Fuente: lenguajehtml.com y lenguajecss.com

El concepto de **cascada**.

Este concepto es un poco más avanzado, por lo que se debe conocer bien el tema de **selectores CSS** y dominar algo de CSS para comprenderlo en su totalidad.

Antes de continuar, supongamos que nos encontramos ante el siguiente escenario:

```
<div>Texto del elemento</div>
<style>
        div {
          color: red;
          padding: 8px
        }
        div {
          color: blue;
          background-color: grey
        }
</style>
```

En este caso, ¿cuál de las dos reglas prevalece, si tenemos en cuenta que se refieren al mismo elemento y están al mismo nivel? La respuesta es muy fácil: Prevalece siempre la última regla definida, la cual **mezcla** y **sobreescribe** las propiedades anteriores.

En el caso anterior, el resultado final (*computado*) sería el siguiente:

```
div {
  color: blue; /* Se sobreescribe la última */
  padding: 8px;
  background-color: grey;
}
```

Sin embargo, puede ocurrir que en determinados casos no esté tan claro cuál es el estilo que debería sobrescribir a los anteriores. Ahí es cuando entra en juego el concepto de **cascada en CSS**, que es el que se encarga de eliminar la ambigüedad y determinar el que tiene prioridad. Supongamos el siguiente caso:

```
<div id="nombre" class="clase">Texto del elemento</div>

<style>
        div {
          color: red;
        }

        #nombre {
          color: blue;
        }

        .clase {
          color: green;
        }
</style>
```

Resultado:
Texto del elemento

¿Cómo sabe CSS que estilo aplicar? ¿Cuál tiene prioridad sobre los demás? Aquí es donde entra en acción el concepto de **cascada en CSS**.

Cascada CSS

Para saber qué bloque de estilos tiene prioridad, CSS analiza (*por orden*) tres conceptos clave del código CSS: su **importancia**, la **especificidad** y su **orden**. Veamos en que se basa cada uno de ellos.

Importancia

La **importancia** de un código CSS se determina dependiendo de las hojas de estilo donde está colocado. Generalmente, no necesitaremos preocuparnos de este factor, pero siempre es una buena idea conocer cómo funciona. Existen varios tipos de hojas de estilo, de menor a mayor importancia:

Tipo de CSS	Descripción	Definido por
Agente de usuario	Son los estilos CSS que aplica el navegador por defecto.	Navegador
CSS de usuario	Son los estilos CSS que añade el usuario, por razones específicas.	Usuario
CSS de autor	Son los estilos CSS que coloca el autor de la página.	Desarrollador

Aunque no es recomendable utilizarlo frecuentemente, se puede añadir al final de cada regla el texto !**important**, consiguiendo que la regla en cuestión tenga prioridad sobre las demás, independientemente del nivel o la altura a la que estén:

```
<div>Texto del elemento</div>

<style>
        div {
          color: red !important;
          padding: 8px
        }
```

```
        div {
          color: blue;
          background-color: grey
        }
</style>
```

El resultado final de este código CSS sería:

```
div {
  color: red;
  padding: 8px;
  background-color: grey
}
```

Resultado:

Texto del elemento

Especificidad

En segundo caso, y si la importancia no elimina la ambigüedad, se pasa a determinar la especificidad de los selectores CSS. Para ello, se sigue un cálculo matemático basado en 4 componentes: **A**, **B**, **C**, **D**.

Componente	Descripción
Componente A	Número de estilos aplicados mediante un atributo **style**.
Componente B	Número de veces que aparece un **id** en el selector.

Componente C	Número de veces que aparece una **clase**, **pseudoclase** o **atributo** en el selector.
Componente D	Número de veces que aparece un **elemento** o un **pseudoelementos** en el selector.

Para saber si un bloque de CSS es más específico que otro (*y, por lo tanto, tiene prioridad*) sólo hay que calcular sus componentes. Se ordenan teniendo en cuenta los valores de cada componente, de izquierda a derecha.

Veamos algunos ejemplos, ordenados de **menor a mayor especificidad**:

```
div { ... }                    /* Especificidad: 0,0,0,1 */
div div { ... }                /* Especificidad: 0,0,0,2 */
#pagina div { ... }            /* Especificidad: 0,1,0,1 */
#pagina div:hover { ... }      /* Especificidad: 0,1,1,1 */
#pagina div:hover a { ... }    /* Especificidad: 0,1,1,2 */
#pagina .sel:hover>a { ... }   /* Especificidad: 0,1,2,1 */
```

Ver: https://specifishity.com/

Orden

En CSS, es posible crear múltiples reglas CSS para definir un mismo concepto. En este caso, la que prevalece ante todas las demás depende de ciertos factores, como es la *«altura»* a la que está colocada la regla:

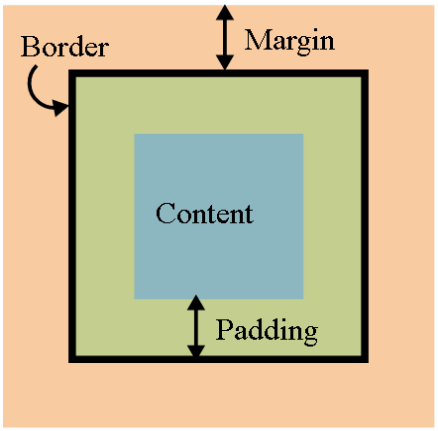
- El **CSS embebido** en un elemento HTML es el que tiene mayor precedencia, por lo que siempre será el que tenga prioridad sobre otras reglas CSS.
- En segundo lugar, el **CSS interno** definido a través de bloques **style** en el propio documento HTML será el siguiente a tener en cuenta en orden de prioridad.
- Por último, los documentos **CSS externos** son la tercera opción de prioridad a la hora de tomar en cuenta las reglas CSS.

Teniendo esto en cuenta, hay que recordar que las propiedades que prevalecerán serán las que estén en último lugar, siempre respetando la prioridad de la lista anterior.

Modelo de cajas

Durante varios años, el denominado **modelo de cajas** fue una pesadilla para los desarrolladores web, puesto que se mostraba visualmente de forma diferente en **Internet Explorer** respecto a los demás navegadores. Por fortuna, todos los navegadores actuales ya interpretan de la misma forma el modelo de cajas, pero conviene aprender bien la diferencia para no ser como Internet Explorer.

La representación básica del **modelo de cajas** 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 **márgen** (*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** (*en azul*). En azul, es la parte interior del elemento, excluyendo el relleno.

Dimensiones (ancho y alto)

Para dar tamaños específicos a los diferentes elementos de un documento HTML, necesitaremos asignarles valores a las propiedades **width** (ancho) y **height** (alto).

Propiedad	Valor	Significado
width	auto |	Tamaño de ancho de un elemento.
height	auto |	Tamaño de alto de un elemento.

En el caso de utilizar el valor **auto** en las propiedades anteriores (*que es lo mismo que no indicarlas, ya que es el valor que tienen por defecto*), el navegador se encarga de calcular el ancho o alto necesario, dependiendo del contenido del elemento. Esto es algo que también puede variar, dependiendo del tipo de elemento que estemos usando, y que veremos más adelante, en el apartado de maquetación.

Hay que ser muy conscientes de que, sin indicar valores de ancho y alto para la caja, el elemento generalmente toma el tamaño que debe respecto a su contenido, mientras que, si indicamos un ancho y alto concretos, **estamos obligando a CSS tener un aspecto concreto** y podemos obtener resultados similares al siguiente (*conocida broma de CSS*) si su contenido es más grande que el tamaño que hemos definido:

Otra forma de lidiar con esto, es 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 unos máximos y unos mínimos, donde el ancho o alto podría variar entre esos valores.

```
div {
  width: 800px;
  height: 400px;
  background: red;
  max-width: 500px;
}
```

En este caso, por ejemplo, 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.

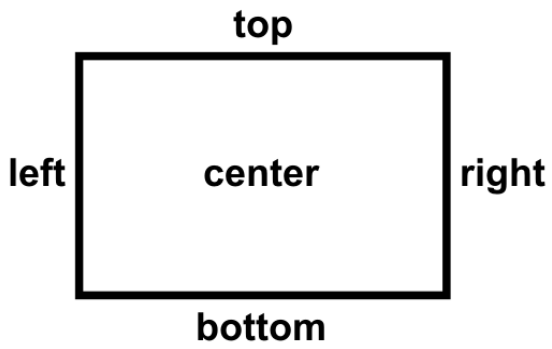
Por un lado, tenemos las propiedades de mínimos **min-width** y **min-height**, que por defecto tienen valor **0**, mientras que, por otro lado, tenemos las propiedades de máximos **max-width** y **max-height**, que por defecto tienen valor **none**:

Propiedad	Propiedad	Significado
max-width	**none** |	Ancho máximo que puede ocupar un elemento.
min-width	**0** |	Ancho mínimo que puede ocupar un

		elemento.
max-height	**none** |	Alto máximo que puede ocupar un elemento.
min-height	**0** |	Alto mínimo que puede ocupar un elemento.

Zonas de un elemento

Antes de continuar, es importante saber que en CSS existen ciertas palabras clave para hacer referencia a una zona u orientación concreta sobre un elemento. Son conceptos muy sencillos y prácticamente lógicos, por lo que no tendrás ningún problema en comprenderlos. Son los siguientes:



- **Top**: Se refiere a la parte superior del elemento.
- **Left**: Se refiere a la parte izquierda del elemento.
- **Right**: Se refiere a la parte derecha del elemento.
- **Bottom**: Se refiere a la parte inferior del elemento.
- **Center**: En algunos casos se puede especificar el valor **center** para referirse a la posición central entre los extremos horizontales o verticales.

Desbordamiento

Volvamos a pensar en la situación de la imagen anterior: Damos un tamaño de ancho y alto a un elemento HTML, pero su contenido de texto es tan grande que no cabe dentro de ese elemento. ¿Qué ocurriría? Probablemente lo que vimos en la imagen: el contenido se desbordaría.

Podemos modificar ese comportamiento con la propiedad de CSS **overflow**, o con alguna de sus propiedades específicas **overflow-x** o **overflow-y**:

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 (horizontal).
overflow-y	visible | hidden | scroll | auto	Establece el desbordamiento sólo para el eje Y (vertical).

Dichas propiedades pueden tomar varios valores, donde **visible** es el valor que tiene por defecto, que permite que haya desbordamiento. Otras opciones son las siguientes, donde **no se permite desbordamiento**:

Propiedad	**Valor**	**Significado**
visible	Se muestra el contenido que sobresale (comportamiento por defecto)	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

> **Nota**: CSS3 añade las propiedades **overflow-x** y **overflow-y** para cada eje individual, que antiguamente solo era posible hacerlo con **overflow** para ambos ejes. Estas propiedades son útiles cuando no quieres mostrar alguna barra de desplazamiento, habitualmente, la barra de desplazamiento horizontal.

Márgenes y rellenos

En el modelo de cajas, los **márgenes** (*margin*) son los espacios exteriores de un elemento. El espacio que hay entre el borde de un elemento y el borde de otros elementos adyacentes, es lo que se considera márgen.

Márgenes

Dichos márgenes se pueden considerar en conjunto (*de forma general*) o de forma concreta en cada una de las zonas del elemento. Veamos primero las propiedades específicas para cada zona:

Propiedad	Valor	Significado
margin-top	**auto** |	Establece un tamaño de margen superior.
margin-left	**auto** |	Establece un tamaño de margen a la izquierda.
margin-right	**auto** |	Establece un tamaño de margen a la derecha.
margin-bottom	**auto** |	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**.

> **Truco**: Existe un truco muy sencillo y práctico para centrar un elemento en pantalla. Basta con aplicar un ancho fijo al contenedor, **width:500px** (*por ejemplo*) y luego aplicar un **margin:auto**. De esta forma, el navegador, al conocer el tamaño del elemento (*y por omisión, el resto del tamaño de la ventana*) se encarga de repartirlo equitativamente entre el margen izquierdo y el margen derecho, quedando centrado el elemento.

Hay que recordar diferenciar bien los **márgenes** de los **rellenos**, puesto que no son la misma cosa. Los **rellenos** (*padding*) son los espacios que hay entre los bordes del elemento en cuestión y el contenido del elemento (*por la parte interior*). Mientras que los márgenes (*margin*) son los espacios que hay entre los bordes del elemento en cuestión y los bordes de otros elementos (*parte exterior*).

Obsérvese también el siguiente ejemplo para ilustrar el **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,

Rellenos

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

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

Atajo: Modelo de cajas

Al igual que en otras propiedades de CSS, también existen atajos para los márgenes y los rellenos:

Propiedad	Valor	Significado
margin		1 parámetro. Aplica el mismo margen a **todos** los lados.
		2 parámetros. Aplica margen **top/bottom** y **left/right**.
		3 parámetros. Aplica margen **top**, **left/right** y **bottom**.
		4 parámetros. Aplica margen **top**, **right**, **bottom** e **left**.

Con las propiedades padding y border-width pasa exactamente lo mismo, actuando en relación a los rellenos, en lugar de los márgenes en el primer caso, y en relación al grosor del borde de un elemento en el segundo.

Ojo: Aunque al principio es muy tentador utilizar márgenes negativos para ajustar posiciones y colocar los elementos como queremos, se aconseja no utilizar dicha estrategia salvo para casos muy particulares, ya que a la larga es una mala práctica que hará que nuestro código sea de peor calidad

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

Propiedad	**Valor**	**Significado**
border-color		Especifica el color que se utilizará en el borde.
border-width	thin | **medium** | thick |	Especifica un tamaño predefinido para el grosor del borde.
border-style	**none** |	Define el estilo para el borde a utilizar (ver más adelante).

En primer lugar, **border-color** establece el color del borde, de la misma forma que lo hicimos en apartados anteriores de colores. En segundo lugar, con **border-width** podemos establecer la anchura o grosor del borde utilizando tanto **palabras clave** predefinidas como un tamaño concreto con cualquier tipo de las **unidades** ya vistas.

Estilos de borde

Por último, con **border-style** podemos aplicar un estilo determinado al borde de un elemento. En **estilo de borde** podemos elegir cualquiera de las siguientes opciones:

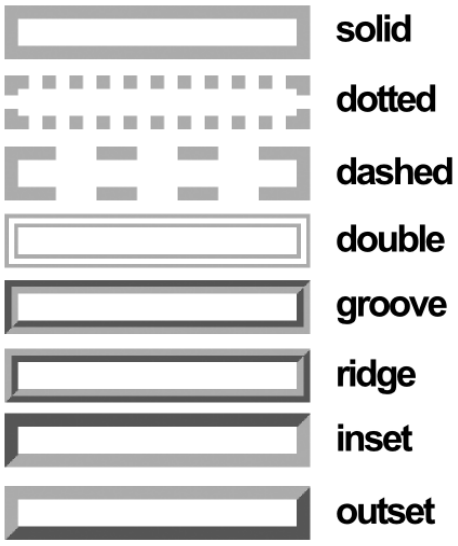
Propiedad	Valor	Significado
hidden	Oculto. Idéntico al anterior salvo para conflictos con tablas.	
dotted	Establece un borde basado en puntos.	
dashed	Establece un borde basado en rayas (línea discontínua).	
solid	Establece un borde sólido (línea contínua).	
double	Establece un borde doble (dos líneas contínuas).	
groove	Establece un borde biselado con luz desde arriba.	
ridge	Establece un borde biselado con luz desde abajo. Opuesto a **groove**.	
inset	Establece un borde con profundidad «hacia dentro».	

outset	Establece un borde con profundidad «hacia fuera». Opuesto a **inset**.	

Veamos un ejemplo sencillo:

```
div {
  border-color: gray;
  border-width: 1px;
  border-style: dotted;
}
```

Sin embargo, el borde más frecuente suele ser **solid**, que no es más que un borde liso. Pueden utilizarse cualquiera de los estilos indicados en la tabla anterior. Veamos cómo se verían los diferentes estilos de borde utilizando **10 píxeles** de grosor y color **gris**:



Bordes múltiples (diferentes)

Hasta ahora, sólo hemos utilizado un parámetro en cada propiedad, lo que significa que se aplica el mismo valor para cada borde de un elemento (*borde superior, borde derecho, borde inferior y borde izquierdo*). Sin embargo, podemos especificar uno, dos, tres o cuatro parámetros, dependiendo de lo que queramos hacer:

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

De la misma forma, podemos hacer exactamente lo mismo con las propiedades **border-width** (*respecto al ancho del borde*) y **border-style** (*respecto al estilo del borde*). Teniendo en cuenta esto, disponemos de mucha flexibilidad a la hora de especificar esquemas de bordes más complejos:

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

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

Atajo: Bordes

Pero ya habremos visto que con tantas propiedades, para hacer algo relativamente sencillo, nos pueden quedar varias lineas de código complejas y difíciles de leer. Al igual que con otras propiedades CSS, podemos utilizar la propiedad de atajo **border**, con la que podemos hacer un resumen y no necesitar utilizar las propiedades individuales por separado, realizando el proceso de forma más corta:

Propiedad	Valor	Significado
border		Propiedad de atajo para simplificar valores.

Por ejemplo:

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

Así pues, estamos aplicando un borde de **1 píxel** de grosor, estilo **sólido** y color **negro** a todos los bordes del elemento, ahorrando mucho espacio y escribiéndolo todo en una sola propiedad.

> **Consejo**: Intenta organizarte y aplicar siempre los atajos si es posible. Ahorrarás mucho espacio en el documento y simplificarás la creación de diseños. El orden, aunque no es obligatorio, si es recomendable para mantener una cierta coherencia con el estilo de código.

Bordes específicos

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

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

Esto dibujaría **sólo un borde inferior** negro de 2 píxeles de grosor y con estilo punteado. Ahora imaginemos que queremos un elemento con todos los bordes en rojo a 5 píxeles de grosor, salvo el borde superior, que lo queremos con un borde de 15 píxeles en color naranja. Podríamos hacer lo siguiente:

```
div {
  border: 5px solid red;
  border-top-width: 15px;
  border-top-color: orange;
  border-top-style: solid;  /* Esta propiedad no es necesaria (se hereda) */
}
```

El ejemplo anterior conseguiría nuestro objetivo. La primera propiedad establece todos los bordes del elemento, sin embargo, las siguientes propiedades modifican sólo el borde superior, cambiándolo a las características indicadas.

Recuerda que también existen atajos para estas propiedades de bordes en zonas concretas, lo que nos permite simplificar aún más el ejemplo anterior, haciéndolo más fácil de comprender:

```
div {
  border: 5px solid red;
  border-top: 15px solid orange;
}
```

> **Ojo**: Es muy importante entender como se está aplicando la herencia en los ejemplos anteriores, puesto que es una de las características más complejas de dominar de CSS junto a la cascada. Por ejemplo, si colocaramos el **border-top** antes del **border**, este último sobreescribiría los valores de **border-top** y no funcionaría de la misma forma.

Box-Sizing

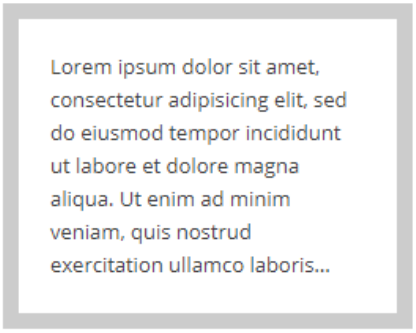
Fuente: emiliocobos.net

box-sizing es una propiedad CSS para cambiar el modelo de caja por defecto de los navegadores.

El ancho de un elemento se altera si se le aplica un borde o un padding. Eso es porque **la anchura del elemento que tu especificas con CSS, por defecto no incluye borde ni padding**.

Un ejemplo: Éste es el efecto que tiene un padding y un borde sobre un elemento de 200px de ancho:

```
<div style="width:200px;
    padding: 20px;
    border: 10px solid #ccc;
    margin: 0 auto;">
Lorem ipsum...
</div>
```



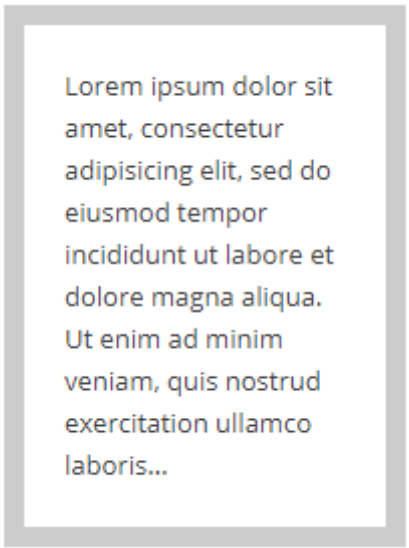
Como se puede comprobar, no mide 200px de ancho, sino **260px**. Es decir: 200px de ancho inicial, más 20px de padding izquierdo, más 20px de padding derecho, más 10px de borde izquierdo, más 10px de borde derecho.

Éste es el modo en el que los navegadores tratan los anchos por defecto. Sería el equivalente a **box-sizing: content-box;**

box-sizing: border-box

Con border-box, hacemos que el ancho especificado sea el equivalente al **ancho total**. Es decir:

```
<div style="width: 200px;
    padding: 20px;
    border: 10px solid #ccc;
    box-sizing: border-box;
    margin: 0 auto;">
Lorem ipsum...
</div>
```



Como se puede comprobar, ahora ese elemento **exactamente 200px**, ni uno más, ni uno menos.

Esto es **muy útil para elementos fluidos**, cuando necesitas que el elemento ocupe (por ejemplo) un 33% del ancho, y si ocupa un píxel más toda la estructura se estropearía.