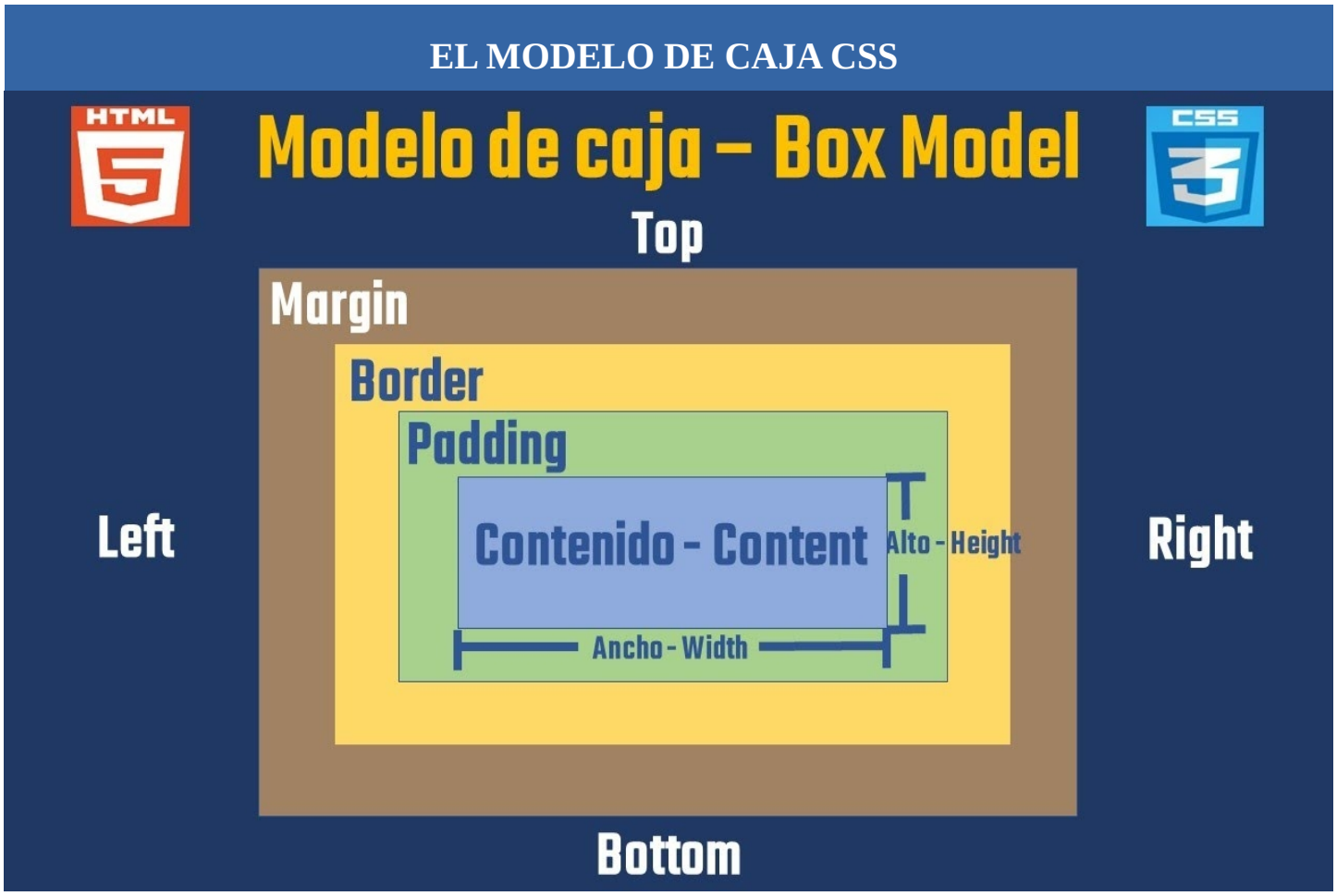
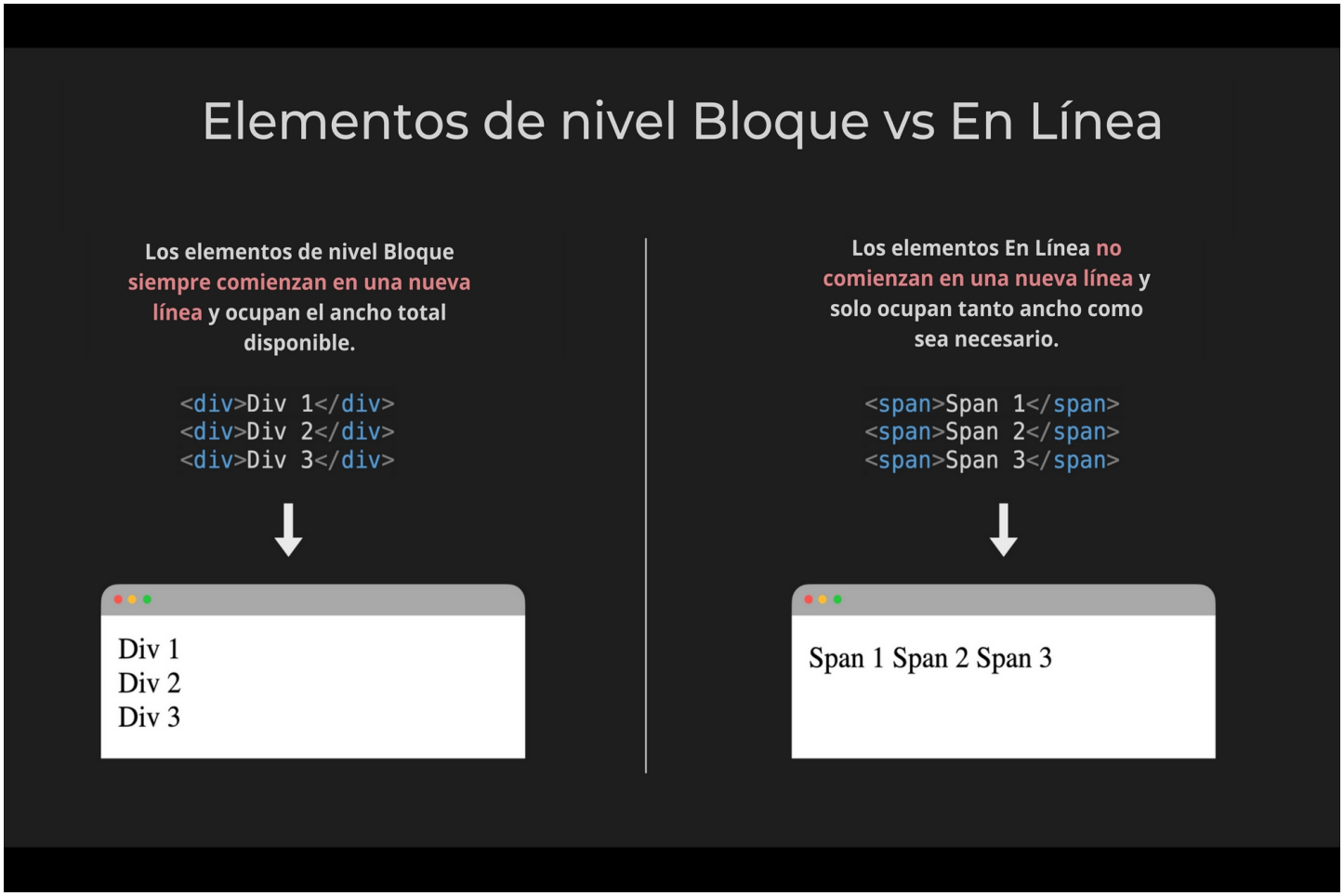
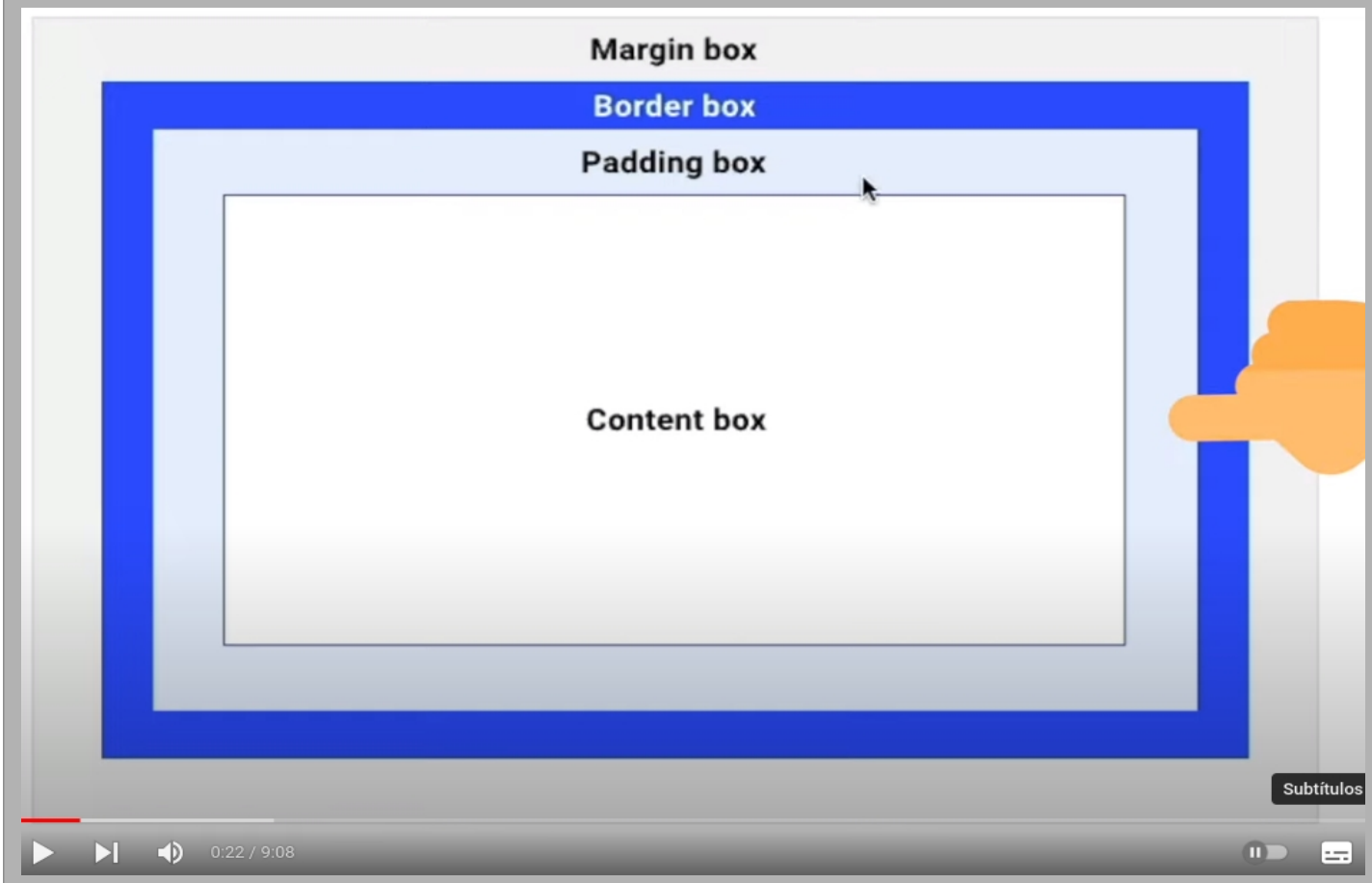
EL MODELO DE CAJA CSS



DIV y SPAN



Vídeo YouTube del Modelo CAJA

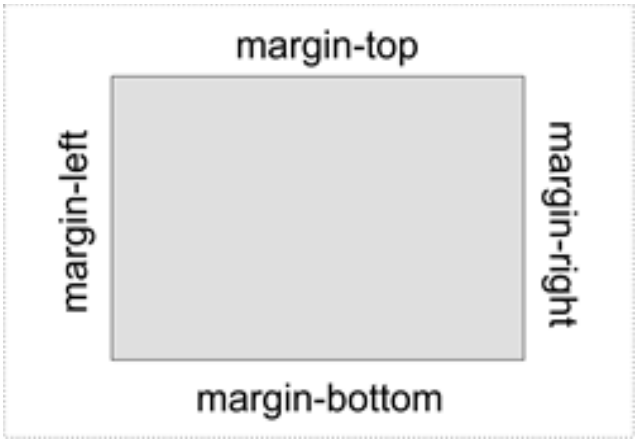


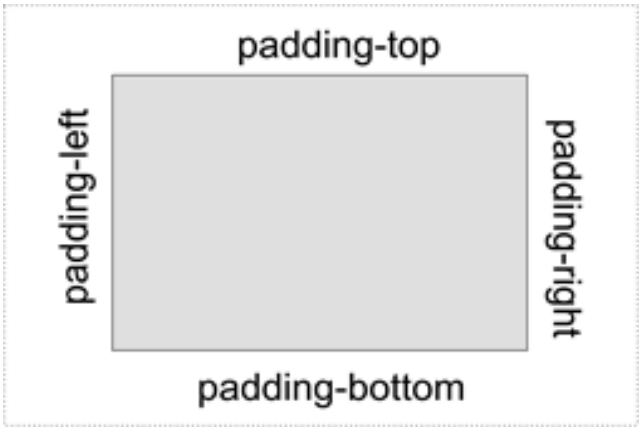
Click en el título amarillo para abrirlos

Vídeo de YouTube BOX-SIZIN

Estos elementos se pueden ver como **cajas o contenedores dentro del cual se puede poner cualquier contenido HTML**.

margin y padding





Estos definiciones de la clase margenes son equivalentes:

```
.margenes {
        margin-top: 2px;
        margin-rigth: 4px;
        margin-bottom: 6px;
        margin-left: 8px;
}
 .margenes {
        margin: 2px 4px 6px 8px;
}
```

BORDES

Los bordes aceptan varios atributos para darles el grosor, color y estilo, curva en las esquinas.

Cada border se compone de cuatro bordes diferentes uno para cada lado de la caja: border-top, border-right, border-bottom y border-left. Todas se pueden abreviar en una sola, border, si todas sus propiedades son iguales .

PROPIEDAD	USO	VALOR
border-width	un valor indicando el grueso de la linea	valor numérico
border-style	El tipo de linea para dibujar el borde	dashed, dotted, double, groove, inset, outset, ridge, solid, inherit, hidden, none
border-color	El color de la linea	código de color
border-radius	El redondeado de las esquinas	Largo del radio de la curva
border-top	Borde superior	
border-right	Borde derecho	
border-bottom	Borde inferipor	
border-left	Borde izquierdo	

Si se quieren aplicar las propiedades individuales ancho, estilo y color para cada lado añadirías en medio del nombre de la propiedad el lado que quieras formatear border-top-width o border-bottom-style. .

Para el radio de las esquianas tendrías que usar una combinación de lados por esquina: superior-derecha sería border-top-right-radius

Se pueden abreviar todas las propiedades de los lados para todos los bordes usando una sola propiedad border:

```
border: 2px solid red;
```

Un ejemplo sencillo para una caja con borde redondeado en las cuatro esquinas

```
div {
        border-width:2px solid #a1a1a1;
        border-style: solid;
        border-color: #a1a1a1;
        padding:40px;
        background:#dddddd;
        width:300px;
        border-radius:25px;
}
```

Una muestra de los distintos estilos de bordes que pueden usarse en las páginas web, algunos apenas son perceptibles. Se pueden usar includo para dar cierto aspecto 3D.

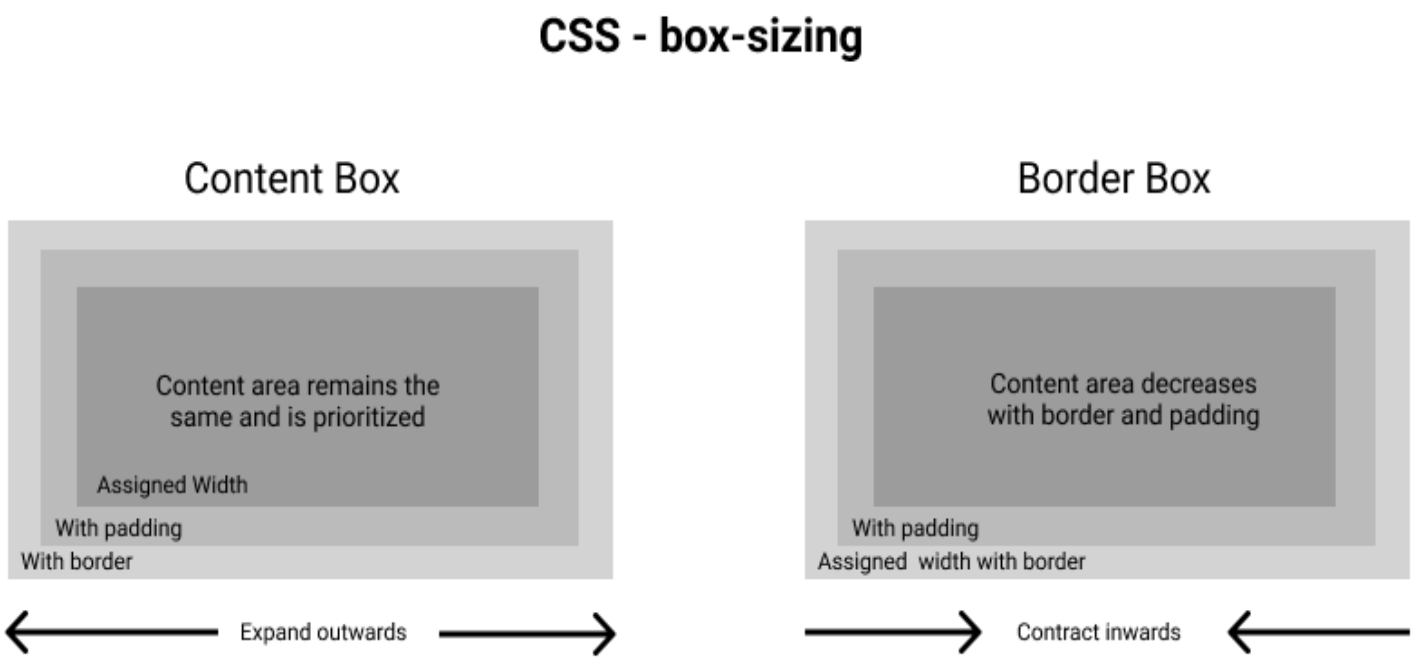
dashed. dotted. double. groove. inset. outset. ridge. solid.

Un ejemplo es éste código

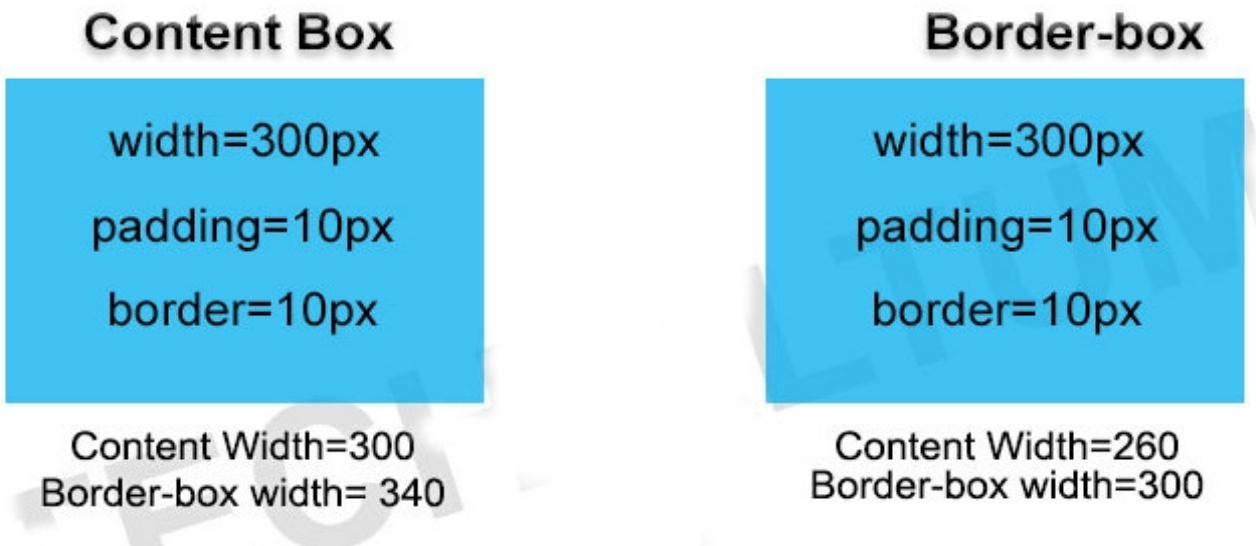
```
#imgborde {
        width: 300px;
        height: 200px;
        padding: 10px;
        border-width: 10px;
        border-color: blue; border-style: solid;
}
```

Si estás en un explorador no compatible se verá el borde normal con el color puesto en border-color, si le pones **transparent** no se vería ningún borde.

box-sizing: border-box



- **Content-box:** Las propiedades de ancho y alto (y las propiedades mínimas y máximas) incluyen solo el contenido. El borde, el relleno o el margen no están incluidos
- **border-box**: Las propiedades de ancho y alto (y propiedades mínimas y máximas) incluyen contenido, relleno y borde, pero no el margen.



Se hace difícil de calcular de forma predecible el ancho o alto de nuestros elementos si tienen padding o border. Para arreglar esto tenemos el valor **border-box.**

El valor border-box en el box-sizing hace que el padding y el border pasen a formar parte del cálculo del ancho de la caja y no lo suman posteriormente.

La propiedad de CSS **box-sizing** indica cómo se deben calcular las medidas de un elemento.

Por defecto **CSS considera** que el **ancho y alto** de la caja es de las propiedades width y height. **Si le añades un padding o un border** el tamaño de renderizado de la caja será:

Ancho total de la Caja: width + padding + border.

Alto total de la Caja: height + **padding + border**.

En el siguiente ejemplo tendríamos una **caja de 290px de ancho** ya que: *250px de width + (10px * 2) de padding + (10px * 2) de border*

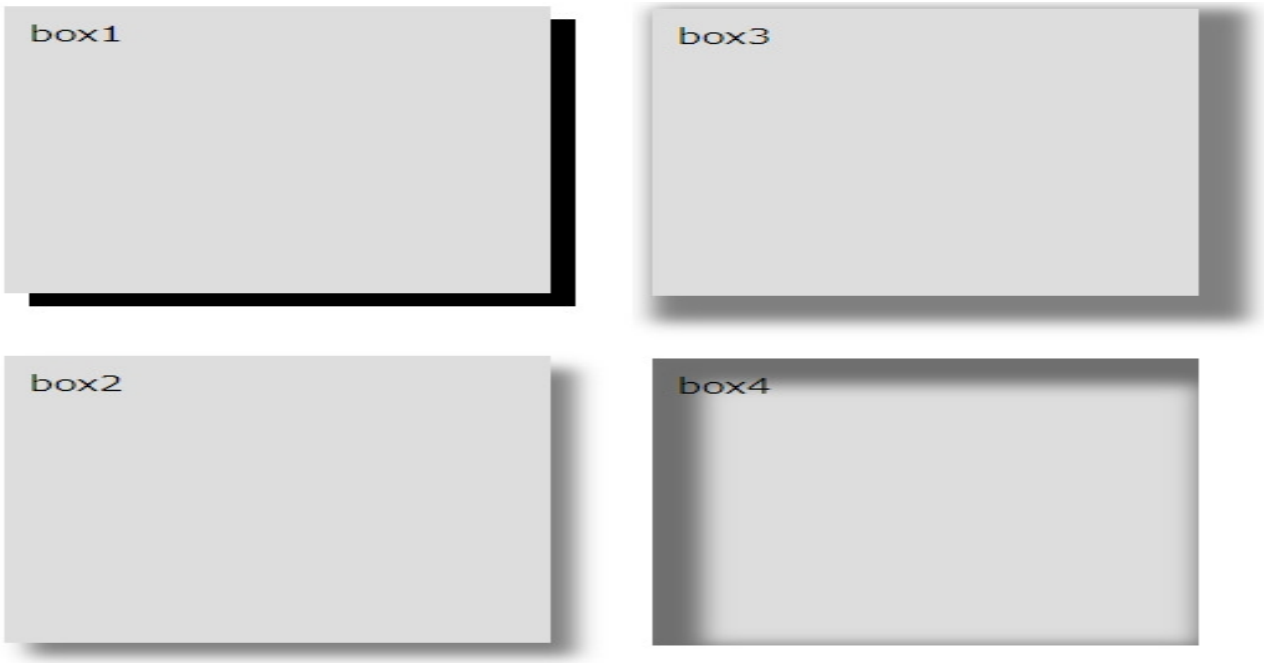
div {width: 250px;border: 10px;padding: 10px;}

En el siguiente ejemplo tendríamos una caja de 250px de ancho ya el padding y el border ya forman parte del cálculo del width del elemento:

```
div {
  box-sizing: border-box;
  width: 250px;
  border: 10px;
  padding: 10px;
}
```

Y con esto simplificamos nuestra vida bastante ya que si queremos una caja que mida 250px de esta forma tendremos la seguridad que será de la medida que le hemos indicado sin importar si usamos bordes o paddings

BORDES CON SOMBRAS



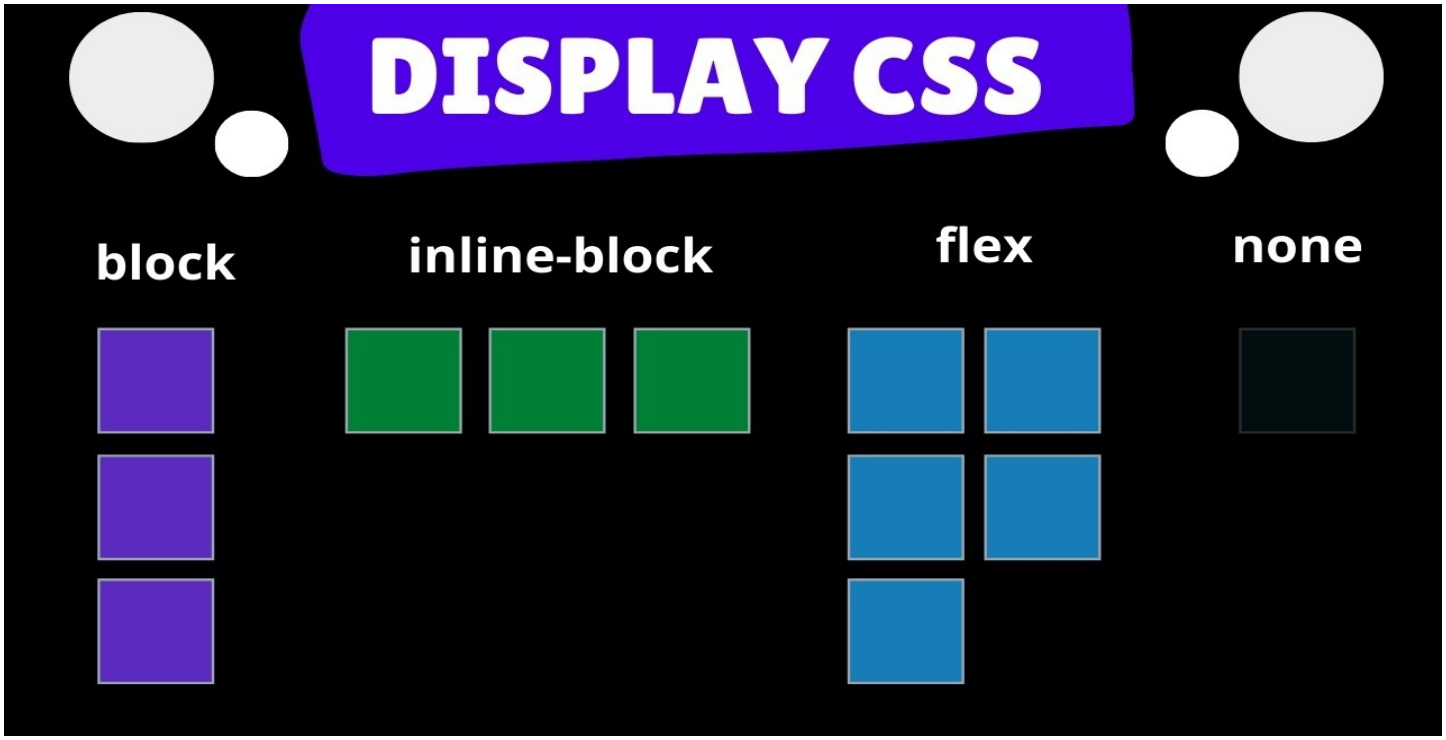
Además de los bordes lineales y los personalizados con imágene, podemos construir cuadros contenedores con un efecto de sombra.

Para eso usaremos la propiedad **box-shadow**. Esta caja o bloque aparecerá como si estuviera por encima del texto de la página o deprimido, como hundido entre el texto y el fondo.

ATRIBUTO	USO	VALOR
inset	Efecto de hundido bajo el texto. Es opcional	
offset-x **offset-y**	Distancia de la sombra al borde. Admite valores positivos y negativos.	Número
blur-radius	Es la cantidad de difuminado, a mayor valor mayor efecto difuminado	Número
spread-radius	Es el ancho de la sombra	Número
color	El color de la sombra. Es opcional	Código color

```
#cajasombra {
        background-color: #ddd;  width: 300px;
         padding: 10px;
        box-shadow: 5px 5px 3px #333;
}
```

CAJAS Y CONTENIDO



Una elemento de una página web que contenga otros elementos (un contenedor o caja) debe estar definida con las dimensiones adecuadas, pero esto no siempre es posible.

CSS ofrece reglas para hacer frente a estas situaciones. Para textos tiene la regla complementaria **text-overflow** que permite recortar. Y para situaciones más generales ofrece las reglas **overflow** con la que podemos optar por distintas opciones cuando el contenido es demasiado grande

- **visible** - Es el valor por defecto, el contenido sobre pasa los límites del contenedor
- **hidden** - En este caso la proporción que no cabe queda oculta
- **scroll** - Se muestran barras de scroll
- **auto** - Es un scroll automático

Al utilizar tamaños debes tener muy en cuenta el diseño responsive:

Una tamaño absoluto puede fallar al verlo en un móvil, mejor usa tamaños relativos: Porcentajes (**%**) o unidades relativas (**em, rem, …**)

Puedes controlar el **tamaño máximo de altura y anchura** con las propiedades

- **max-width**
- **min-width**
- **max-height**
- **min-height**

```
.caja {
        resize:both;
        max-width: 600px;
        width: 400px;
        max-height: 700px;
        overflow:auto;
        border: solid thin black;
}
```

DISPLAY

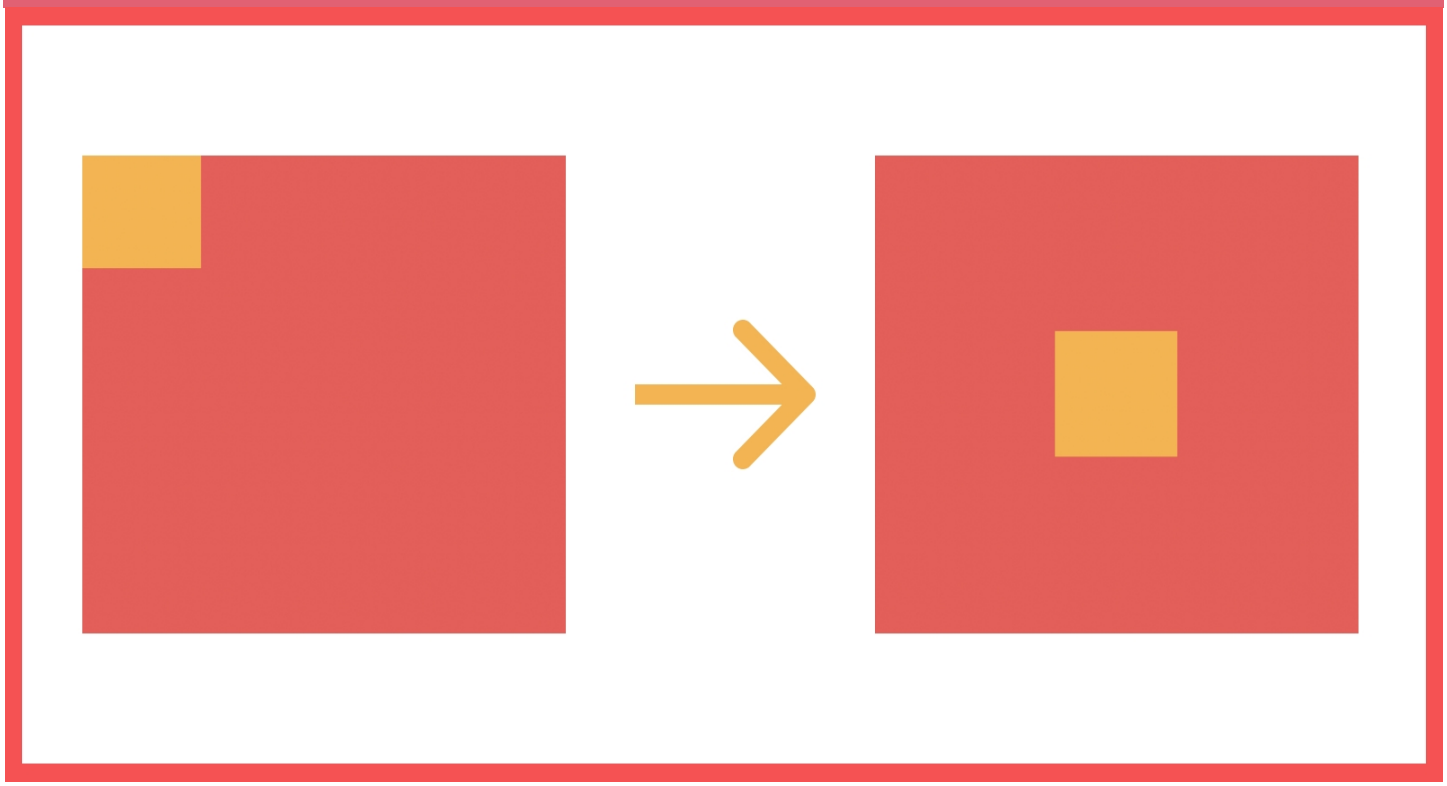
Para eso utiliza la propiedad **display** o sea, mostrar que dice como mostrar el elemento. Básicamente los valores más usados son:

- **block**: el elemento es un bloque (como un párrafo)
- **inline**: es un elemento que va en linea, en el flujo natural del texto
- **flex**: modo files columnas
- **grid**: tipo cuadrícula
- **table:** como una tabla
- **table-cell, table-roe, teble-column**: usados para los contenidos de las tablas
- **display: none**

VISIBILITY

- **visibillity: visible**
- **visibillity: hidden**

CÓMO POSICIONAR UN BLOQUE - POSITION



El **archivo HTML principal** es así:

```
<!DOCTYPE html>
<html lang="es">
        <head>
                <meta charset="UTF-8" />
                <meta http-equiv="X-UA-Compatible" content="IE=edge" />
                <meta name="viewport" content="width=device-width, initial-scale=1.0" />
                <title>Centrando divs</title>
                <link rel="stylesheet" href="" />
                <style>
                        * {
                          box-sizing: border-box;
                          margin: 0px;
                          padding: 0px;
                        }
                </style>
        </head>
        <body>
                <div id="contenedorPadre">
                        <div id="contenedorHijo"></div>
                </div>
        </body>
</html>
```

LA PROPIEDAD POSITION

```
#contenedorPadre {
        position: relative;
}
#contenedorHijo {
        position: absolute;
        top: 50%;
        left: 50%;
        transform: translate(-50%, -50%);
}
```

La propiedad **POSITION** en CSS establece cómo se posiciona el elemento en la página. El valor por defecto de la propiedad position es static. Los otros valores que toma la propiedad position son: **relative, absolute, fixed y sticky**.

Ahora bien, cuando damos un **position: absolute** a un elemento del DOM, **éste se vuelve absoluto con respecto a toda la página**. Esto sería útil si quisiéramos centrar el div con respecto a toda la página.

Por otro lado, **al poner el elemento padre en position: relative, hace que el elemento hijo (con position: absolute) sea absoluto, relativo al elemento padre y no a toda la página.**

En el ejemplo anterior hacemos precisamente eso. Le damos al elemento padre una position: relative y al hijo una position: absolute.

Junto con la propiedad **position**, podemos especificar otras cuatro propiedades: **top, right, bottom y left**, que determinan la posición final del elemento.

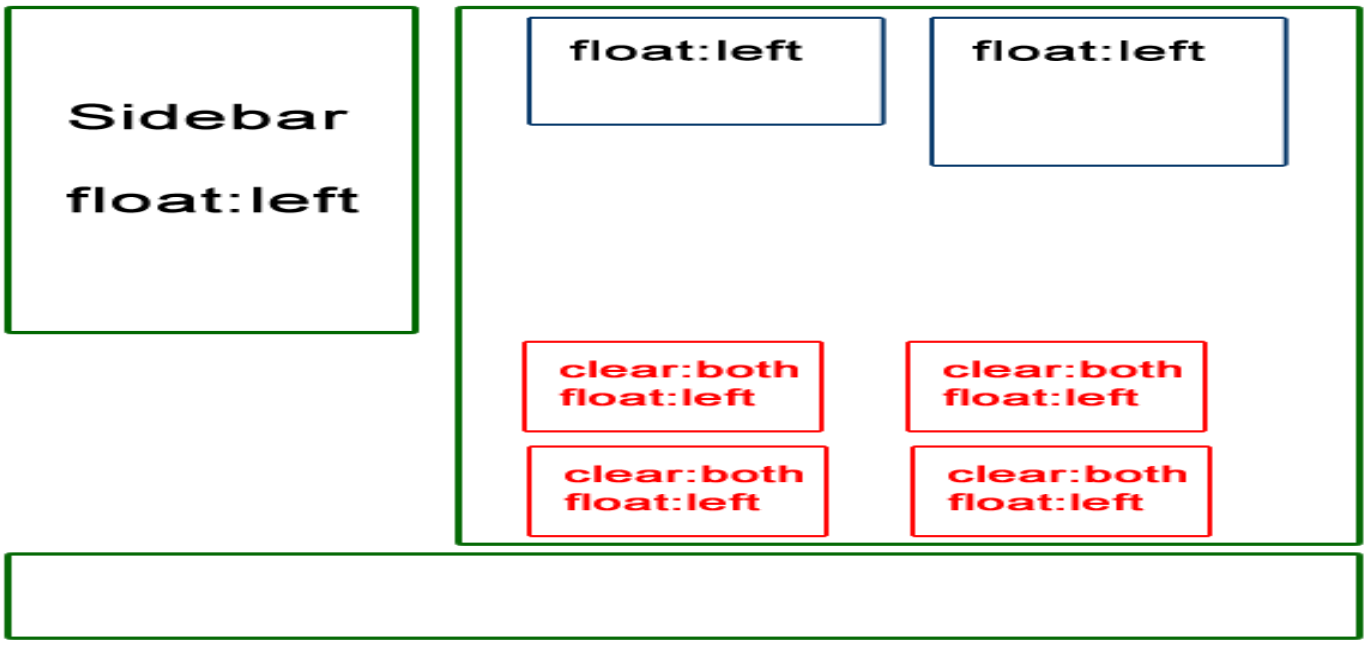
La posición top y bottom especifican la **posición vertical** del elemento, mientras que left y right especifican la **posición horizontal**.

CÓMO USAR POSITION: **relative, absolute,** valores de desplazamiento: **top, left, right, bottom y margin: auto**

```
#contenedorPadre {
        position: relative;
}
#contenedorHijo {
        position: absolute;
        top: 0;
        left: 0;
        bottom: 0;
        right: 0;
        margin: auto;
}
```

Si mencionamos sólo **top: 0, bottom: 0 y margin: auto**, centra el elemento hijo verticalmente.

LA PROPIEDAD FLOAT



Con la propiedad float podemos conseguir que un elemento **«flote» a la izquierda o a la derecha de otro elemento**. Para ello podemos utilizar las siguientes propiedades:

PROPIEDAD	VALOR	SIGNIFICADO
float	none | left | right	Cambia el flujo para que el elemento flote a la izquierda o derecha.
clear	none | left | right | both	Impide que los elementos puedan flotar en la orientación indicada.

De esta forma, utilizando **la propiedad float podemos cambiar donde aparecía un elemento, y mediante la propiedad clear podíamos resetearlo**.

Veamos algunos **EJEMPLOS** para entenderlo bien

CSS	BODY

ul {
background: grey;
padding: 0;
}
li {
list-style-type: none;
background: blue;
width: 100px;
padding: 8px;
margin: 8px;
color: white;
}
ul, li {
float: left;
}
	
	 - Primer elemento
	 - Segundo elemento
	 - Tercer elemento
	 - Cuarto elemento
	 - Quinto elemento

Con esto conseguimos que los ítems de la lista floten uno a continuación de otro. No obstante, **hoy en día** para conseguir este comportamiento siempre recomiendo utilizar la propiedad display en lugar de float. De esta forma se suele conseguir una solución y código más limpio y organizado.

EJERCICIO:

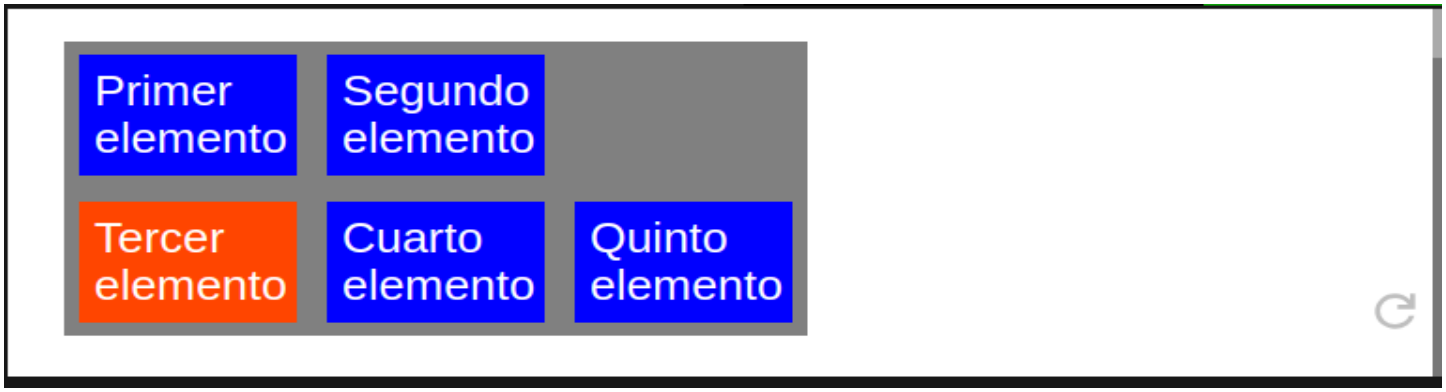
1. Construir el documento HTML y Comprobar el resultado en el navegador
2. Probar el resultado para los distintos valores de la propiedad float

LA PROPIEDAD CLEAR		
PROPIEDAD	**VALOR**	**SIGNIFICADO**
clear	**left | right | both**	Se encarga de impedir elementos flotantes en la zona indicada, a la izquierda del elemento (*left*), a la derecha (*right*) o en ambos lados (*both*).

Si queremos que, a partir del segundo elemento se posicionen en la siguiente línea:

CSS	BODY

ul {
background: grey;
padding: 0;}
li {
list-style-type: none;
background: blue;
width: 100px;
padding: 8px;
margin: 8px;
color: white;}
ul, li {
float: left;
}
.fix {
background: orangered;
clear: both;}
	
	 - Primer elemento
	 - Segundo elemento
	**<li class="fix">Tercer elemento **
	 - Cuarto elemento
	 - Quinto elemento



La propiedad float es una propiedad que podría ser interesante en determinadas condiciones, sin embargo, **utilizarla para creación de layout o colocación de elementos, como se hacía en el pasado, si se puede considerar una mala práctica**, ya que el código resultante suele ser más sucio y complejo de lo que sería mediante otros métodos actuales, como **Flex** o **Grid CSS**.