

Tipos de selectores CSS

Selector de tipo `h1 { }`

Selector universal `* { }`

Selector de clase `.box { }`

Selector de ID `#unique { }`

Selector de atributo `a[href="https://example.org"]`

Pseudoclase `p:first-child { }`

Pseudoelemento `p::first-line { }`

Operadores de combinación descendentes `article p`

Operador de combinación de elementos hijo `article > p` (Solo afecta a los <p> hijos de <article>)

Operador de combinación de elementos hermanos adyacentes `h1 + p`

Operador de combinación general de elementos hermanos `h1 ~ p`

Agrupación de selectores `h1,p`

Especificidad

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**:

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









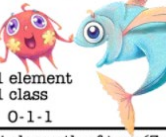

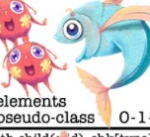
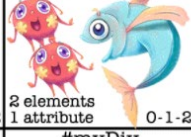
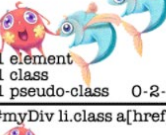
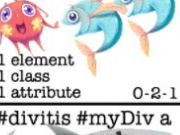
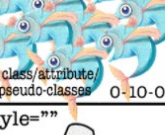
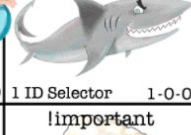
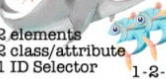



Selector	No of IDs	No of Classes	No of Types	Specificity	Winner
<code>p</code>	0	0	1	0-0-1	
<code>.foo</code>	0	1	0	0-1-0	
<code>#bar</code>	1	0	0	1-0-0	
<code>p.foo#bar</code>	1	1	1	1-1-1	✓

Selector	No of IDs	No of Classes	No of Types	Specificity	Winner
p	0	0	1	0-0-1	
p:last-child	0	1	1	0-1-1	
p.foo.bar.baz	0	3	1	0-3-1	
#bar	1	0	0	1-0-0	✓

Selector	No of IDs	No of Classes	No of Types	Specificity	Winner
a:not(.ul)	0	1	1	0-1-1	✓
nav a	0	0	2	0-0-2	

CSS SPECIFISHITY

WITH PLANKTON, FISH AND SHARKS

*  universal selector 0-0-0	div  1 element 0-0-1	li > ul  2 elements 0-0-2	body div ...ul li p a  12 elements 0-0-12
.myClass  1 class 0-1-0	*.myClass  1 universal selector 1 class 0-1-0	[type=checkbox]  1 attribute selector 0-1-0	:only-of-type  1 pseudo-class 0-1-0
li.myClass  1 element 1 class 0-1-1	li[attr]  1 element 1 attribute 0-1-1	li:nth-of-type(3n)~li  2 elements 1 pseudo-class 0-1-2	form input[type=email]  2 elements 1 attribute 0-1-2
li.class:nth-of-type(3n)  1 element 1 class 1 pseudo-class 0-2-1	input[type]:not(.class)  1 element 1 class 1 attribute 0-2-1	cl:nth-child(4n+1):checked[type=...]  10 class/attribute/pseudo-classes 0-10-0	#myDiv  1 ID Selector 1-0-0
#myDiv li.class a[href]  2 elements 2 class/attribute 1 ID Selector 1-2-2	#divitis #myDiv a  2 ID Selectors 1 type selector 2-0-1	style=""  inline style 1-0-0-0	!important  !important 1-0-0-0-0

Fuente: <https://specifishity.com/>

Box-Sizing

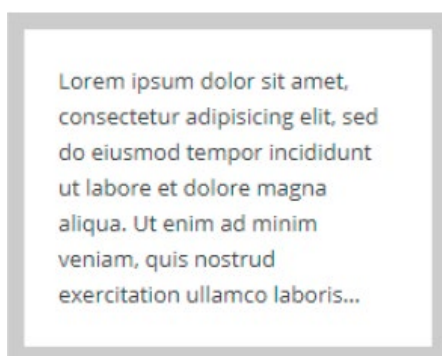
Fuente: emiliocobos.net

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>
```



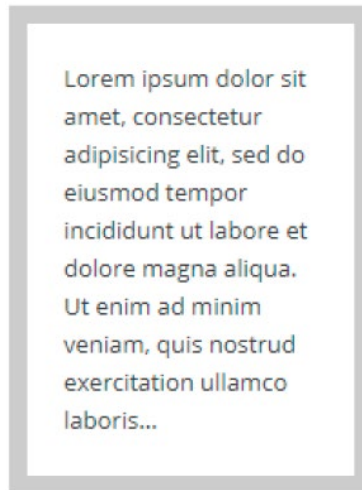
Como se puede comprobar, no mide 200px de ancho, sino **260px**. Es decir: 200px de ancho inicial, más 20px de padding izquierdo, más 20px de padding derecho, más 10px de borde izquierdo, más 10px de borde derecho.

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.

Snippet – Set de todos los componentes a border-box

Setea todos los box-sizing de los componentes (Incluidos sus hijos) a **border-box**.

```
html {  
    box-sizing: border-box;  
}  
*, *:before, *:after {  
    box-sizing: inherit;  
}
```

Fuente: <https://css-tricks.com/inheriting-box-sizing-probably-slightly-better-best-practice/>

Unidades de Medida

Unidades absolutas

Se llaman absolutas porque su valor es el declarado. No hay cálculo del mismo. Es el que es y de ahí no se mueve

Px: Los píxeles son unidades de tamaño fijo. Un pixel es igual a un punto en la pantalla del ordenador (la división más pequeña de la resolución de su pantalla) y por lo tanto es indivisible.

Su problema es que no es escalable, atentado contra la accesibilidad. Era una cuestión que muchos obviaban porque les permitía un control *"al milímetro"* de sus realizaciones.

Pero hoy a esa falla en la accesibilidad por algunos colectivos se suma el comportamiento en dispositivos de pequeñas dimensiones y grandes resoluciones.

Al usar pixeles para definir las dimensiones de los objetos y los estilos de fuente no solo estamos definiendo tamaños rígidos, sino que también estamos ignorando las configuraciones que cada usuario pueda tener en su navegador.

Ni permite escalar a más los textos a los usuarios con problemas visuales ni a menos en dispositivos como los smartphones.

Pt: El punto es una herencia de los medios impresos. Un punto es igual a 1/72 de pulgada (según diversas fuentes), aprox. 0'04mm.

Al ser también una medida absoluta tiene todos los problemas de los px aumentados.

Otras unidades absolutas:

- mm, cm, in: Milímetros, centímetros, o pulgadas.
- pc: picas (12 puntos)

Unidades relativas

El valor final resultante (computado) está en función de un valor previo.

Em, es básicamente el tamaño de una letra «M» (bien podría ser cualquier otra letra) del elemento al cual se esté aplicando esta medida. Es decir, si en elemento tiene aplicado un tamaño de fuente de 16 pixeles, entonces 1 em será igual a 16px (los navegadores de manera predeterminada definen un font-size de 16px al elemento HTML, por lo tanto, por defecto 1em es igual a 16px).

La unidad **em** es escalable y siempre depende de su elemento padre. Por ejemplo, si el elemento body tiene un tamaño de fuente de 16px y un elemento hijo tiene una fuente con tamaño 1.3em, este texto se mostrará de un tamaño un 30% más grande que el del body (20.8px), mientras que si dentro de ese elemento tenemos otro hijo con un font-size de 1.3 em, el tamaño de fuente de este objeto sería un 30% más grande que el tamaño de su padre (27.04px).

Body = 1em (16px)

Hijo = 1.3em (16px x 1.3 = 20.8px)

Nieto = 1.3em (20.8px x 1.3 = 27.04px)

Porcentaje % La unidad de medida porcentual es la que se usa por defecto en los elementos HTML en donde de manera predeterminada cada elemento de bloque usa un ancho del 100%, es por eso que cuando achicamos la ventana del navegador con una página que no tenga estilos, la página se adapta, ya que siempre usará el ancho total visible. Pero nosotros podemos utilizar los porcentajes de una manera más avanzada tratando de generar layouts más complejos.

Supongamos, por ejemplo, que tienes un div que contiene todos los elementos de la página y, según el diseño, este elemento debiera medir 1200 píxeles. En lugar de caer en la tentación de simplemente usar esa medida en píxeles, te recomendaría usar una medida en porcentajes, en donde el máximo ancho del elemento sean esos 1200px:
CSS

```
.container {  
    margin: 0 auto;  
    width: 90%;  
    max-width: 1200px;  
}
```

Con estas 3 propiedades de CSS conseguimos que

- a) El elemento se centre en la página,
- b) tenga un ancho del 90% de la ventana y
- c) su ancho nunca sea superior a 1200 píxeles.

Es decir, hemos conseguido que el elemento con la clase container sea responsive sin la necesidad siquiera de escribir un media query.

El uso de los porcentajes también lo podemos llevar a elementos interiores del layout, en donde, por ejemplo, podemos asignar a la columna principal de contenido y a la barra lateral unas medidas de ancho del 70% y el 30% respectivamente, haciendo que sean completamente adaptables al tamaño de su elemento contenedor.

Rem:

La unidad de medida rem es muy similar a em, con la única diferencia de que no es escalable, esto quiere decir que no depende del elemento padre, sino del elemento raíz del documento, el elemento HTML. Rem significa «Root Em», o sea, es un em basado en la raíz.

Esto significa que si el elemento HTML tiene un tamaño de fuente de 16px (como es por defecto), entonces 1rem, sería igual a 16px, y si queremos aplicar un tamaño

basado en rem a cualquier elemento de la página, no importará cual sea el tamaño de fuente que tenga asociado ese elemento, ya que 1 rem siempre será igual a 16 pixeles a no ser que se modifique el elemento raíz.

Usar rem nos permite cierta estructura para poder definir ciertas partes del layout, pero al mismo tiempo nos entrega cierta escalabilidad para respetar las configuraciones de cada usuario.

Esta unidad de medida es recomendable para aplicar a elementos del layout que requieran medidas fijas y eventualmente también para textos que deseemos que tengan un tamaño de fuente que no dependa de su elemento padre.

Si quisiéramos refinar el ejemplo anterior para no usar pixeles deberíamos usar algo como lo siguiente:

CSS

```
.container {  
    margin: 0 auto;  
    width: 90%;  
    max-width: 75rem;  
}
```

Para poder convertir una medida de pixeles a rem solo tienes que multiplicar el tamaño que quieres obtener por el número 0.0625, eso te dará el tamaño que debes usar en rem. Así es como se define que 75rem es igual a 1200px:

$$75\text{rem} = 1200\text{px} \times 0.0625$$

Para evitarnos este cálculo se puede definir el font-size del elemento html con un tamaño de 62.5%, de esta forma, conseguimos que 1 rem sea equivalente a 10px, haciendo más fácil el cálculo.

- **ex, ch**: Son respectivamente la altura de la x minúscula, y el ancho del número 0. Aunque no son tan soportadas por los navegadores como los rem.
- **vw, vh**: Estas son respectivamente $1/100$ del ancho de la ventana, y $1/100$ de la altura de la ventana. Tampoco son tan soportadas como los rem.

Posicionamiento

A grandes rasgos, si tenemos varios elementos en línea (uno detrás de otro) aparecerán colocados de izquierda hacia derecha, mientras que si son elementos en bloque se verán colocados desde arriba hacia abajo. Estos elementos se pueden ir combinando y anidando (incluyendo unos dentro de otros), construyendo esquemas más complejos.

Hasta ahora, hemos estado trabajando sin saberlo en lo que se denomina posicionamiento estático (static), donde todos los elementos aparecen con un orden natural según donde estén colocados en el HTML. Este es el modo por defecto en que un navegador renderiza una página.

Sin embargo, existen otros modos alternativos de posicionamiento, que podemos cambiar mediante la propiedad position, que nos pueden interesar para modificar la posición en donde aparecen los diferentes elementos y su contenido.

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

Valor	Significado
static	Posicionamiento estático. Utiliza el orden natural de los elementos HTML.
relative	Posicionamiento relativo. Los elementos se mueven ligeramente en base a su posición estática.
absolute	Posicionamiento absoluto. Los elementos se colocan en base al contenedor padre.
fixed	Posicionamiento fijo. Idem al absoluto, pero aunque hagamos scroll no se mueve.

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

Propiedad	Valor	Significado
top:	auto TAMAÑO	Empuja el elemento una distancia desde la parte superior hacia el inferior.
bottom:	auto TAMAÑO	Empuja el elemento una distancia desde la parte inferior hacia la superior.
left:	auto TAMAÑO	Empuja el elemento una distancia desde la parte izquierda hacia la derecha.
right:	auto TAMAÑO	Empuja el elemento una distancia desde la parte derecha hacia la izquierda.
z-index:	auto nivel	Ordena en el eje de profundidad, superponiendo u ocultando.

Antes de pasar a explicar los tipos de posicionamiento, debemos tener claras las propiedades top, bottom, left y right, que 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.

Pero pasemos a ver cada tipo de posicionamiento por separado y su comportamiento:

Posicionamiento relativo

Si utilizamos la palabra clave relative activaremos el modo de posicionamiento relativo, que es el más sencillo de todos. 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 variaremos ligeramente la posición del elemento.

Ejemplo: Si establecemos left:40px, el elemento se colocará 40 píxeles a la derecha desde la izquierda donde estaba colocado en principio, mientras que si especificamos right:40px, el elemento se colocará 40 píxeles a la izquierda desde la derecha donde estaba colocado en principio.

Posicionamiento absoluto

Si utilizamos la palabra clave absolute estamos indicando que el elemento pasará a utilizar posicionamiento absoluto, que no es más que utilizar el documento completo como referencia. Esto no es exactamente el funcionamiento de este modo de posicionamiento, pero nos servirá como primer punto de partida para entenderlo.

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.

Como mencionaba anteriormente, aunque este es el funcionamiento del posicionamiento absoluto, hay algunos detalles más complejos en su modo de trabajar. Realmente, este tipo de posicionamiento coloca los elementos utilizando como punto de origen el primer contenedor con posicionamiento diferente a estático.

Por ejemplo, si el contenedor padre tiene posicionamiento estático, pasamos a mirar el posicionamiento del padre del contenedor padre, y así sucesivamente hasta encontrar un contenedor con posicionamiento no estático o llegar a la etiqueta <body>, en el caso que se comportaría como el ejemplo anterior.

Posicionamiento fijo

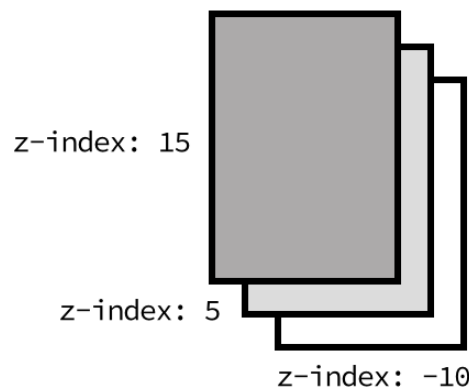
Por último, el posicionamiento fijo es hermano del posicionamiento absoluto. Funciona exactamente igual, salvo que hace que el elemento se muestre en una posición fija dependiendo de la región visual del navegador. Es decir, aunque el usuario haga scroll y se desplace hacia abajo en la página web, el 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.

Profundidad (niveles)

Es interesante conocer también la existencia de la propiedad `z-index`, que establece el nivel de profundidad en el que está un elemento sobre los demás. De esta forma, podemos hacer que un elemento se coloque encima o debajo de otro.

Su funcionamiento es muy sencillo, sólo hay que indicar un número que representará el nivel de profundidad del elemento. Los elementos un número más alto estarán por encima de otros con un número más bajo, que permanecerán ocultos detrás de los primeros.



Nota: Los niveles `z-index`, así como las propiedades `top`, `left`, `bottom` y `right` no funcionan con elementos que estén utilizando posicionamiento estático. Deben tener un tipo de posicionamiento diferente a estático.

Float

Ver documentación: <https://developer.mozilla.org/es/docs/Web/CSS/float>