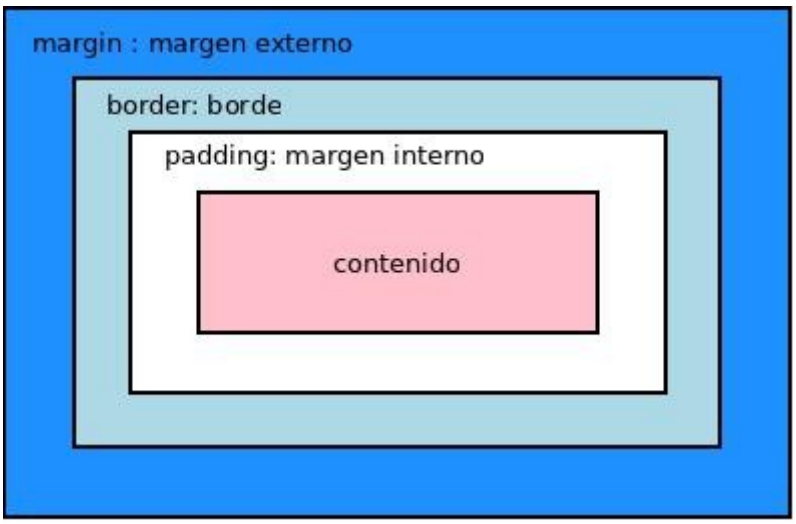
Unidad 4

El modelo de cajas

"El aprendizaje no es un deporte
para espectadores"
D. Blocher

El modelo de cajas se trata, posiblemente, del concepto más importante relacionado con las hojas de estilo CSS. Como ya hemos dicho antes, todo en HTML está formado por cajas rectangulares que contienen a los diferentes elementos. Mediante las reglas que veremos a continuación podremos cambiar el aspecto y comportamiento de las mismas.

HTML y CSS definen una caja como una zona rectangular constituida por los siguientes elementos:



Tenemos, además, control total y de forma individual de cada una de las dimensiones de las tres capas externas. El contenido suele ser dinámico (en una o en sus dos dimensiones) o especificarse con dos valores: ancho y alto. Como unidades de medida tenemos todas las que ya conocemos

La principal confusión que suelen tener los "primerizos" es la forma de calcular las dimensiones totales de la caja. Supongamos una caja cuyo contenido tiene un ancho de 200 píxeles, el margen interior es de 20 píxeles, el borde de 5 y el margen externo de 25 píxeles ¿Cuál es el ancho total de la caja?

Haz las cuentas: son 300 píxeles ¿verdad?. Es mucho mejor verlo antes de echarle un vistazo a la sintaxis que es aún más engañosa:

```
.ejemplo { width: 200px;
    height: 100px;
    border: 5px solid blue;
    padding: 20px;
    margin: 25px;
    background-color: yellow; }
```

Dos nuevas etiquetas: div y span

div y span son dos nuevas etiquetas HTML cuyo objetivo es no hacer nada por defecto. Puedes hacer la prueba de colocarlas en un documento HTML y podrás comprobarlo por ti mismo. El objeto de ambas es dar estructura a nuestro documento. Se usan conjuntamente con las hojas de estilo y, particularmente, con los selectores de clase e ID.

div es una etiqueta de bloque ideada para estructurar de forma lógica nuestra página. Sería similar a un párrafo pero, mucho más flexible y, además, por defecto no conlleva la aplicación de ningún estilo.

La etiqueta span es muy similar pero se trata de una etiqueta de línea y se usa para agrupar elementos a los que aplicar estilos diferenciados.

Veamos el body del siguiente HTML:

```
<body>
   <div class="verde">
      <h1><a id="INICIO">Titulo de la <strong>pagina</strong></a></h1>
```

```
      <p>Si meliora dies... bra, bra...</p>
      <hr />
      <p>Otro párrafo... bra, bra, bra...</p>
  </div>
  <p>Este bloque ya está fuera del efecto del anterior bloque div y,
por tanto, su fondo será blanco.</p>
</body>
```

Y lo acompañamos de esta regla CSS:

```
.verde { background-color: green;}
```

Observamos que, efectivamente, mediante div podemos agrupar diferentes elementos de bloque y aplicar propiedades de forma conjunta a todos ellos. Podríamos conseguir el mismo efecto añadiendo class=”verde” como atributo a todos los elementos de bloque incluidos en el elemento div, pero sería más pesado, menos claro y podría dar lugar a más errores e inconsistencias.

Imaginemos ahora que queremos marcar de forma especial con otro estilo y otro tamaño, la primera letra del primer párrafo de un libro de cuentos que queremos publicar en la web. Usaremos para ello una fuente de Google Fonts, así que incluimos esto en el head de nuestro HTML:

```
<link href='http://fonts.googleapis.com/css?family=Rock+Salt'
rel='stylesheet' type='text/css'>
```

Luego, en nuestras hojas de estilo, definimos lo siguiente:

```
.primeraletra {font-family: 'Rock Salt', cursive; font-size: 500%;
font-weight: bolder; }
```

Y los párrafos de nuestra web irían así:

```
<p><span class="primeraletra">S</span>i meliora dies, ut vina</p>
<p><span class="primeraletra">V</span>Veteresne poetas, an quos et
praesens et postera respuat aetas?</p>
```

NOTA: *Más adelante veremos que existe otra forma más profesional de hacer esto mismo mediante los pseudo-elementos).*

Aunque muchos elementos de bloque de HTML son susceptibles de ser usados como cajas, la que por regla general usaremos como tal es la etiqueta **div**. Es la más versatil ya que no tiene estilos propios salvo los que definamos nosotros.

Propiedades de las cajas

Las dimensiones del contenido

Las dimensiones del contenido de la caja se delimitan con las propiedades width y height. Ambas puede tener por valor un porcentaje, una medida o la palabra auto que haría que su tamaño se ajustara lo mejor posible al contenido real de la misma. Los valores por defecto de estas dos propiedades son width: 100% y height: auto;

Cuando llevan un porcentaje, este se aplica sobre el tamaño de la caja dentro de la que, a su vez, se encuentran (su elemento padre o contenedor). Por ejemplo, si una caja tiene por ancho el 50% y no está dentro de ninguna otra, esto se aplica sobre el tamaño total de la superficie útil del navegador. Si dentro de esta metemos otra también con un ancho del 50%, tendrá un ancho de la mitad de la caja donde se encuentra, es decir, un 25% de la superficie útil del navegador. Cambiemos el ancho de la regla .ejemplo vista antes por 50% y pongamos esto en nuestro HTML:

```
<div class="ejemplo">Hola caja 1
        <p class="ejemplo">Hola caja 2</div>
</div>
```

IMPORTANTE: *La altura sólo puede valer un porcentaje si su padre o*

elemento contenedor tiene definida una altura con un valor concreto. En caso contrario se ignorará y tomará el valor auto.

Alternativamente tenemos las propiedades max-height, min-height, max-width y min-width para fijar las dimensiones máximas y mínimas. En este caso no podemos usar la palabra auto sino sólo porcentajes o medidas concretas.

Márgenes interior y exterior

Los márgenes interiores y exteriores se controlan con las propiedades padding y margin respectivamente y tenemos varias posibilidades. Podemos poner auto (para dejarlo al arbitrio de lo predefinido por el navegador), un valor exacto o un porcentaje que estaría referido al tamaño del elemento padre. Además, también podemos especificar un margen igual para cada una de las cuatro dimensiones de la caja o especificar que valor concreto queremos para ellas. Las propiedades para esto son margin-top, margin-right, margin-bottom y margin-left para el margen externo y padding-top, padding-right, padding-bottom y padding-left para el interno. Una última posibilidad, más compacta, es usar la propiedad margin con uno, dos, tres o cuatro valores:

```
        /* 20px para cada margen */
        margin: 20px;

        /* 20px para superior e inferior y 10px para derecho e
izquierdo */
        margin: 20px 10px;

        /* 20px para el superior 10px para los laterales y 30 para el
inferior */
        margin: 20px 10px 30px;

/* los cuatro en el sentido de las agujas del reloj: 20px superior,
10px derecho, 30px inferior y 10px izquierdo */
        margin: 20px 10px 30px 10px;
```

Un truco habitual para mostrar una caja centrada con respecto al elemento donde está contenida es hacer que los márgenes a izquierda y derecha sean iguales y calcular el tamaño exacto que deben tener. Esto es relativamente fácil cuando trabajamos únicamente con porcentajes, pero no siempre es posible. El navegador lo hará solo estableciendo los márgenes laterales a **auto**.

Existe una peculiaridad con respecto a los márgenes exteriores de las cajas. Así como los márgenes derecho e izquierdo de cajas contiguas se suman, los superior e inferior de cajas adyacentes se solapan. Imaginemos, por ejemplo, dos cajas con margen exterior de 20 píxeles en cada una de sus cuatro dimensiones. Si las situamos una junto a otra tendremos 40 píxeles de separación pero si ponemos una encima de la otra sólo estarán separadas por 20 píxeles.

Para la propiedad padding la sintaxis es similar. Además, el margen exterior permite valores negativos (¡experimenta con los resultados!) mientras que el interior traducirá un valor negativo por cero.

Y un último detalle: el body de un documento HTML es también una caja y puede tener predefinidos valores para el margin y el padding de forma que nos quede un pequeño borde sin usar en la superficie del navegador. Si no queremos que esto sea así podemos usar esta regla:

```
body {margin: 0px; padding: 0px; }
```

El borde de la caja

El borde de la caja se define mediante tres propiedades: color, grosor y estilo: border-color, border-width y border-style, aunque luego veremos que tenemos más posibilidades tanto para simplificar la definición como para particularizarla con mucho más detalle como hemos hecho con los márgenes.

La forma abreviada de especificar las características del borde de la caja es esta:

```
border: red double 5px;
```

Donde el primer parámetro es el color (de cualquiera de las formas que hemos aprendido a definirlo), el segundo el estilo (siendo válidos los valores solid, dashed, dotted, double, groove, ridge, inset, outset, hidden o none) y el tercero es el ancho del borde que puede ser un valor de longitud (nunca un porcentaje) o las palabras clave thin, medium o thick.

Este formato abreviado sólo es válido cuando las cuatro dimensiones del borde son iguales. Los parámetros pueden ir en cualquier orden pero deberían de ir los tres (aunque la mayoría de los navegadores nos admitirían uno o dos solamente). Los valores por defecto si prescindimos de alguno son black para el color, medium para el grosor y none para el estilo.

La diferencia entre none y hidden es que aunque en ninguno de los casos el borde es visible, en el segundo influye respecto al lugar donde se dibujaría un elemento adyacente mientras que el primero se considera con un ancho de 0px.

Podemos especificar cada una de estas tres propiedades por separado con los atributos border-color, border-style y border-width. Asimismo, podemos cambiar los valores de cada una de las cuatro dimensiones en cada uno de ellos usando atributos específicos (border-top-color, border-bottom-color, border-left-color y border-right-color, por ejemplo) o con una lista de dos, tres o cuatro valores sobre el atributo principal igual que hemos hecho con los márgenes:

```
/* 5px para superior e inferior y 50px para derecho e izquierdo */

border-width: 5px 50px;

/*  dotted el superior, solid para los laterales
y dashed el inferior */
```

```
border-style: dotted solid dashed;

/* los cuatro en el sentido de las agujas del reloj: red superior,
blue derecho, gree inferior y purple izquierdo */

border-color: red blue green purple;
```

IMPORTANTE: *Todos estos elementos son válidos para cualquier selector que apliquemos tanto a un elemento de línea como de bloque aunque, tienen mucho más sentido (sobre todo algunos de ellos) sobre uno de bloque.*

Existe otra propiedad llamada outline mucho menos flexible para aplicar bordes a las cajas.

```
outline: 3px solid black;
```

outline no "roba" espacio a las dimensiones de la caja y, por tanto, no afecta a las dimensiones de esta. Es, por decirlo de alguna forma, como si usáramos un borde exterior al borde de la caja. Por contra es mucho menos flexible que border y, por ejemplo, no podemos especificar propiedades diferentes a sus diferentes segmentos.

Imágenes de fondo en las cajas y sus propiedades

Nuestras cajas no sólo pueden tener un color de fondo: también pueden tener una imagen. La sintaxis de la propiedad es la siguiente:

```
background-image: url(http://cort.as/1KYH);
```

Si la imagen está en el propio directorio donde se encuentra la imagen y, por ejemplo, se llama burbujas.jpg sería así: url(burbujas.jpg). Al igual que ocurría con el valor del atributo src de la etiqueta img, si la imagen se encuentra en el propio servidor pero en otro directorio diferente podríamos especificar una ruta absoluta, relativa, etc.

Por defecto la imagen se coloca en formato de mosaico. Podemos controlar este valor mediante la propiedad background-repeat cuyos posibles valores son repeat, repeat-x, repeat-y o no-repeat. El valor por defecto es repeat.

Mediante la propiedad background-position podemos dar posición a nuestra imagen de fondo, tanto si estamos usando no-repeat, como si lo estamos haciendo (en cuyo caso posicionaría una de las imagenes y crearía el mosaico a partir de esta.) Tenemos tres posibilidades:

```
/* Posiciona la esquina superior izquierda de la imagen en la caja.
Está permitido el uso de valores negativos */
background-position: 50px 100px;

/* Idem al anterior pero usando porcentajes sobre las dimensiones de
la caja */
background-position: 20% 50%;

/* alinea la imagen en la caja admitiendo los valores left center o
right para la posición horizontal y top center o bottom para la
vertical. El orden es indiferente */
background-position: center center;
```

Otra propiedad interesante es background-attachment. Tiene dos posibles valores: scroll o fixed. El primer valor es el del comportamiento por defecto: la imagen se desplaza al mismo tiemp que se desplaza la página. El segundo permite que esta se fije en su posición y que lo único que se desplace sobre ella sea el contenido de la página.

Al igual que en otras de las opciones vistas, también tenemos un formato abreviado para definir todas estas propiedades de una sóla vez. Por ejemplo, si tenemos las siguientes reglas:

```
background-image: url(gotas.gif);
background-color: blue;
background-repeat: repeat-x;
background-position: center top;
background-attachment: fixed;
```

Podríamos sustituirlas por esta otra:

```
background: url(gotas.gif) blue repeat-x top center fixed;
```

El orden de los valores es indiferente y si omitimos alguno de ellos el navegador lo completará con la opción que tenga definida por defecto.

Cajas y jerarquía de elementos

Observa la forma en que aparece este código HTML en tu navegador usando la regla CSS correspondiente a la clase .ejemplo vista anteriormente:

```
<h1 class="ejemplo">Hola Mundo</h1>
<div class="ejemplo">Soy la caja número 1
       <div class="ejemplo">Soy la caja número 2</div>
</div>
<p>Hola <strong>mundo</strong> de las cajas. Esto es una prueba.</p>
```

Pon atención a dos cosas: en primer lugar la posición del segundo div se realiza de forma relativa al primero. Además, el párrafo final se superpone a la caja número 2 ¿verdad? Para entender la causa de ambas cosas piensa en el árbol formado por las etiquetas HTML. El segundo div es hijo del primero y por eso calcula su posición de forma relativa a la de su padre. Por el mismo motivo, como el párrafo final es hermano del primer div constituye su referencia para fijar su posición en la pantalla.

Posicionando las cajas y otros elementos

Hasta ahora no nos hemos preocupado de controlar la posición de los elementos que colocamos en nuestro HTML y dejamos que estos se vayan situando uno debajo o al lado del anterior y que sea el navegador el que lo disponga. Esto se debe a que, por defecto, usamos un modo de posicionamiento que se denomina estático (static). La propiedad CSS que lo controla se llama position.

Aparte de static, esta propiedad dispone de tres posibles valores más: relative, absolute y fixed.

Si usamos el valor relative, la posición se indica respecto a la que debería de tener de forma normal dicho elemento. Se hace mediante las propiedades top, bottom, left y rigth. left y top, las más usadas, aplican un desplazamiento medido desde la esquina superior izquierda del elemento pudiendo ser este negativo si se expresa mediante un valor mientras que bottom y right miden ese desplazamiento desde la esquina inferior derecha del elemento. Lógicamente, no tiene mucho sentido usar conjuntamente top y bottom o left y right. Se pueden usar porcentajes siendo estos relativos al tamaño del elemento padre al que pertenecen.

```
div.caja1 {position: relative; left: 10%; top -20px;}
div.caja2 {position: relative;right: -40px; bottom: 40px;}
```

Si usamos el valor absolute el elemento se posiciona exactamente en las coordenadas que indiquemos. La forma de indicar la posición usa las mismas propiedades vistas en el caso del posicionamiento relativo pero en este caso están referidas a su elemento padre. Si dicho elemento padre es el body del documento HTML estamos haciendo que la posición de este elemento sea independiente del resto del documento HTML de forma que nada que añadamos o quitemos a este puede alterar su posición. Si estás trabajando con cajas, no olvides que el margen exterior de la caja, es parte de la misma.

El último valor es fixed. Funciona de forma muy similar al anterior en cuanto a la forma de marcar el lugar donde lo posicionamos salvo que en este caso la posición del elemento queda fija incluso cuando nos desplazamos a través del documento HTML usando las barras de desplazamiento.

Las propiedades float y clear

La propiedad float, una de las más útiles en el mundo de las cajas, nos va a permitir modificar el posicionamiento de nuestros bloques y elementos de forma horizontal, colocándolas de forma flotante en el lugar que deseamos. Tiene tres posibles valores: right, left o none que es

su valor por defecto. Vamos a verlo con un ejemplo. Estas serán nuestras reglas CSS:

```
div#contenedor{width: 90%;
                 margin-left: 5%;
                 margin-top: 30px; }

div#top{width: 100%;
        background-color: green;
        text-align: center;
        color: white;
        font-weight: bold;
        font-size: 3em;
        font-family: Verdana, Helvetica, Arial, Sans; }

div#izquierda{width: 80%;
               height: 50%;
               background-color: aqua;
               float: left;}

div.caja{width:90%;
          margin-left:5%;
          margin-top:10px;
          background-color: red;}

div#derecha{width: 20%;
              height: 50%;
              background-color: teal;
              float: right; }

div#pie{width: 100%;
        background-color: orange;
        font-size: .8em;
        text-align: right;
        font-family: Courier, Monoespace;}
```

Y ahora pondremos esto en el body de nuestro HTML:

```
<div id="contenedor">
       <div id="top">Cabecera de la página</div>
       <div id="izquierda">
              <div class="caja">Caja 1<br/>       En un lugar de la
mancha de cuyo
              nombre no quiero acordarme no hace mucho que vivía un
              hidalgo... </div>
              <div class="caja">Caja 2</div>
              <div class="caja">Caja 3</div>
       </div>
```

```
        <div id="derecha">Esto está en la columna de la derecha</div>
        <div id="pie">Todos los derechos reservados</div>
</div>
```

Observamos que las cajas que usan el selector de id derecha e izquierda se posicionan, respectivamente, a derecha e izquierda de la superficie útil de su elemento padre, que aquí marcamos como contenedor. El resultado ya empieza a parecerse a una web actual, pero aún nos queda mucho trabajo para que pueda considerase aceptable desde el punto de vista estético. Observa también que puesto que hemos usado porcentajes en las dimensiones clave de las reglas, la página se redimensiona cuando cambiamos la superficie útil de nuestro navegador. Y una última cosa: para que te des cuenta de lo importante de las medidas en CSS, trata de modificar con una cantidad irrisoria el ancho de la caja izquierda (pon, por ejemplo, 80.005%) o introduce un pequeño padding en las reglas correspondientes al estilo del selector id izquierda y verás como se descuadra todo el conjunto.

float es un parámetro que se usa no sólo para posicionar cajas sino, por ejemplo, para posicionar una imagen dentro de un párrafo. La regla CSS:

```
        .aderecha {float: right; }
```

Y el HTML:

```
<img class="aderecha" src="cup.png" alt="copa" />

<p> Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed non
risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec,
ultricies sed, dolor. Cras elementum ultrices diam. Maecenas ligula
massa, varius a, semper congue, euismod non, mi.</p>
```

Por lógica, se trata de un atributo que no tiene ningún efecto cuando estamos usando posicionamiento absolute o fixed.

La propiedad clear nos permite anular los efectos de la propiedad float. Tiene cuatro valores posibles: none (por defecto), right, left o both.

Los dos primeros anularían un float right o left, respectivamente. El valor both los anularía a ambos. Para ver el efecto, añadamos la siguiente regla a nuestro CSS anterior:

```
.sinfloat {clear: right;}
```

Y ahora añadamos al final del HTML este nuevo párrafo:

```
<p class="sinfloat"> Lorem ipsum dolor sit amet, consectetuer
adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim
sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum
ultrices diam. Maecenas ligula massa, varius a, semper congue, euismod
non, mi.</p>
```

Como podemos ver, el nuevo párrafo salta a la parte inferior de la imagen como se habría comportado si esta no tuviera el atributo float mientras que el párrafo anterior sigue ateniéndose a su existencia.

Introduciendo la tercera dimensión en el documento

La propiedad z-index nos permitirá estructurar en tres dimensiones los elementos de nuestro documento HTML. Por defecto los elementos aparecen estratificados según su aparición en el documento, de forma que un elemento aparece siempre sobre todos los anteriores a él. La propiedad z-index nos permite alterar este orden indicando un número entero, positivo o negativo, que indica el nivel o la capa en la que situamos el elemento. Un ejemplo. Primero veamos las reglas CSS:

```
.caja1 { position: relative;
          background-color: yellow;
          border: 5px solid blue;
          z-index: 2; }

.caja2 { position: relative;
          background-color: purple;
          border: 5px solid blue;
          top: 1px;
          left: 80px;
          z-index: 1; }

.caja3 { position: relative;
```

```
        background-color: orange;
        border: 5px solid blue;
      top: -100px;
        left: -1px;
        Z-index: -1; }
```

Y luego este fragmento de HTML:

```
<p class="caja1">Hola caja 1</p>
<p class="caja2">Hola caja 2</p>
<p class="caja3">Hola caja 3</p>
```

Si prescindimos de la propiedad z-index, la caja 3 solapará a la 2 y esta a la 1. Con la propiedad z-index la caja 3 será la que vaya al fondo mientras que la 1 se encontrará en primer plano. Por defecto el valor de z-index es auto y el índice de capa base se asigna al 0. Por convención se suele usar el -1 como la capa de fondo y el 999 como la capa que queremos en primer plano, pero en realidad como hemos dicho, la propiedad admite cualquier valor entero negativo o positivo. Entenderás, además, que esta propiedad no tiene ningún efecto ante elementos que no se solapan.

Por último, como ves en la caja 1 se ha establecido la propiedad position como relative aunque no se efectua ningún desplazamiento sobre su posición normal. Esto es porque la propiedad z-index sólo tiene efecto sobre elementos en los que se han modificado las propiedades position o float.

Visibilidad, desbordamientos y recortes

La propiedad visibility, como su nombre indica, permite ocultar un elemento sin que este deje de ocupar su espacio en la página. Tiene dos posibles valores: visible o hidden y son autoexplicativos. Insisto: el elemento no se ve pero está ahí y si entramos en la opción de ver código del navegador (habitualmente Ctrl+U) podremos verlo, ¡así que cuidado con donde y para qué lo usamos!

La propiedad overflow le dice al navegador lo que tiene que hacer cuando un elemento no cabe completamente dentro de la caja o elemento que lo contiene. Su valor por defecto es auto y deja esta gestión al navegador. Otras posibilidades son hidden (corta y oculta lo que no cabe), scroll (coloca unas barras de desplazamiento para poder ver lo que no cabe) o visible (lo que no cabe desborda a su contenedor y se muestra en la página). Un pequeño código CSS para verlo

```
.caja { width: 200px;
         padding: 5px;
         height: 100px;
         border: 1px dashed black;
         overflow: hidden;}
```

Y este sería el HTML de prueba:

```
<div class="caja">
       <p> Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
       Sed non risus. Suspendisse lectus tortor, dignissim sit amet,
       adipiscing nec, ultricies sed, dolor.</p>
</div>
```

Modifica el valor de overflow para ver las diferentes posibilidades. Y cuidado. Al igual que ocurría con la propiedad visibility, si aquí usamos el valor hidden esto no implica que no se pueda ver en la vista de código de la página. Existen las propiedades overflow-x e overflow-y por si queremos controlar individualmente las dos posibles barras de scroll posibles.

La propiedad clip nos permite fijar la parte visible de un elemento definiendo una zona rectangular y sólo es aplicable en elementos cuya posición se ha fijado como absolute. La sintaxis es así:

```
clip: rect(0 200px 100px 0);
```

Esto dejaría visible un rectángulo que, respecto al elemento en el que se aplique, empieza a 0px de la parte superior, llega hasta los 200px hacía la derecha, 100px hacia abajo y 0px desde la izquierda (o sea, cuatro

coordenadas en el sentido de las agujas del reloj empezando desde arriba, como siempre)

Puedes ver un ejemplo completo con esta regla CSS:

```
.caja { width: 25%;
         padding: 5px;
         position: absolute;
        border: 1px dashed black;
        clip: rect(0 200px 100px 0);}
```

Y este código HTML que la usa:

```
<div class="caja">
       <p> Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
       Sed non risus. Suspendisse lectus tortor, dignissim sit amet,
       adipiscing nec, ultricies sed, dolor.</p>
</div>
```

Y la misma advertencia de siempre: recuerda que la parte que no es visible se puede ver con la opción de Ver código del navegador!