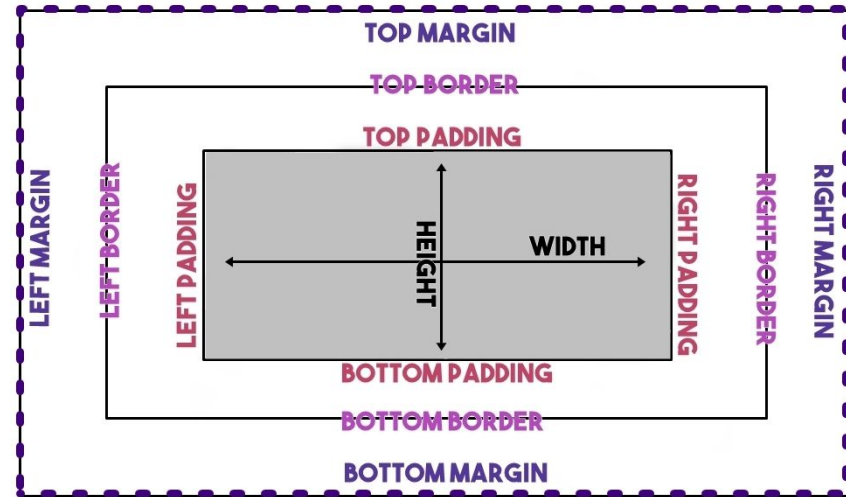
margin o padding:	10px; top/right/bottom/left			
	top/bottom	right/left		
	10px	20px;		
	top	right/left	bottom	
	10px	20px	10px;	
	top	right	bottom	left
	10px	20px	10px	20px;

Bordes

Borde (border)

Establece un límite entre la parte interior y la parte exterior de la caja. Se pueden especificar estilo, ancho y color. Las opciones son:

- border-top
- border-right
- border-bottom
- border-left



Borde (border)

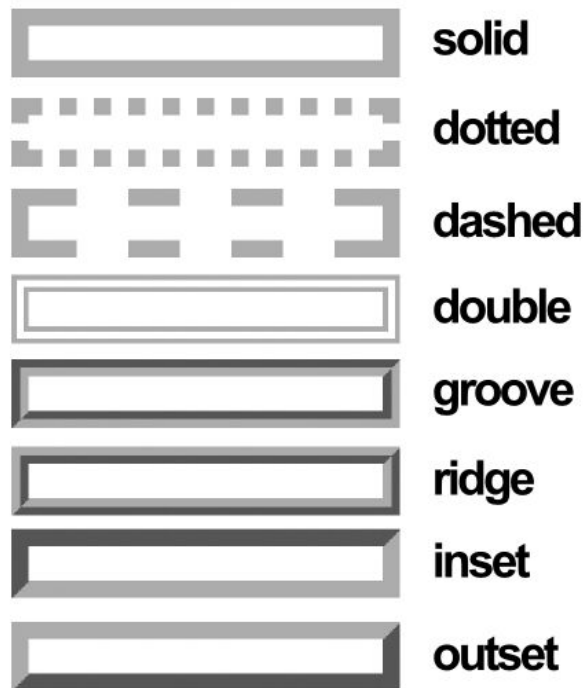
Las propiedades básicas y específicas de los bordes en CSS son las siguientes:

Propiedad	Valor	Significado
<code>border-color</code> <small>1234</small>	<code>black</code> <code>COLOR</code>	Especifica el color que se utilizará en el borde.
<code>border-width</code> <small>1234</small>	<code>thin</code> <code>medium</code> <code>thick</code> <code>SIZE</code>	Especifica un tamaño predefinido para el grosor del borde.
<code>border-style</code> <small>1234</small>	<code>none</code> <code>STYLE</code>	Define el estilo para el borde a utilizar

El estilo de borde más frecuente es **solid** (borde liso y continuo), y que además es la opción por defecto. Pueden utilizarse cualquiera de los estilos indicados en la tabla anterior e incluso combinar con otras propiedades. [+info](#)

Borde (border)

- **hidden:** Oculto. Idéntico a none, salvo para conflictos con tablas.
- **dotted:** Borde basado en puntos.
- **dashed:** Borde basado en rayas (línea discontinua).
- **solid:** Borde sólido (línea continua).
- **double:** Borde doble (dos líneas continuas).
- **groove:** Borde biselado con luz desde arriba.
- **ridge:** Borde biselado con luz desde abajo. Opuesto a groove.
- **inset:** Borde con profundidad «hacia dentro».
- **outset:** Borde con profundidad «hacia fuera». Opuesto a inset.



Bordes múltiples

Sólo hemos utilizado un parámetro en cada propiedad, aplicando el mismo valor a cada borde de un elemento. Sin embargo, podemos especificar de uno a cuatro parámetros, dependiendo de lo que queramos hacer:

Propiedad	Valor	Significado
<code>border-color</code> 1234	COLOR	1 parámetro. Aplica el mismo color a todos los bordes.
	COLOR COLOR	2 parámetros. Aplica al borde top/bottom , y al left/right .
	COLOR COLOR COLOR	3 parámetros. Aplica al top , al left/right y al bottom .
	COLOR COLOR COLOR COLOR	4 parámetros. Aplica al top , right , bottom y left .

Podemos hacer lo mismo con las propiedades **border-width** y **border-style**. Teniendo en cuenta esto, disponemos de una gran flexibilidad a la hora de especificar esquemas de bordes más complejos.

Bordes múltiples

En el ejemplo utilizamos 3 parámetros, indicando un elemento con borde superior rojo sólido de 2 píxeles de grosor, con borde izquierdo y derecho doble azul de 10 píxeles de grosor y con un borde inferior verde sólido de 5 píxeles de grosor.

```
div {  
  border-color: red blue green;  
  border-width: 2px 10px 5px;  
  border-style: solid double solid;  
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Con la propiedad **border-width** pasa exactamente lo mismo que con **margin** y **padding**, actuando en este caso en relación al grosor del borde de un elemento. Se pueden utilizar de 1 a 4 parámetros.

Bordes - Atajos

Con tantas propiedades, incluso para hacer algo sencillo necesitamos varias líneas de código. Pero podemos utilizar la propiedad de atajo **border**, con la que podemos hacer un resumen sin necesidad de indicar múltiples propiedades individuales por separado, realizando el proceso de forma más corta:

Propiedad	Valor	Significado
border	SIZE STYLE COLOR	Propiedad de atajo para simplificar valores.

```
div {  
  border: 1px solid #000000;  
}
```

Lorem ipsum dolor sit amet, con
labore et dolore magna aliqua.

Bordes específicos

Una forma más intuitiva, es utilizar las propiedades de bordes específicos (por zonas) y aplicar estilos combinándolos junto a la herencia de CSS. Para utilizarlas bastaría con indicar la zona justo después de border-:

```
div {  
  border-bottom-width: 2px;  
  border-bottom-style: dotted;  
  border-bottom-color: black;  
}
```

Lorem ipsum dolo
labore et dolore m
.....

```
div {  
  border: 5px solid red;  
  border-top-width: 15px;  
  border-top-color: orange;  
  border-top-style: solid; /* se hereda */  
}
```

Lorem ipsum dolor sit ame
labore et dolore magna alic

Box-sizing

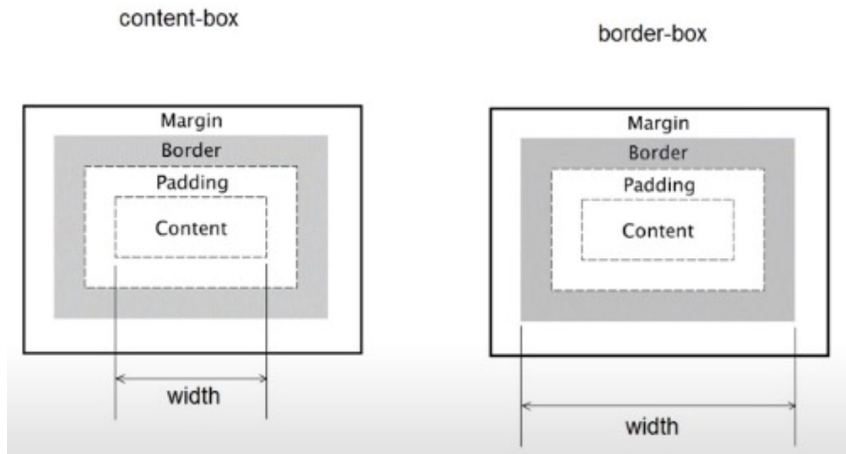
Indica cómo se debe calcular el ancho y el alto total de un elemento. Acepta los valores:

- **box-sizing: content-box:** Es el valor que cualquier caja tiene asignada por defecto. Las propiedades width y height no incluyen el borde, padding o margin.
- **box-sizing: border-box:** Las propiedades width y height incluyen el contenido, padding y borde pero no el margin.
- **box-sizing: initial:** Establece esta propiedad en su valor predeterminado. [+info](#)
- **box-sizing: inherit:** Hereda esta propiedad de su elemento padre. [+info](#)

En el modelo de caja CSS “clásico”, el borde y los márgenes interior y exterior se añaden al tamaño del elemento definido con las propiedades width y height.

Box-sizing

La propiedad box-sizing, permite modificar este comportamiento y hacer que el borde y los márgenes interior y exterior se puedan incluir en el interior del tamaño definido con las propiedades **width** y **height**. En este caso se reducirá el espacio disponible para el contenido. [+info](#)



Posicionamiento

Los elementos en línea y en bloque nos permiten diseñar esquemas complejos, alineando y combinando elementos.

El posicionamiento por defecto es el **estático** (*static*): todos los elementos aparecen con un orden natural según donde estén colocados en el HTML.

No obstante, CSS permite al diseñador modificar la posición en la que se muestra cada caja.

Existen otros modos alternativos de posicionamiento, que podemos cambiar mediante la propiedad **position**, que nos permiten modificar la posición de los elementos y su contenido.

Utilizando las propiedades que proporciona CSS para alterar la posición de las cajas es posible realizar efectos muy avanzados y diseñar estructuras de páginas que de otra forma no serían posibles.

Posicionamiento

A la propiedad `position` se le pueden indicar los siguientes valores:

- **static:** es el valor por defecto, un elemento con este valor no está posicionado con CSS.
- **relative:** mediante las propiedades `top` | `bottom` | `right` y/o `left` el elemento se posiciona en forma relativa a su contenedor.
- **absolute:** hace que un elemento se ubique considerando su contenedor posicionado más cercano. Si no encuentra ningún contenedor cercano, el elemento se colocará respecto al viewport. El resto de elementos de la página ignoran la nueva posición del elemento.
- **fixed:** la caja está posicionada con respecto a la ventana del navegador. Se mantiene en el mismo lugar incluso al hacer scroll en la página. La referencia es el `viewport`, la parte visible del navegador. [+info](#)
- **sticky:** al realizar un scroll y después de alcanzar una posición de desplazamiento determinada, se “pega” al borde elegido. [+info](#)

Posicionamiento

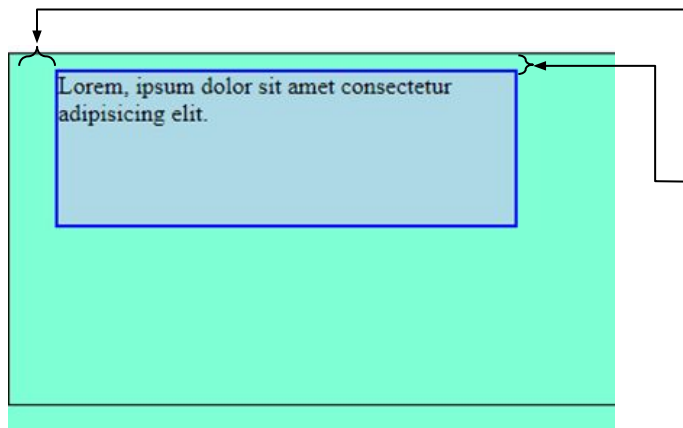
Si utilizamos un modo de posicionamiento diferente al estático (*absolute*, *fixed*, *sticky* o *relative*), podemos emplear una serie de propiedades para modificar la posición de un elemento. Estas propiedades son las siguientes:

Propiedad	Valor	Significado
top:	auto <small>SIZE</small>	Empuja el elemento una distancia desde la parte superior hacia el inferior.
bottom:	auto <small>SIZE</small>	Empuja el elemento una distancia desde la parte inferior hacia la superior.
left:	auto <small>SIZE</small>	Empuja el elemento una distancia desde la parte izquierda hacia la derecha.
right:	auto <small>SIZE</small>	Empuja el elemento una distancia desde la parte derecha hacia la izquierda.
z-index:	auto <small>NUMBER</small>	Coloca un elemento en el eje de profundidad, más cerca o más lejos del usuario.

Las propiedades **top**, **bottom**, **left** y **right** sirven para mover un elemento desde la orientación que su propio nombre indica hasta su extremo contrario. Esto es, si utilizamos **left** e indicamos **20px**, estaremos indicando mover **desde la izquierda** 20 píxeles hacia la **derecha**.

Posicionamiento relativo (relative)

Si utilizamos la palabra clave **relative** activaremos el modo de posicionamiento relativo. En este modo, los elementos se colocan exactamente igual que en el posicionamiento estático (permanecen en la misma posición), pero dependiendo del valor de las propiedades *top*, *bottom*, *left* o *right* variamos la posición del elemento.



```
.contenedor {  
  position: relative;  
  left: 30px;  
  top: 10px;  
  border: 2px solid blue;  
  background-color: lightblue;  
  width: 300px;  
  height: 100px;  
  margin: 0px;  
}
```

Posicionamiento absoluto (absolute)

Utilizando la palabra clave **absolute** indicamos que el elemento utiliza **posicionamiento absoluto**. Este tipo de posicionamiento coloca los elementos utilizando como punto de origen el primer contenedor con posicionamiento diferente a estático. Si no existe, emplea el documento completo como referencia.

Si el contenedor padre tiene posicionamiento estático, se analiza el posicionamiento del padre del contenedor padre, y así sucesivamente hasta encontrar un contenedor con posicionamiento no estático o llegar a la etiqueta `<body>`, situación en la que se comportaría como en el ejemplo.

Ejemplo: Si establecemos `left:40px`, el elemento se colocará 40 píxeles a la derecha del extremo izquierdo de la página. Sin embargo, si indicamos `right:40px`, el elemento se colocará 40 píxeles a la izquierda del extremo derecho de la página.

Posicionamiento fijo (fixed) y “pegajoso” (sticky)

El posicionamiento **fijo (fixed)** hace que el elemento se muestre en una posición fija **dependiendo de la región visual del navegador**. Es decir, aunque el usuario haga scroll y se desplace hacia abajo en la página web, el elemento seguirá en el mismo sitio posicionado.

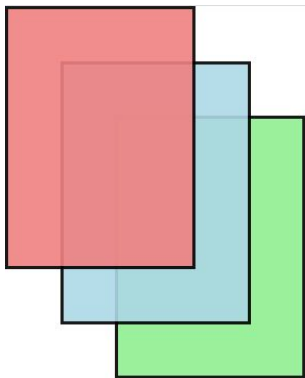
Ejemplo: Si establecemos **top:0** y **right:0**, el elemento se colocará justo en la esquina superior derecha y se mantendrá ahí aunque hagamos scroll hacia abajo en la página.

El **posicionamiento sticky** se utiliza cuando queremos que un elemento se posicione en un lugar específico de forma fija («sticky», pegajoso).

Ejemplo: Al hacer scroll y llegar a un elemento este podrá quedarse fijo en la parte superior mientras continuamos haciendo scroll. No debemos confundir con el fijo ya que no queda en una posición fija, sino que flota respecto del fondo y se queda adherido a la parte superior.

Profundidad (z-index)

Establece el nivel de profundidad de un elemento sobre los demás. De esta forma, podremos superponer los elementos, quedando “apilados”. Los elementos se posicionan de acuerdo al nivel de profundidad, quedando “encima” los que poseen un valor de index mayor. [+info](#)



```
.box{  
  position: absolute;  
  width:100px;  
  height:140px;  
  border: 2px solid black;  
  opacity: 90%;  
}  
.z1 {  
  background-color: lightcoral;  
  left: 0px;  
  top: 0px;  
  z-index: 20;  
}
```

```
.z2 {  
  background-color: lightblue;  
  left: 30px;  
  top: 30px;  
  z-index: 10;  
}  
.z3 {  
  background-color: lightgreen;  
  left: 60px;  
  top: 60px;  
  z-index: 0;  
}
```

Propiedad visibility y display

La propiedad de **visibilidad** especifica si un elemento es visible o no.

Nota: los elementos ocultos ocupan espacio en la página. Puede usar la propiedad display para ocultar y eliminar un elemento del diseño del documento.

Sintaxis: `visibility: visible|hidden|collapse|initial|inherit;` [+info](#)

La propiedad **display** especifica el comportamiento de visualización de un elemento. Ejemplo: especifica si un elemento es tratado como block or inline.

En HTML, el valor de la propiedad de visualización predeterminada se toma de las especificaciones de HTML o de la hoja de estilo predeterminada del navegador/usuario.

Sintaxis: `display: value;` [+info](#)

Display

Cada elemento tiene un valor de display por defecto. Recordemos que los navegadores le dan a los elementos valores por defecto block e inline:

- **block**: el elemento empieza en una nueva línea (div, h1-h6, header, etc)
- **inline**: puede contener algo de texto dentro de un párrafo sin interrumpir el flujo del párrafo.
- **none**: es utilizado para ocultar elementos sin eliminarlos, no deja un espacio donde el elemento se encontraba.
- **inline-block**: Los elementos inline-block fluyen con el texto y demás elementos como si fueran elementos en línea y además respetan el ancho, el alto y los márgenes verticales.

Cada etiqueta HTML tiene un tipo de representación visual en un navegador, lo que habitualmente se suele denominar el *tipo* de caja. En principio, se parte de dos tipos básicos: **inline** y **block**.

Tipos de display

La tabla siguiente muestra una lista de los valores de la propiedad:

Tipo de caja	Características
block	Se apila en vertical. Ocupa todo el ancho disponible de su etiqueta contenedora.
inline	Se coloca en horizontal. Se adapta al ancho de su contenido. Ignora <code>width</code> o <code>height</code> .
inline-block	Combinación de los dos anteriores. Se comporta como <code>inline</code> pero no ignora <code>width</code> o <code>height</code> .
flex	Utiliza el modelo de cajas flexibles Flexbox . Muy útil para diseños adaptables.
inline-flex	La versión en línea (ocupa sólo su contenido) del modelo de cajas flexibles flexbox.
grid	Utiliza cuadrículas o rejillas con el modelo de cajas Grid CSS .
inline-grid	La versión en línea (ocupa sólo su contenido) del modelo de cajas grid css.
list-item	Actúa como un ítem de una lista. Es el comportamiento de etiquetas como <code></code> .
table	Actúa como una tabla. Es el comportamiento de etiquetas como <code><table></code> .
table-cell	Actúa como la celda de una tabla. Es el comportamiento de etiquetas como <code><th></code> o <code><td></code> .
table-row	Actúa como la fila de una tabla. Es el comportamiento de etiquetas como <code><tr></code> .

Ocultar elementos

En la lista anterior, falta un valor de la propiedad **display**. Mediante la mencionada propiedad, es posible aplicar un valor **none** y **ocultar completamente elementos** que no queramos que se muestren. Es muy útil para hacer desaparecer información cuando el usuario realiza alguna acción, por ejemplo.

Tipo de caja	Características
none	Hace desaparecer visualmente el elemento, como si no existiera.

Ocultar elementos

No obstante, también existe una propiedad CSS llamada **visibility** que realiza la misma acción, con la ligera diferencia de que no sólo oculta el elemento, sino que además mantiene un vacío con el mismo tamaño de lo que antes estaba ahí. Dicha propiedad **visibility** puede tomar los siguientes valores:

Valor	Significado
visible	El elemento es visible. Valor por defecto.
hidden	El elemento no es visible pero sigue ocupando su espacio y posición.
collapse	Sólo para tablas. El elemento se contrae para no ocupar espacio.

Ocultar elementos

Utilizar **visibility: hidden** es muy interesante si queremos que un elemento y su contenido se vuelva invisible, pero siga ocupando su espacio y así evitar que los elementos adyacentes se desplacen, lo que suele ser un comportamiento no deseado en algunas ocasiones cuando se aplica **display: none**.

Otra opción interesante es utilizar la propiedad **opacity** junto a transiciones o animaciones, desplazarse desde el valor **0** al **1** o viceversa. De esta forma conseguimos una animación de aparición o desvanecimiento. [+info](#)

Material extra

Artículos de interés

Propiedades de ancho y alto:

- [min-height](#), [max-height](#), [min-width](#) y [max-width](#)

Bordes:

- https://www.w3schools.com/css/css3_borders.asp
- <https://lenguajecss.com/css/modelo-de-cajas/border-radius/>
- <https://lenguajecss.com/css/introduccion/herencia-css/>

Modelo de caja:

- <https://www.mclibre.org/consultar/amaya/css/css-modelo-caja.html>

Artículos de interés

Posicionamiento:

- <https://developer.mozilla.org/es/docs/Web/CSS/position>
- <https://www.aprenderaprogramar.com>
- [Video sobre posicionamiento](#)
- https://aprende-web.net/css/css7_3.php

Juegos para aprender CSS:

- <http://cssgridgarden.com>
- <http://www.flexboxdefense.com>
- <https://flexboxfroggy.com>
- <https://cssbattle.dev>
- <https://flukeout.github.io/>

Tarea para el Proyecto:

- Seleccionar una paleta de colores acorde al proyecto.
- Registrar al menos 3 sitios web de referencia que guiarán los estilos del sitio.
- Crear un banco de imágenes para el proyecto.

No te olvides de dar el presente

Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.
- Realizar los Ejercicios obligatorios.

Todo en el Aula Virtual.

Muchas gracias por tu atención.

Nos vemos pronto