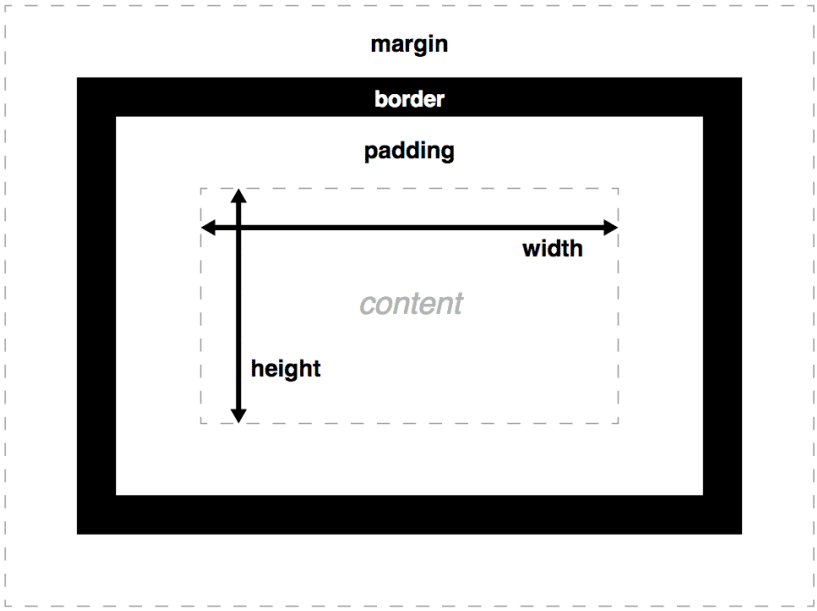
Docs_

Posicionamiento

El modelo de cajas

Los elementos de un documento HTML son considerados como una caja rectangular invisible al usuario y en la que se pueden aplicar ciertas propiedades para ubicar los elementos respecto a otros.



El modelo de caja (Box Model) en HTML y CSS es un concepto fundamental en el diseño web, que describe cómo se diseñan y se estructuran los elementos en una página web. Este modelo se aplica a todos los elementos de la página y es crucial para entender cómo se maneja el diseño y el espacio entre los distintos componentes de una página web. Cada elemento se considera como una caja que puede tener margen, borde, relleno (padding) y el contenido en sí. Aquí te explico cada uno de estos componentes:

1. **Contenido (Content)**: Es la parte central del modelo de caja, donde se muestra el contenido real del elemento, como texto, imágenes o videos.

2. **Relleno (Padding)**: Es el espacio que rodea el contenido dentro del elemento. El relleno aumenta el área del elemento pero no su transparencia; es decir, el color de fondo del elemento se extiende hasta el borde del relleno. El relleno se sitúa entre el contenido y el borde del elemento.

3. **Borde (Border)**: Es una línea que envuelve el relleno (y, por ende, el contenido). Puede tener un estilo (sólido, punteado, etc.), un color y un grosor definidos. El borde es parte del tamaño total del elemento, aumentando el espacio que ocupa en la pantalla.

4. **Margen (Margin)**: Es el espacio que separa el elemento de otros elementos en la página. A diferencia del relleno, el margen no tiene color; es un espacio transparente que se utiliza para crear distancia entre los elementos. Los márgenes pueden ser negativos, lo que permite que los elementos se superpongan entre sí.

La importancia del modelo de caja radica en su utilidad para controlar el diseño y el espaciado entre los elementos en una página web. Permite a los desarrolladores web y diseñadores crear layouts precisos y responsivos.

CSS utiliza propiedades específicas para manipular cada parte del modelo de caja:

- **width** y **height** controlan el tamaño del contenido.
- **padding** controla el tamaño del relleno.
- **border** especifica el estilo, el ancho y el color del borde.
- **margin** controla el espacio exterior al borde.

Es importante mencionar que el tamaño total de un elemento, según el modelo de caja estándar, es la suma de su ancho/alto de contenido, relleno, borde y margen. Sin embargo, CSS introdujo la propiedad **box-sizing** que permite cambiar cómo se calcula el tamaño de las cajas:

- **content-box** (valor por defecto): El tamaño especificado aplica solo al contenido, y el relleno y borde se añaden al tamaño final.
- **border-box**: El tamaño especificado incluye el contenido, el relleno y el borde, pero no el margen. Esto hace más fácil el diseño, especialmente en layouts responsivos, ya que el tamaño final de la caja es más predecible.

Overflow

La propiedad **overflow** en CSS controla qué sucede con el contenido que desborda de su contenedor. Este desbordamiento ocurre cuando el contenido de un elemento es demasiado grande para caber en el área asignada para el elemento. La propiedad **overflow** puede aplicarse a cualquier elemento bloque o reemplazado que tenga un tamaño especificado (ya sea a través de **width**, **height** o ambos) o **inline-blocks**, **table-cells**, y otros contextos específicos.

La propiedad **overflow** tiene varias opciones:

- **visible**: Es el valor predeterminado. El contenido desbordado no se recorta y puede ser renderizado fuera del contenedor.
- **hidden**: El contenido desbordado se recorta, y el resto del contenido será invisible.
- **scroll**: Se añaden barras de desplazamiento al contenedor, tanto horizontal como verticalmente, independientemente de si el contenido desborda realmente. Esto permite al usuario desplazarse para ver el contenido oculto.
- **auto**: El navegador decide si añadir barras de desplazamiento solo cuando el contenido desborda el contenedor.

Además, CSS3 introdujo **overflow-x** y **overflow-y**, permitiendo controlar el desbordamiento en el eje horizontal y vertical independientemente, con los mismos valores que **overflow**.

Ejemplos de uso:

```
/* Oculta el contenido que desborda el contenedor */
.elemento {
  overflow: hidden;
}

/* Añade barras de desplazamiento si es necesario */
.contenedor {
  overflow: auto;
}

/* Control independiente para desbordamiento horizontal y vertical */
.caja {
  overflow-x: scroll; /* Añade una barra de desplazamiento horizontal */
  overflow-y: hidden; /* Oculta el desbordamiento vertical */
}
```

La propiedad **overflow** es especialmente útil en el diseño web responsivo y en la creación de componentes de interfaz de usuario como menús desplegables, cajas de diálogo y paneles deslizantes, donde el manejo del espacio y el contenido desbordado son consideraciones críticas. Al ajustar el desbordamiento, los desarrolladores pueden asegurar que el contenido sea accesible y presentado de manera adecuada, independientemente del tamaño del dispositivo o de la ventana del navegador.

Z-index

La propiedad **z-index** en CSS controla el orden de apilamiento de los elementos que se superponen en el eje Z; es decir, determina qué elementos aparecen frente a otros en la página. Se utiliza principalmente en elementos con posicionamiento distinto de **static** (por ejemplo, **relative**, **absolute**, **fixed**, o **sticky**) porque estos tipos de posicionamiento sacan al elemento del flujo normal de la página, permitiendo que se apilen.

El valor de **z-index** puede ser un número entero positivo o negativo, **auto**, o **0**. Los elementos con un valor de **z-index** más alto se sitúan encima de los elementos con un valor más bajo en la página web. Si dos elementos tienen el mismo valor de **z-index**, su orden relativo seguirá el flujo del documento (el que aparece último en el código HTML se renderiza encima del otro). Aquí algunos puntos clave sobre **z-index**:

1. **Valores Positivos**: Coloca el elemento por encima de aquellos con valores menores o los elementos sin un **z-index** definido.
2. **Valores Negativos**: Coloca el elemento detrás de otros elementos, incluso detrás de aquellos que no tienen un **z-index** definido, siempre que estén en el mismo contexto de apilamiento.
3. **auto**: El valor predeterminado. El elemento se apila en el orden natural definido por el HTML, siguiendo las reglas normales del flujo del documento.
4. **0**: Coloca el elemento en el mismo nivel de apilamiento que los elementos sin un **z-index** definido, pero aún permite que se apile por encima o por debajo de otros elementos mediante el ajuste de **z-index** en otros elementos.

Contexto de apilamiento: Es importante notar que **z-index** solo funciona dentro del mismo contexto de apilamiento. Un contexto de apilamiento es, en esencia, un contenedor en el cual el orden de apilamiento de sus hijos está contenido y no afecta a elementos fuera de este. Un nuevo contexto de apilamiento puede ser creado por propiedades como **opacity** menor a 1, **transform**, **filter** diferentes de **none**, y otros, además del mencionado posicionamiento no **static** con **z-index** diferente de **auto**.

Ejemplo de uso:

```
.elemento-detras {
  position: absolute;
  z-index: -1;
}

.elemento-delante {
  position: absolute;
  z-index: 2;
}
```

En este ejemplo, **.elemento-delante** siempre aparecerá frente a **.elemento-detras** debido a su mayor valor de **z-index**. La propiedad **z-index** es crucial para el diseño web complejo, especialmente en interfaces ricas en elementos superpuestos como modales, menús desplegables, y elementos de navegación fija.

```
<!DOCTYPE html>
<html>

<head>
  <style>
    img {
      position: absolute;
      left: 0px;
      top: 0px;
      z-index: -1;
    }

  </style>
</head>

<body>

  <h1>This is a heading</h1>
  <img src="https://api.lorem.space/image/face?w=150&h=150">
  <p>Because the image has a z-index of -1, it will be placed behind the text.</p>

</body>

</html>
```

Posicionamiento de cajas

Podemos diferenciar dos tipos de elementos HTML, de los que podemos usar dentro del cuerpo de las páginas para agregar contenido, en función de cómo se representan en pantalla. En bloque (*block*) o en línea (*inline*).

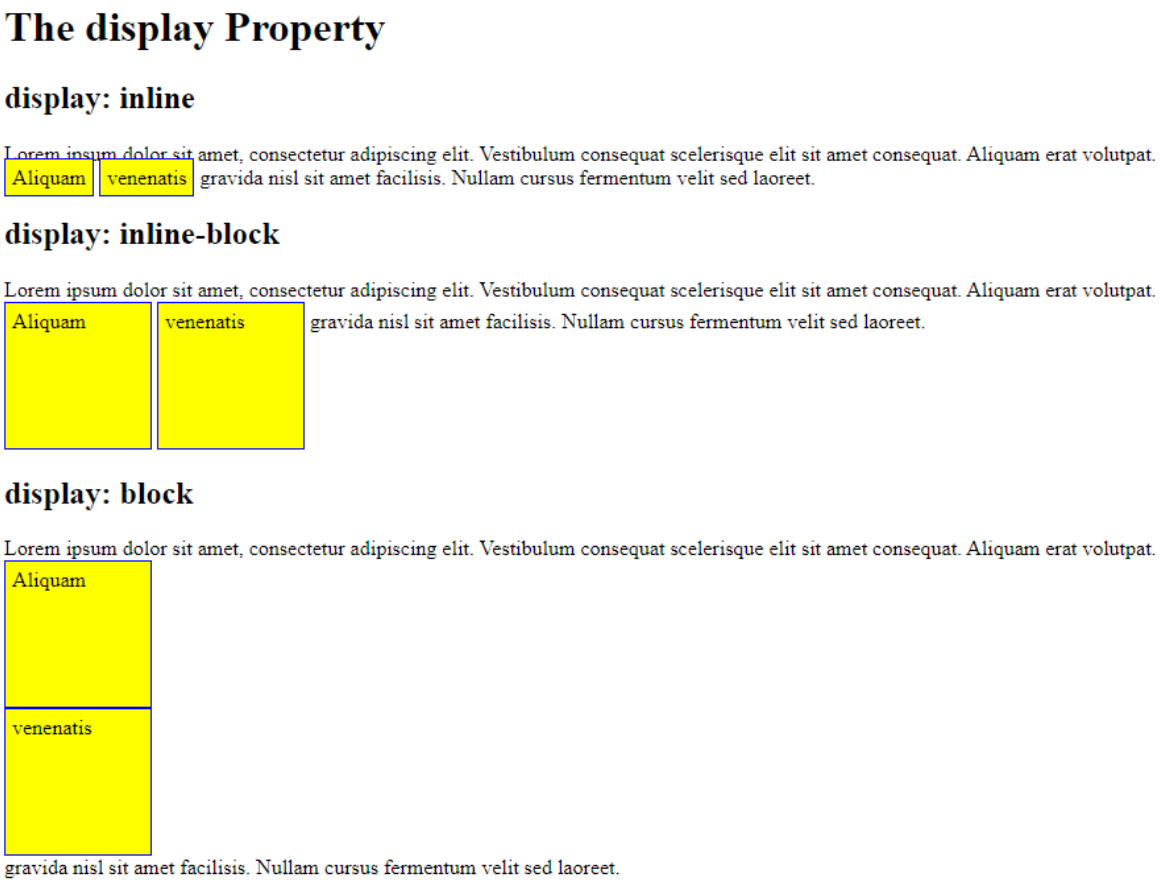
Block

Los bloques ocupan todo el ancho de pantalla y siempre comienzan en una línea nueva. Estos elementos pueden contener a otros elementos de bloque o en línea y su altura varía en función de su contenido. Ejemplos son *div, form, h1-h6, ol, p, table o li*.

Podemos especificar su tamaño en pantalla mediante CSS como hemos visto con *width* y *height*.

Inline

Ocupan el espacio de su contenido, no tienen por qué comenzar en una nueva línea, sólo pueden contener a otros elementos *inline* y sus dimensiones no se pueden especificar mediante CSS. Ejemplos: *a, br, img, input, label, select, span, strong…*



```
<!DOCTYPE html>
<html>
<head>
<style>
span.a {
  display: inline; /* the default for span */
  width: 100px;
  height: 100px;
  padding: 5px;
  border: 1px solid blue;
  background-color: yellow;
}
span.b {
  display: inline-block;
  width: 100px;
  height: 100px;
  padding: 5px;
  border: 1px solid blue;
  background-color: yellow;
}
span.c {
  display: block;
  width: 100px;
  height: 100px;
  padding: 5px;
  border: 1px solid blue;
  background-color: yellow;
}
</style>
</head>
<body>
<h1>The display Property</h1>
<h2>display: inline</h2>
<div>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. <span class="a">Aliquam</span> <span class="a">venenatis</span> gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet. </div>
<h2>display: inline-block</h2>
<div>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. <span class="b">Aliquam</span> <span class="b">venenatis</span> gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet. </div>
<h2>display: block</h2>
<div>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. <span class="c">Aliquam</span> <span class="c">venenatis</span> gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet. </div>
</body>
</html>
```

Posicionamiento predefinido (estático)

De forma predefinida los elementos de bloque aparecen unos debajo de otros según el flujo en que los pongamos en el documento. Si asignamos una altura o anchura a uno, los que pongamos en su interior estarán limitados a ese tamaño.

```
<!DOCTYPE html>
<html>
<head>
<style>
div.static {
  position: static;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>
<h2>position: static;</h2>

<p>An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:</p>

<div class="static">
This div element has position: static;
</div>

</body>
</html>
```

Posicionamiento relativo

Podemos especificar desplazamiento a una caja con respecto a su posición original en el flujo de ejecución (predefinida).

Utilizamos la propiedad:

```
position: relative;
```

Y después para aplicar el desplazamiento, las propiedades top, left, right y bottom.

Por ejemplo:

```
left: 15px;
```

Con la regla anterior desplazamos el elemento 15px respecto a su borde izquierdo, de modo que observaremos un desplazamiento hacia la derecha.

Este tipo de posicionamiento no modifica el posicionamiento de las demás cajas a su alrededor, pero sí puede provocar solapamientos.

```
<!DOCTYPE html>
<html>
<head>
<style>
div.relative {
  position: relative;
  left: 30px;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>

<h2>position: relative;</h2>

<p>An element with position: relative; is positioned relative to its normal position:</p>

<div class="relative">
This div element has position: relative;
</div>

</body>
</html>
```

Posicionamiento absoluto

Con este modo de posicionamiento tomamos como referencia al primer elemento padre o contenedor que no esté estático. Si no existe, la referencia sería la ventana del navegador.

```
position: absolute;
```

Las propiedades top, left, right y bottom aplican desplazamiento desde el borde en cuestión del elemento hasta los bordes correspondientes en el elemento padre. En este caso, el desplazamiento influye en el resto de elementos de la página, tratando los demás elementos de rellenar el hueco abandonado.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    div.relative {
      position: relative;
      width: 400px;
      height: 200px;
      border: 3px solid #73AD21;
    }
    div.absolute {
      position: absolute;
      top: 80px;
      right: 0;
      width: 200px;
      height: 100px;
      border: 3px solid #73AD21;
    }
  </style>
</head>
<body>
  <h2>position: absolute;</h2>
  <p>An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of
    positioned relative to the viewport, like fixed):</p>

  <div class="relative">This div element 1 has position: relative;
    <div class="absolute">This div element has position: absolute;</div>
  </div>
  <div class="absolute">This div element 2 has position: absolute;</div>
</body>
</html>
```

Fixed

El posicionamiento fijo lo empleamos mediante el valor *fixed* y es como el absoluto salvo que en este caso siempre se toma como referencia la ventana del navegador. En este caso aunque hagamos scroll, el elemento permanecerá en la misma posición.

```
<!DOCTYPE html>
<html>
<head>
<style>
div.fixed {
  position: fixed;
  bottom: 0;
  right: 0;
  width: 300px;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>

<h2>position: fixed;</h2>

<p>An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled:</p>

<div class="fixed">
This div element has position: fixed;
</div>

</body>
</html>
```

Sticky

en este caso la posición del elemento pasa de ser relativa a fija (fixed) cuando el usuario hace scroll de forma que el elemento dejara de verse. Hay navegadores en que no funcionará correctamente.

```
<!DOCTYPE html>
<html>
<head>
<style>
div.sticky {
  position: -webkit-sticky;
  position: sticky;
  top: 0;
  padding: 5px;
  background-color: #cae8ca;
  border: 2px solid #4CAF50;
}
</style>
</head>
<body>

<p>Try to <b>scroll</b> inside this frame to understand how sticky positioning works.</p>

<div class="sticky">I am sticky!</div>

<div style="padding-bottom:2000px">
  <p>In this example, the sticky element sticks to the top of the page (top: 0), when you reach its scroll position.</p>
  <p>Scroll back up to remove the stickyness.</p>
  <p>Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.</p>
  <p>Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.</p>
</div>

</body>
</html>
```

Float

Es una propiedad que puede tomar los valores *right*, *left*, *none* (predefinido) e *inherit*. Lo que provoca es que ele elemento flote hacia el lado indicado de su contenedor. Cuando un elemento flota, deja el hueco que ocupaba y otro puede ocupar su ubicación anterior.

```
float: right;
```

Si otros elementos flotan hacia una misma dirección en el mismo contenedor, entonces sí respetan el orden y posiciones y no se solaparían.

Si flotamos varios elementos hacia la derecha, aparecerán colocados en orden inverso.

Otro ejemplo de esto es la colocación de texto alrededor de una imagen en un artículo de un blog. Este es el uso original y más indicado para el que se creó esta propiedad.

A pesar de que el uso para el que se creó *float* fue para colocar texto alrededor de otro objetos como imágenes, durante mucho tiempo se ha empleado para maquetar las estructuras de las páginas web, creando zonas dentro de las mismas como las barras laterales o elementos dispuestos en filas y columnas como una galería de imágenes o un listado de productos.

```
<!DOCTYPE html>
<html>

<head>
  <style>
    img {
      float: right;
    }
  </style>
</head>

<body>
  <h2>Float Right</h2>
  <p>In this example, the image will float to the right in the paragraph, and the text in the paragraph will wrap
    around the image.</p>

  <p><img src="https://api.lorem.space/image/face?w=300&h=300" alt="face"
      style="width:170px;height:170px;margin-left:15px;">
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi
    lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue
eget,
    auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu,
    lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum
nisi,
    sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae
dui
```

```
    eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor
    porta. Cras ac leo purus. Mauris quis diam velit.</p>

</body>

</html>
```

Clear

Es otra propiedad que sirve para evitar que un elemento tenga a sus lados algún elemento flotante pasándolo a la siguiente línea o hueco en el flujo del documento.

Por tanto, el valor *left* provocará que el elemento en cuestión se desplace hacia abajo hasta encontrar un hueco en el que no exista un elemento flotante a su lado izquierdo.
Podemos asignarle los valores *left*, *right*, *both* o *inherit*.

both implica ambos lados.

Solución al problema del alto de la caja contenedora de flotantes

Cuando hacemos flotar elementos surge el problema de que, como los elementos flotantes salen del flujo normal del documento, los elementos que estén a continuación pasan a ocupar los huecos disponibles.

Esto tiene el problema de que la caja contenedora de los elementos flotantes, si no contiene a otros elementos, no llega hasta el final de los elementos flotantes.

En el siguiente ejemplo podemos ver como la caja negra, que contiene a las grises, no llega al final:

Soluciones:

1. Indicar *overflow: auto* a la caja contenedora. (no siempre funciona).
2. Agregar un div limpiador tras los flotantes.
3. Emplear la pseudoclase after con un clear.

Dos recursos interesantes (English):
HTML: Give Parent Div 100% Height Of Child Floated Elements
CSS Tricks: Float

```
<!DOCTYPE html>
<html>

<head>
  <style>
    .div1 {
      float: left;
      padding: 10px;
      border: 3px solid #73AD21;
    }

    .div2 {
      padding: 10px;
      border: 3px solid red;
    }

    .div3 {
      float: left;
      padding: 10px;
      border: 3px solid #73AD21;
    }

    .div4 {
      padding: 10px;
      border: 3px solid red;
      clear: left;
    }
  </style>
</head>

<body>

  <h2>Without clear</h2>
  <div class="div1">div1</div>
  <div class="div2">div2 - Notice that div2 is after div1 in the HTML code. However, since div1 floats to the left,
    the text in div2 flows around div1.</div>
  <br><br>

  <h2>With clear</h2>
  <div class="div3">div3</div>
  <div class="div4">div4 - Here, clear: left; moves div4 down below the floating div3. The value "left" clears
    elements floated to the left. You can also clear "right" and "both".</div>

</body>

</html>
```

Display

Es interesante considerar que podemos modificar el comportamiento de un elemento HTML sobrescribiendo la propiedad *display* e intercambiando su comportamiento de bloque o en línea a conveniencia.

Los valores más comunes de asignar al valor de esta propiedad son por tanto *block, inline, none, hidden e inline-block*.

Tanto *none* como *hidden* hacen desaparecer al elemento, la diferencia es que con *hidden* simplemente se oculta, aunque este sigue ocupando su lugar. Con *none*, desaparece totalmente.

Un uso común es poner los elementos de una lista en línea para que muestren en horizontal y aplicarlo en menús.

li {display: inline;}

o poner un *span* en bloque para que ocupe todo el ancho o un ancho específico.

span {display: block;}

Es importante recordar que a los elementos *inline* no les podemos asignar un ancho o un alto.

Inline-block

Inline-block nace tras la necesidad de un método más eficaz que *float* para colocar elementos en disposición horizontal.

Un uso muy común es para generar menús de navegación, en los que, a partir de una lista desordenada, ubicamos los elementos *li* en hilera asignándoles también un ancho específico.

Una vez hayamos practicado con *inline* y *block*, podemos ver esta posibilidad. Respecto a *inline*:

- Podemos asignar *width* y *height* al elemento.
- Nos respeta *padding* y *margin*.

Respecto a block

- No genera una nueva línea al final.

Te propongo un ejercicio. La siguiente tabla periódica la ha hecho uno de mis alumnos, Antonio Fernández. Con lo que has aprendido hasta ahora lo puedes hacer tú también:

Tabla Periódica de los Elementos

1 1,00794 **H** HIDRÓGENO																	2 4.0026 **He** HELIO
3 6.941 **Li** LITIO	4 9.0122 **Be** BERILIO											5 10.811 **B** BORO	6 12.011 **C** CARBONO	7 14.007 **N** NITRÓGENO	8 15.999 **O** OXÍGENO	9 18.998 **F** FLÚOR	10 20.179 **Ne** NEÓN
11 22.989 **Na** SODIO	12 24.31 **Mg** MAGNESIO											13 26.981 **Al** ALUMINIO	14 28.085 **Si** SILICIO	15 30.973 **P** FÓSFORO	16 32.065 **S** AZUFRE	17 35.453 **Ci** CLORO	18 39.948 **Ar** ARGÓN
19 39.098 **K** POTASIO	20 40.078 **Ca** CALCIO	21 44.955 **Sc** ESCANDIO	22 47.867 **Ti** TITANIO	23 50.941 **V** VANADIO	24 51.996 **Cr** CROMO	25 54.938 **Mn** MANGANESO	26 55.847 **Fe** HIERRO	27 58.933 **Co** COBALTO	28 58.71 **Ni** NÍQUEL	29 63.536 **Cu** COBRE	30 65.38 **Zn** ZINC	31 69.723 **Ga** GALIO	32 72.64 **Ge** GERMANIO	33 74.921 **As** ARSÉNICO	34 78.96 **Se** SELENIO	35 79.904 **Br** BROMO	36 83.80 **Kr** KRIPTÓN
37 85.4678 **Rb** RUBIDIO	38 87.62 **Sr** ESTRONCIO	39 88.905 **Y** ITRIO	40 91.224 **Zr** CIRCONIO	41 92.906 **Nb** NIOBIO	42 95.94 **Mo** MOLIBDENIO	43 98,9063 **Tc** TECNECIO	44 101.07 **Ru** RUTENIO	45 102.90 **Rh** RODIO	46 106.42 **Pd** PALADIO	47 107.86 **Ag** PLATA	48 112.411 **Cd** CADMIO	49 114.818 **In** INDIO	50 118.710 **Sn** ESTAÑO	51 121.760 **Sb** ANTIMONIO	52 127.6 **Te** TELURIO	53 126.90 **I** YODO	54 131.293 **Xe** XENÓN
55 132.90 **Cs** CESIO	56 137.32 **Ba** BARIO	57 - 71 **La-Lu** LANTÁNIDOS	72 178.49 **Hf** HAFNIO	73 180.94 **Ta** TÁNTALO	74 183.84 **W** WOLFRAMIO	75 186.20 **Re** RENIO	76 190.23 **Os** OSMIO	77 192.21 **Ir** IRIDIO	78 195.08 **Pt** PLATINO	79 196.9 **Au** ORO	80 200.59 **Hg** MERCURIO	81 204.38 **Tl** TALIO	82 207.2 **Pb** PLOMO	83 208.98 **Bi** BISMUTO	84 209,98 **Po** POLONIO	85 (210) **At** ASTATO	86 (222) **Rn** RADÓN
87 (223) **Fr** FRANCIO	88 (226) **Ra** RADIO	89 - 103 **Ac-Lr** ACTÍNIDOS	104 (261) **Rf** RUTHERFORDIO	105 (262) **Db** DUBNIO	106 (269) **Sg** SEABORGIO	107 (264) **Bh** BOHRIO	108 (269) **Hs** HASIO	109 (268) **Mt** MEITNERIO	110 (281) **Ds** DARMSTATIO	111 (281) **Rg** ROENTGENIO	112 (285) **Cn** COPERNICIO	113 (286) **Nh** NIHONIO	114 (287) **Fl** FLEROVIO	115 (288) **Mc** MOSCOVIO	116 (291) **Lv** LIVERMORIO	117 (294) **Ts** TENESO	118 (294) **Og** OGANESÓN

5 10.811 **B**	57 138,90 **La** LANTANO	58 140,11 **Ce** CERIO	59 140,90 **Pr** PRASEODIMIO	60 144,24 **Nd** NEODIMIO	61 145,00 **Pm** PROMETIO	62 150,35 **Sm** SAMARIO	63 151,96 **Eu** EUROPIO	64 157,25 **Gd** GADOLINIO	65 158,92 **Tb** TERBIO	66 162,50 **Dy** DISPROSIO	67 164,93 **Ho** HOLMIO	68 167,25 **Er** ERBIO	69 168,93 **Tm** TULIO	70 173,04 **Yb** ITERBIO	71 174,96 **Lu** LUTECIO
	89 (227) **Ac** ACTINIO	90 (232) **Th** TORIO	91 (231) **Pa** PRASEODIMIO	92 (238) **U** URANIO	93 (237) **Np** NEPTUNIO	94 (244) **Pu** PLUTONIO	95 (243) **Am** AMERICIO	96 (247) **Cm** CURIO	97 (247) **Bk** BERKELIO	98 (251) **Cf** CALIFORNIO	99 (252) **Es** EINSTENIO	100 (257) **Fm** FERMIO	101 (258) **Md** MENDELEVIO	102 (259) **No** NOBELIO	103 (262) **Lr** LAWRENCIO

Box-sizing

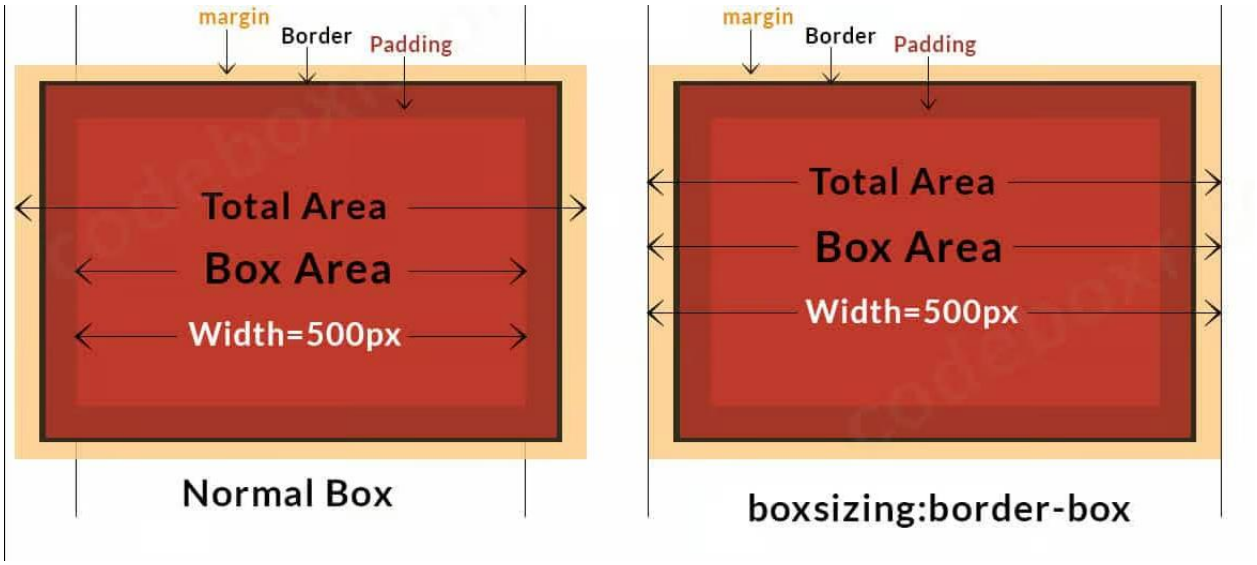
Esta propiedad tiene dos posibles valores distintos, aparte de *initial* e *inherit*: *content-box* y *border-box.*

content-box es el valor predefinido, en el que cuando aplicamos medidas a la caja con *width* o *height* el grosor del *padding*, *border* y *margin* se sumará al indicado.

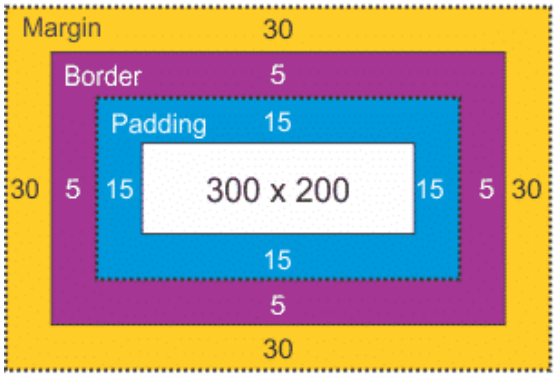
Con *border-box* conseguimos que el *padding* y el borde estén ya incluidos en esta medida, siendo así más fácil de cuadrar el resultado final cuando trabajamos con varias cajas.

El valor *border-box* de la propiedad *boxsizing* se emplea para facilitarnos la tarea de calcular y ajustar las dimensiones de los elementos cuando pretendemos que quepan varios colocados horizontalmente.

Podemos verlo gráficamente aquí:

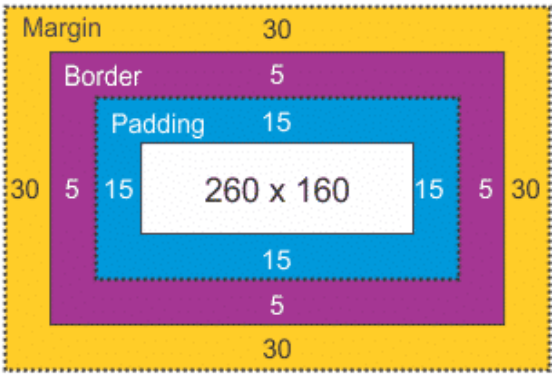


Box Model is `content-box`




```
div{
      width: 300px;
      height: 200px;
      padding: 15px;
      border: 5px solid grey;
      margin: 30px;
      -moz-box-sizing: content-box;
      -webkit-box-sizing: content-box;
      box-sizing: content-box;
    }
```

Box Model is `border-box`




```
div{
      width: 300px;
      height: 200px;
      padding: 15px;
      border: 5px solid grey;
      margin: 30px;
      -moz-box-sizing: border-box;
      -webkit-box-sizing: border-box;
      box-sizing: border-box;
    }
```