

## RESUMEN T4: Diseño Web

---

[4.1. Cajas](#)

[4.2. Modelo de caja tradicional.](#)

[Cajas Flotantes](#)

[Con Float](#)

[Sin Float](#)

[¿Qué es \*\*clear\*\* ?](#)

[Columnas](#)

[4.3. Modelo de caja flexible](#)

[Contenedor flexible](#)

[Organizando elementos flexibles.](#)

---

### 4.1. Cajas

Los navegadores crean cajas virtuales alrededor de los elementos para marcar el área que ocupan.

Se pueden organizar de estas dos formas:

- **Block (bloque):** El tamaño es personalizado y generan salto de línea.
- **Inline (en línea):** Tiene un tamaño predeterminado por caja y no generan.

Se escriben utilizando la variable **display**, ejemplo:

```
header {  
    display: block;  
    display: inline;  
    display: none;  
}
```

Con el valor **none** lo que haríamos sería eliminar elementos de la página.

Podemos usar tambien la propiedad llamada **visibility** que nos deja elegir entre los valores:

- visible
- hidden

Aunque lo escondamos, el hueco en la página se quedará.

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
h3 {  
    visibility: hidden;  
}  
</style>  
</head>  
<body>  
  
<h1>The visibility Property</h1>  
  
<h2>This heading is visible</h2>  
  
<h3>This heading is hidden</h3>  
  
<p>Notice that the hidden heading still takes up space on the page.</p>  
  
</body>  
</html>
```

The visibility Property  
This heading is visible

Notice that the hidden heading still takes up space on the page.

Usando la variable **inline-block** conseguimos hacer que los elementos inline como `<span>` puedan tener un tamaño personalizado.

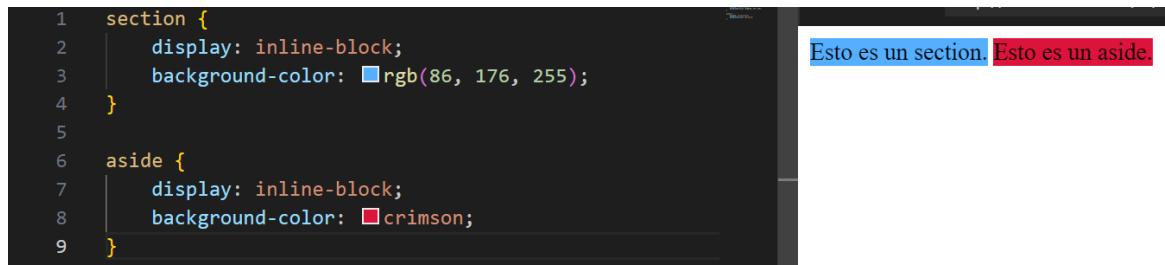
## Elementos Inline-Block

Contenido 1

Contenido 2

Contenido 3

Con esto, nos permite crear secciones en nuestra página web en la misma línea.



```
1  section {
2      display: inline-block;
3      background-color: #rgb(86, 176, 255);
4  }
5
6  aside {
7      display: inline-block;
8      background-color: #crimson;
9 }
```

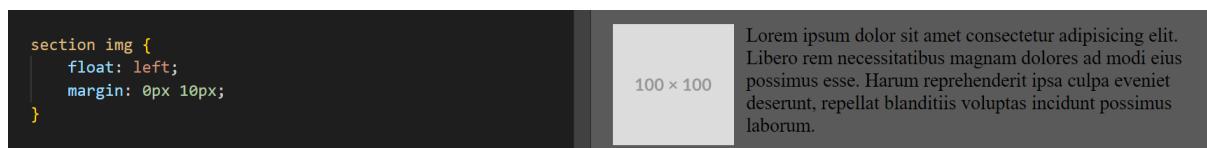
El conjunto de reglas que dicen como se va a ver una caja, el espacio que va a ocupar y como se organiza, se llaman "*modelo de cajas*."

Los estándar de modelos de cajas son:

- El modelo de caja tradicional.
- El modelo de caja flexible.

## 4.2. Modelo de caja tradicional.

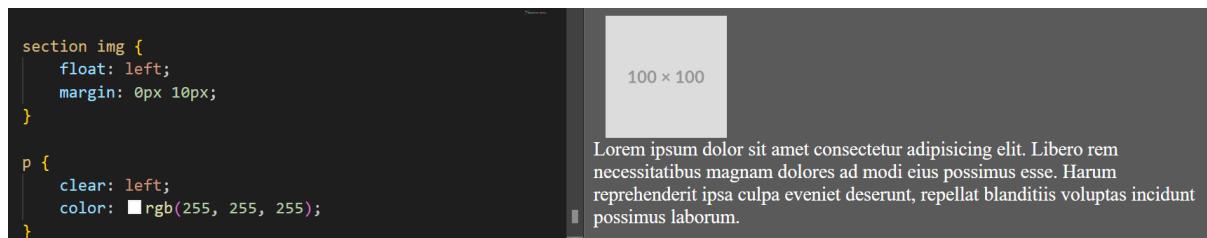
Usando el elemento **float** podemos hacer que, en este caso, la imagen flote a la izquierda permitiendo al texto ponerse a su derecha.



```
section img {
    float: left;
    margin: 0px 10px;
}
```

100 × 100  
Lorem ipsum dolor sit amet consectetur adipisicing elit. Libero rem necessitatibus magnam dolores ad modi eius possimus esse. Harum reprehenderit ipsa culpa eveniet deserunt, repellat blanditiis voluptas incidunt possimus laborum.

Con **clear** podemos hacer que el texto deje el espacio a la derecha de la imagen y se quede debajo pero permitiendo que la imagen se quede flotando.

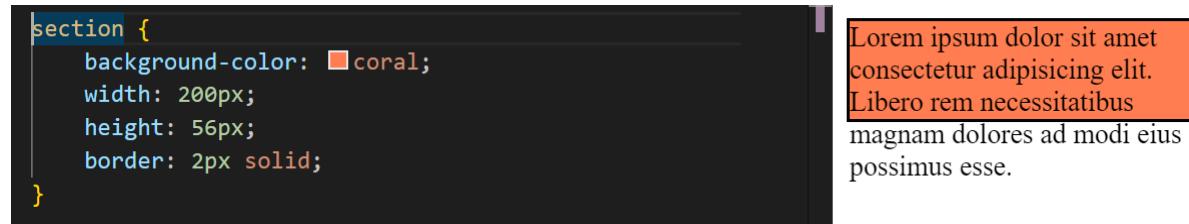


```
section img {
    float: left;
    margin: 0px 10px;
}

p {
    clear: left;
    color: #rgb(255, 255, 255);
```

100 × 100  
Lorem ipsum dolor sit amet consectetur adipisicing elit. Libero rem necessitatibus magnam dolores ad modi eius possimus esse. Harum reprehenderit ipsa culpa eveniet deserunt, repellat blanditiis voluptas incidunt possimus laborum.

Usando la propiedad **overflow** hacemos que el contenido no se desborde extendiendo el tamaño del contenedor o verlo con sliders.

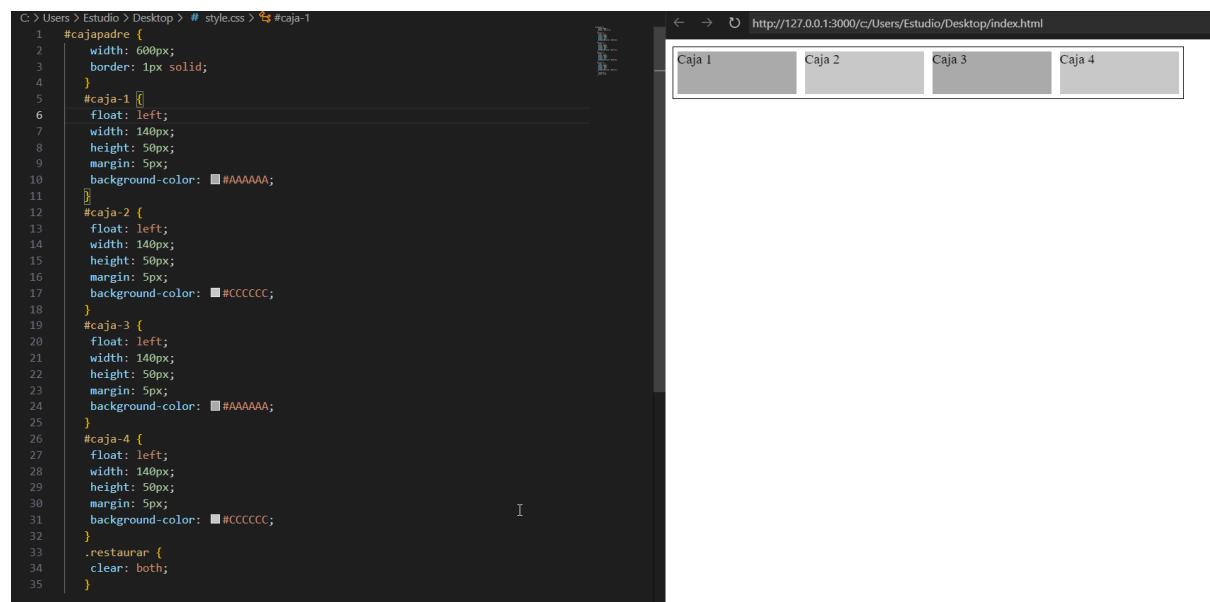


```
section {  
    background-color: #coral;  
    width: 200px;  
    height: 56px;  
    border: 2px solid;  
}  
  
Lorem ipsum dolor sit amet  
consectetur adipisicing elit.  
Libero rem necessitatibus  
magnam dolores ad modi eius  
possimus esse.
```

## Cajas Flotantes

Podemos ver en las siguientes imágenes, que usamos el elemento **float** para hacer que las cajas floten todas a la izquierda por lo que, si no tuvieramos **float**, las cajas se mostrarían una encima de otra.

### Con Float



```
C:\Users\Estudio\Desktop> # style.css > #caja-1  
1 #caja-padre {  
2     width: 600px;  
3     border: 1px solid;  
4 }  
5 #caja-1 {  
6     float: left;  
7     width: 140px;  
8     height: 50px;  
9     margin: 5px;  
10    background-color: #AAAAAA;  
11 }  
12 #caja-2 {  
13     float: left;  
14     width: 140px;  
15     height: 50px;  
16     margin: 5px;  
17     background-color: #CCCCCC;  
18 }  
19 #caja-3 {  
20     float: left;  
21     width: 140px;  
22     height: 50px;  
23     margin: 5px;  
24     background-color: #AAAAAA;  
25 }  
26 #caja-4 {  
27     float: left;  
28     width: 140px;  
29     height: 50px;  
30     margin: 5px;  
31     background-color: #CCCCCC;  
32 }  
33 .restaurar {  
34     clear: both;  
35 }
```

### Sin Float

```
C:\> Users > Estudio > Desktop > # style.css > ...
1  #cajapadre {
2  width: 600px;
3  border: 1px solid;
4  }
5  #caja-1 {
6  width: 140px;
7  height: 50px;
8  margin: 5px;
9  background-color: #AAAAAA;
}
10 #caja-2 {
11 width: 140px;
12 height: 50px;
13 margin: 5px;
14 background-color: #CCCCCC;
}
15 #caja-3 {
16 width: 140px;
17 height: 50px;
18 margin: 5px;
19 background-color: #AAAAAA;
}
20 #caja-4 {
21 width: 140px;
22 height: 50px;
23 margin: 5px;
24 background-color: #CCCCCC;
}
25 .restaurar {
26 clear: both;
}
27
28
29
30
31 }
```

## ¿Qué es **clear** ?

**Clear** "limpia" el espacio sobrante de los **float**, para continuar debajo de ellos.

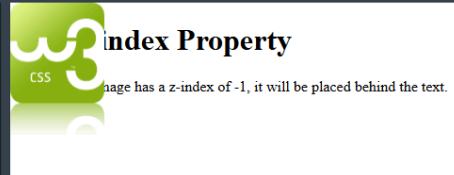
Cuando una caja contiene otras cajas dentro pero tiene la anchura limitada, las cajas una vez alcanzadas el límite de la caja padre, se pondrán debajo.

```
C:\> Users > Estudio > Desktop > # style.css > #caja-1
1  #cajapadre {
2  width: 500px;
3  border: 1px solid;
4  }
5  #caja-1 {
6  float: left;
7  width: 140px;
8  height: 50px;
9  margin: 5px;
10 background-color: #AAAAAA;
}
11 #caja-2 {
12 float: left;
13 width: 140px;
14 height: 50px;
15 margin: 5px;
16 background-color: #AAAAAA;
}
17 #caja-3 {
18 float: left;
19 width: 140px;
20 height: 50px;
21 margin: 5px;
22 background-color: #AAAAAA;
}
23 #caja-4 {
24 float: left;
25 width: 140px;
26 height: 50px;
27 margin: 5px;
28 background-color: #AAAAAA;
}
29 .restaurar {
30 clear: both;
}
31
32
33
34
35 }
```

La propiedad **z-index** se utiliza para mover elementos como si fueran capas, hacia delante o hacia detrás:

**z-index: 1**

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



**z-index: -1**

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



## Columnas

- **column-count** → Especificamos el número de columnas donde el navegador tiene que mostrar el contenido.
- **column-width** → El ancho de las columnas.
- **column-span** → Nos dice si los elementos se muestran en varias columnas o ninguna. Los valores son:
  - all
  - none
- **column-fill** → Rellena las columnas con estas dos propiedades:
  - auto → Se completan secuencialmente
  - balance → El contenido se divide en partes iguales en todas las columnas.
- **columns** → Podemos declarar tanto el número de columnas como el ancho de ellas.
- **column-gap** → Espaciado entre las columnas.
- **column-rule-style** → El tipo de línea que dividirá las columnas.
- **column-rule-color** → El color de la línea.

- **column-rule-width** → Es el ancho de la línea.
- **column-rule** → Nos permite definir todos los valores de rule anteriores en una misma línea.

## 4.3. Modelo de caja flexible

El objetivo de un modelo de caja es ofrecer un mecanismo con el que dividir el espacio de la ventana, crear filas y columnas. Sin embargo, lo que ofrece el modelo de caja tradicional no cumplen este objetivo.



**IMPORTANTE:** El modelo de caja flexible es mejor que el modelo de caja tradicional, pero se encuentra en fase experimental y algunos navegadores no pueden procesar sus propiedades.

### Contenedor flexible

En vez de hacer flotar los elementos, organiza las cajas usando contenedores flexibles. Este nuevo modelo, requiere que cada grupo de cajas estén dentro de una caja padre, que es la encargada de configurar sus características.

Para que un elemento dentro de un contenedor se vuelva flexible hay que declararlo:

**flex-grow** → Indica lo que va a expandir o encoger el elemento.

**flex-shrink** → Indica que el elemento se va a reducir.

**flex-basis** → Indica el tamaño inicial de un elemento.

**flex** → nos permite escribir flex-grow, flex-shrink y flex-basis en la misma línea.

La distribución del espacio depende de las propiedades del resto de las cajas. Si todas las cajas se configuran como flexibles, el tamaño de cada dependerá de la caja padre y el valor flex.



Cuando asignamos el valor 0 a flex-grow y flex-shrink, no deja que se expanda o reduzca.

## Organizando elementos flexibles.

Un elemento flexible se muestra horizontalmente por defecto en la misma línea, pero no se organiza con la orientación normal. Debemos organizarlo usando ejes.

- Eje principal → main-start, main-end. Es el eje del contenido.
- Eje transversal → cross-start, cross-end. Es el eje contrario.

**flex-direction** → Define el orden de orientación de las cajas.

- row
- row-reverse
- column
- column-reverse

**order** → Especifica el orden de las cajas. Acepta números enteros que determinan la ubicación de cada caja.

**justify-content** → Indica la distribución del espacio libre.

- flex-start
- flex-end
- center
- space-between
- space-around

**align-items** → Alinea las cajas del eje transversal.

- flex-start
- flex-end
- center
- baseline

- stretch

**aling-self** → Esta propiedad alinea una caja del eje transversal.

- auto
- flex-start
- flex-end
- center
- baseline
- stretch

**flex-wrap** → Múltiples líneas de cajas.

- nowrap
- wrap
- wrap-reverse

**aling-content** → Alinea las cajas en el eje vertical.

- flex-start
- flex-end
- center
- space-between
- space-around
- stretch



CSS ofrece una propiedad llamada **writing-mode** que determina la orientación de las líneas de texto.