

Propiedades Modelo Caja

width

Hace referencia al ancho de un elemento, es privativa de los elementos de bloque o de los elementos de línea bloque, es decir, puedo aplicar esta propiedad a un div, un header, como así también a una imagen, un elemento de formulario, pero no a un vínculo, o a un strong o em porque estos son elementos de línea.

Se puede trabajar con medidas de longitud absolutas, recuerden qué son aquellas medidas que no dependen del dispositivo o entorno sino que siempre es lo mismo :

cm

mm

pt

pc

in , etc

También se puede trabajar con medidas relativas, medidas que sí varían dependiendo el dispositivo o entorno:

px

%

em

ex

Es importante aclarar, que estas medidas anteriores, no descubren grandes novedades pues ya las habíamos visto en el tamaño de la tipografía, sin embargo, en este caso el % (porcentaje), guarda una diferencia, pues en el caso de la propiedad font-size, el % se maneja con referencia a la tipografía base, sin embargo en este caso, es el %(porcentaje) en referencia al ancho del contenedor padre, por ejemplo:

```
#encabezado div {  
width: 30%;  
}
```

Es equivalente a decir, que el div es el 30% del valor del width del elemento cuyo id es #encabezado, por lo tanto, en este caso si vemos la regla previamente escrita:

```
#encabezado, #pie {  
width: 100%;  
background-color: lightgray;  
}
```

Es el 30% del 100%, teniendo en cuenta que el 100% es el *total del ancho de la pantalla* ya que en nuestro HTML, el #encabezado se encuentra de la siguiente forma anidado en el *body* :

```
<body>  
  <header id="encabezado"> </header>  
</body>
```

height

El alto de un elemento , es fijado a través de esta propiedad, donde se puede tener valores de medidas de longitud tanto absolutos como relativos.

El height , no es obligatorio y podemos omitir, siendo que los elementos sin height tienen como valor **predeterminado : auto**, por tanto eso significa que se adaptarán a su contenido.

```
#encabezado, #pie {  
width: 100%;  
background-color: lightgray;  
height: 90px ; }
```

De esta forma anterior, establecemos que el #encabezado, y el #pie, van a tener estas mismas características.

Otro detalle no menor, es que si por caso, colocamos un height pero el contenido desborda al contenedor, se verá en el navegador de la siguiente manera:

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Nostrum, possimus alias quos recusandae hic odio, harum quae amet maiores

Es decir qué naturalmente, es contenido desbordado se ve, esto tiene relación con otra propiedad overflow.

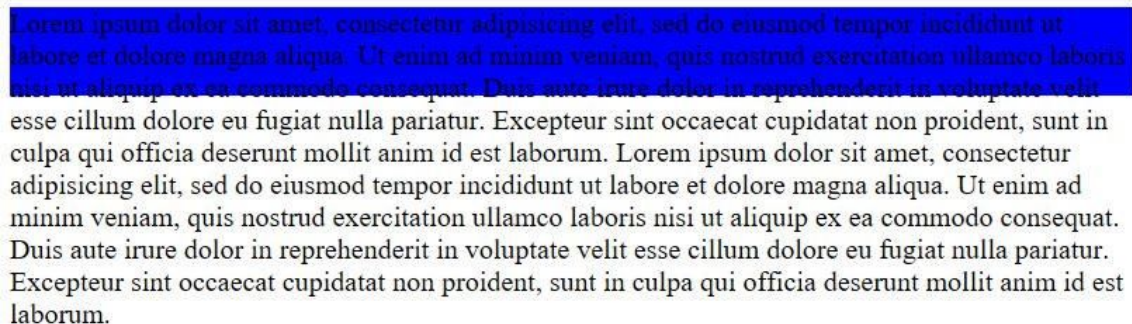
overflow

Está propiedad permite establecer con el contenido cuando desborda a su contenedor.

Valores posibles:

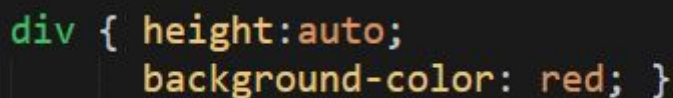
#1 Visible

El contenido se visualiza por más que el contenedor sea más corto o el height no alcance a cubrir todo el contenido.



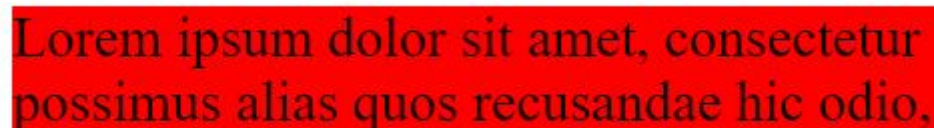
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Si el height , fuese o tuviese su valor predeterminado qué es **auto**, entonces este problema y está propiedad no se aplicarían porque el valor auto de **height** permite qué el contenedor fluya con su contenido por ejemplo



```
div { height:auto;
      background-color: red; }
```

En el navegador se verá de la siguiente forma:



Lorem ipsum dolor sit amet, consectetur
possimus alias quos recusandae hic odio,

#2 Hidden

Este valor, hace qué el contenido qué se desborda se enconda. Por ejemplo en nuestro editor se verá de la siguiente manera:

```
div {  
  height: 2em;  
  background-color: red;  
  overflow: hidden; }
```

En el navegador no se podrá ver el total del párrafo:



Lorem ipsum dolor sit amet.

#3 Scroll

El valor scroll muestra ambas barras de scroll para poder ver el resto del contenido, sea necesario o no, por ejemplo:

```
div {  
  height: 2em;  
  background-color: red;  
  overflow: scroll; }
```

En el navegador:



#4 Auto

En el editor aplicaremos este valor de la siguiente forma:

```
div {  
  height: 2em;  
  background-color: red;  
  overflow: auto; }
```

El valor auto, sólo muestra las barras de scroll si son necesarias, por ejemplo en el navegador se visualizará de la siguiente manera:



margin

Esta propiedad fija un espacio entre elementos contiguos, se puede trabajar con medidas de longitud ,

absolutas

cm

mm

pt

pc

in

También se puede trabajar con medidas relativas

px

%

em

ex

```
<!DOCTYPE html>
<html>
<head>
  <title> CSS</title>
  <style>

  a {
margin-top: 12px;
margin-left: 10%;
margin-right: -12px;
margin-bottom: 10px;
  }

</style>
</head>
<body>
```

O también valiéndonos del shorthand:

```
#encabezado {
width: 100%;
margin-top: 1em;
background-color: lightgray;
}
}
```

En el caso anterior los cuatro lados son iguales, si por caso generó otras alternativas:

margin : valor;

==

margin-top: valor;

margin-bottom: valor;

margin-left: valor;

margin-right: valor;

```
#encabezado {
width: 100%;
margin-top: 1em;
background-color: lightgray;
}
}
```

margin: valor1 valor2;

margin-top: valor1;

margin-bottom: valor1;

margin-left: valor2;

margin-right: valor2;

Ejemplo:

```
#seccion-principal img {
width: 50%;
float: left;
margin: 1em 1%;
}
```

margin: valor1 valor2 valor3;

margin-top: valor1;
margin-bottom: valor3;
margin-left: valor2;
margin-right: valor2;

```
#seccion-principal img {  
width: 50%;  
float: left;  
margin: 1em 1% 2em;  
}
```

margin: valor1 valor2 valor3 valor4;

margin-top: valor1;
margin-bottom: valor3;
margin-left: valor4;
margin-right: valor2;

Ejemplo:

```
#seccion-principal img {  
width: 50%;  
float: left;  
margin: 1em 1% 2em 2%;  
}
```

Hay elementos que tienen asignado, este tipo de shorthand, por ejemplo el body:

```
body {  
display: block;  
margin: ▼ 8px;  
margin-top: 8px;  
margin-right: 8px;  
margin-bottom: 8px;  
margin-left: 8px;  
}
```

Si queremos quitar el margin lo haremos de la siguiente manera:

```
body {font: 90% 'Lato', sans-serif; margin:0;}
```

Por ejemplo también podemos ver , de forma predeterminada un margin individualizado:

```
h1 {
  display: block;
  font-size: 2em;
  -webkit-margin-before: 0.67em;
  -webkit-margin-after: 0.67em;
  -webkit-margin-start: 0px;
  -webkit-margin-end: 0px;
  font-weight: bold;
}
```

Y en mi caso también quiero quitarlo y lo haré de la siguiente forma:

```
h1 { font-size: 3em; margin: 0; }
```

padding

Esta propiedad indica el espacio entre los bordes de un elemento y su contenido. A diferencia del margin , pocos elementos tienen padding, por ejemplo las listas tanto ordenadas como desordenadas tienen , si queremos quitarles este espacio no tendremos más que:

```
<!DOCTYPE html>
<html>
<head>
  <title> CSS</title>
  <style>

<style>
ul, ol { margin: 0; padding: 0;}

</style>

</style>
</head>
```

Tiene en cuanto a su sintaxis la misma construcción que margin, por ejemplo:

```
#encabezado, #pie {
width: 100%;
background-color: lightgray;
height: 90px ;
padding: 1%;
}
```


En este caso, los cuatro lados son iguales. O si trabajamos con 4 valores distintos, debemos seguir el sentido de las agujas del reloj.

Por otro lado, es importante saber, qué el padding suma al total del ancho de un elemento, por ejemplo si en nuestra estructura ponemos 1% de padding al encabezado pero al main no, por más que ambos de width tengan 100% el resultado será el siguiente:



Es decir uno queda más grande que el otro, pues ahora el #encabezado, no es más 100% sino 100% más 1% de padding de ambos lados.

De esta forma, entonces, debemos implementar una propiedad que permite hacer que el padding forme parte del width y no que se sume, esta propiedad es **box-sizing**

box-sizing

Esta propiedad tiene dos valores posibles que explicarán su uso:

content-box

El $width\ total == width + padding + border$

border-box

El $width\ total = width(padding + border)$

```
* {box-sizing: border-box;}
```

border

Esta propiedad permite indicar el borde de un elemento, si bien es una propiedad fácil tiene varias aristas que veremos a continuación:

border-style

Es el estilo del borde tiene los siguientes valores:

- dotted** - Borde un punto al lado del otro
- dashed** - Borde una línea al lado de la otra
- solid** - Borde Sólido
- double** - Borde Doble
- groove** - Borde 3d
- ridge** - Borde 3d
- inset** - Borde hacia adentro
- outset** - Borde hacia afuera
- none** - No hay borde

Por ejemplo en el caso de los campos de texto o en general cualquier elemento de formulario nos pasa que muchas veces queremos quitarle el borde lo haremos de la siguiente manera:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>

  <style>
input { border-style: none; }

  </style>
</head>
<body>

<input type="text">

</body>
</html>
```

border-width

Me permite establecer el grosor o ancho del borde, acá podemos trabajar con cualquier medida de longitud de las ya vistas en otras propiedades como width, height o padding.

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>

  <style>
div { border-width: 1em; border-style: solid; }

  </style>
</head>
<body>

<div>
  <p> div con borde </p>

</div>

</body>
</html>

```

border-color

El color del borde se puede aplicar con cualquier valor de los ya conocidos, valores hexadecimales, palabras o también cantidad de rgb() o rgba()

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>

  <style>
div { border-width: 1em; border-style: solid; border-color: rgba(0,0,0,0.5); }

  </style>
</head>
<body>

<div>
  <p> div con borde </p>

</div>

</body>
</html>

```

outline

El outline es un borde luego del border, por eso toma el nombre de outline, y está por fuera del elemento, si bien al principio puede parecer una propiedad un tanto extraña, cobra sentido porque está de forma predeterminada en elementos de formularios tales como el campo de texto que un poco más arriba elaboramos. Ese campo al cual el habíamos quitado el border con la siguiente regla:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>

  <style>
input { border-style: none; }

  </style>
</head>
<body>

<input type="text">

</body>
</html>
```

Al quitarle ese **border** pero visualizarlo en el navegador, muestra al hacerle foco (cuando empiezo a escribir en él) un borde azul o celeste (dependiendo del navegador) , ese es el **outline**.



El **outline** cómo el **border** posee las mismas propiedades es decir:

#1 outline-width

#2 outline-style

#3 outline-color

Ejemplo desde el editor:

```
<style>
input { outline-width: 1em; outline-color: red; outline-style: dashed ; }

</style>
```

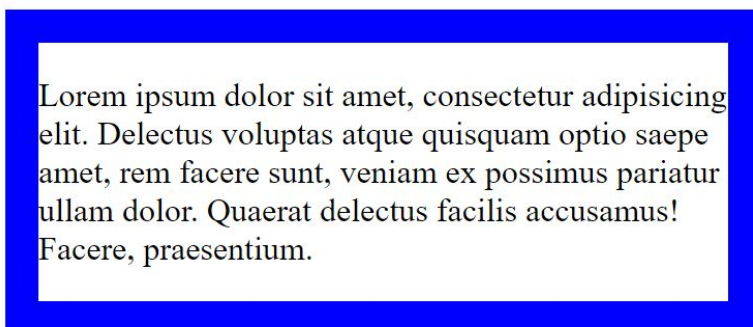
Ejemplo desde el navegador:



Luego también ambas propiedades tienen shorthands, qué si bien son fáciles de aprender al poseer tantas propiedades relacionadas (width, style, color) las posibilidades son amplias, veamos algunos ejemplos:

```
div { width: 50%; border: 1em solid blue;}
```

Esta regla de estilo, generará el siguiente resultado:



A tener cuidados, porque si olvidamos poner el style en el shorthand, el resultado será que el borde desaparecerá, ya que el width indicarlo o el color no es obligatorio pero sí el style, por eso la siguiente regla está mal :

```
<style>
div { width: 50%; border: 1em blue;}

</style>
```

También puedo encarar solo un lado, por ejemplo, si quiero lograr el siguiente resultado:

- Lorem ipsum dolor sit amet, consectetur adipisicing elit. Delectus voluptas atque quisquam optio saepe
- amet, rem facere sunt, veniam ex possimus pariatur ullam dolor. Quaerat delectus facilis accusamus!
- Facere, praesentium.

El editor deberá decir lo siguiente:

```
<style>
div { width: 50%; border-left: 1em dotted red;}

</style>
```

Por otro lado, si por caso los lados son distintos podemos manejarnos con la lógica de margin o padding de la siguiente manera:

```
<style>
div { width: 50%; border-width: 1em 2%;
      border-color: blue red green orange;
      border-style: solid outset inset;

      }

</style>
```

Lograremos el siguiente resultado:

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Delectus voluptas atque quisquam optio saepe amet, rem facere sunt, veniam ex possimus pariatur ullam dolor. Quaerat delectus facilis accusamus! Facere, praesentium.

Con outline la lógica es equivalente, por caso si mis cuatro lados son iguales haré lo siguiente:

```
<style>
div { width: 50%;
      border-width: 1em 2%;
      border-color: blue red green orange;
      border-style: solid outset inset;
      outline: 0.4em solid rgba(0,0,0,0.5);
      }

</style>
```

El resultado será el siguiente:

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Delectus voluptas atque quisquam optio saepe amet, rem facere sunt, veniam ex possimus pariatur ullam dolor. Quaerat delectus facilis accusamus! Facere, praesentium.