

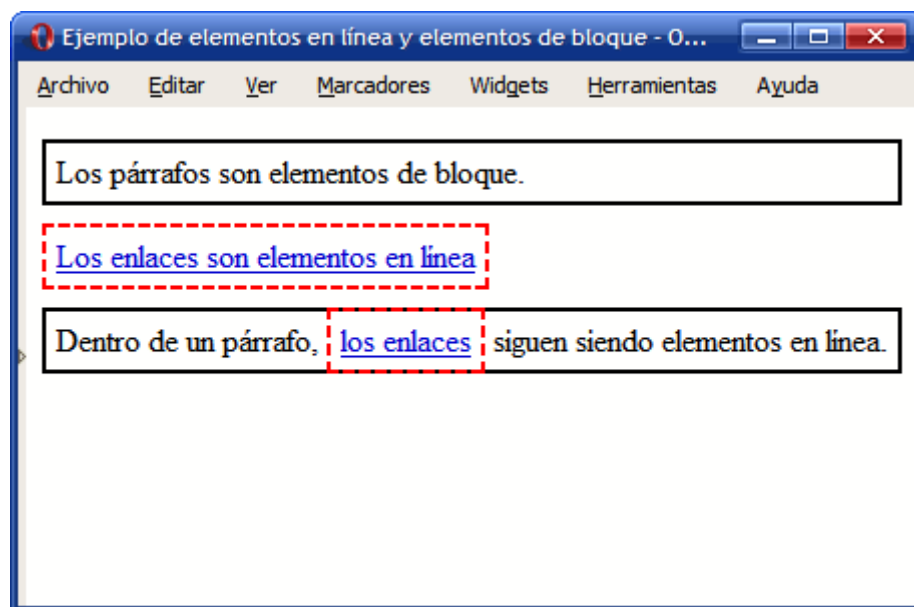
Tema 6. Posicionamiento y visualización

1. Tipos de elementos

El estándar HTML clasifica a todos sus elementos en dos grandes grupos: elementos en línea y elementos de bloque.

Los elementos de bloque siempre empiezan en una nueva línea y ocupan todo el espacio disponible hasta el final de la línea. Por su parte, los elementos en línea no empiezan necesariamente en nueva línea y sólo ocupan el espacio necesario para mostrar sus contenidos.

La siguiente imagen muestra las cajas que crea el navegador para representar los diferentes elementos que forman una página HTML:



El primer elemento de la página anterior es un párrafo. Los párrafos son elementos de bloque y por ese motivo su caja empieza en una nueva línea y llega hasta el final de esa misma línea.

El segundo elemento de la página es un enlace. Los enlaces son elementos en línea, por lo que su caja sólo ocupa el espacio necesario para mostrar sus contenidos.

El tercer elemento de la página es un párrafo que se comporta de la misma forma que el primer párrafo. En su interior, se encuentra un enlace que también se comporta de la misma forma que el enlace anterior.

Los elementos de bloque no pueden insertarse dentro de elementos en línea y tan sólo pueden aparecer dentro de otros elementos de bloque. En cambio, un elemento en línea puede aparecer tanto dentro de un elemento de bloque como dentro de otro elemento en línea.

Los elementos en línea definidos por HTML son: a , abbr , acronym , b , basefont , bdo , big , br , cite , code , dfn , em , font , i , img , input , kbd , label , q , s , samp , select , small , span , strike , strong , sub , sup , textarea , tt , u , var .

Los elementos de bloque definidos por HTML son: address , blockquote , center , dir , div , dl , fieldset , form , h1 , h2 , h3 , h4 , h5 , h6 , hr , isindex , menu , noframes , noscript , ol , p , pre , table , ul .

2. Posicionamiento

Los navegadores crean y posicionan de forma automática todas las cajas que forman cada página HTML. No obstante, CSS permite al diseñador modificar la posición en la que se muestra cada caja.

El estándar de CSS define cinco modelos diferentes para posicionar una caja:

- Posicionamiento normal o estático: se trata del posicionamiento que utilizan los navegadores si no se indica lo contrario.
- Posicionamiento relativo: variante del posicionamiento normal que consiste en posicionar una caja según el posicionamiento normal y después desplazarla respecto de su posición original.
- Posicionamiento absoluto: la posición de una caja se establece de forma absoluta respecto de su elemento contenedor y el resto de elementos de la página ignoran la nueva posición del elemento.
- Posicionamiento fijo: variante del posicionamiento absoluto que convierte una caja en un elemento inamovible, de forma que su posición en la pantalla siempre es la misma independientemente del resto de elementos e independientemente de si el usuario sube o baja la página en la ventana del navegador.
- Posicionamiento flotante: se trata del modelo más especial de posicionamiento, ya que desplaza las cajas todo lo posible hacia la izquierda o hacia la derecha de la línea en la que se encuentran.

El posicionamiento de una caja se establece mediante la propiedad **position** y los valores que puede tomar son: static | relative | absolute | fixed

- static : corresponde al posicionamiento normal o estático.
- relative : corresponde al posicionamiento relativo. El desplazamiento de la caja se controla con las propiedades top , right , bottom y left .
- absolute : corresponde al posicionamiento absoluto. El desplazamiento de la caja también se controla con las propiedades top , right , bottom y left , pero su interpretación es mucho más compleja, ya que el origen de coordenadas del desplazamiento depende del posicionamiento de su elemento contenedor.
- fixed : corresponde al posicionamiento fijo. El desplazamiento se establece de la misma forma que en el posicionamiento absoluto, pero en este caso el elemento permanece inamovible en la pantalla.

La propiedad **position** sólo indica cómo se posiciona una caja, pero no la desplaza, para desplazarla se utilizan las propiedades **top , right , bottom y left** que toman como valor una medida o un porcentaje.

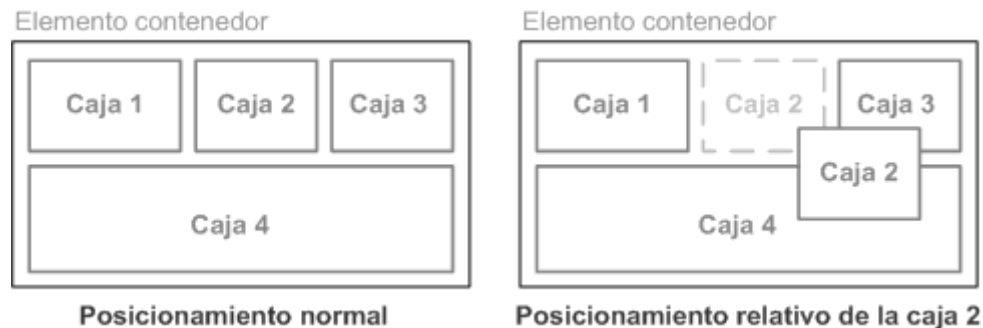
En el caso del posicionamiento relativo, cada una de estas propiedades indica el desplazamiento del elemento desde la posición original de su borde superior/derecho/inferior/izquierdo. Si el posicionamiento es absoluto, las propiedades indican el desplazamiento del elemento respecto del borde superior/derecho/inferior/izquierdo de su primer elemento padre posicionado.

3. Posicionamiento normal

El posicionamiento normal o estático es el modelo que utilizan por defecto los navegadores para mostrar los elementos de las páginas. En este modelo, ninguna caja se desplaza respecto de su posición original, por lo que sólo se tiene en cuenta si el elemento es de bloque o en línea.

4. Posicionamiento relativo

El desplazamiento de una caja no afecta al resto de cajas adyacentes, que se muestran en la misma posición que si la caja desplazada no se hubiera movido de su posición original.



En la imagen anterior, la caja 2 se ha desplazado lateralmente hacia la derecha y verticalmente de forma descendente. Como el resto de cajas de la página no modifican su posición, se producen solapamientos entre los contenidos de las cajas.

El siguiente ejemplo muestra tres imágenes posicionadas de forma normal:



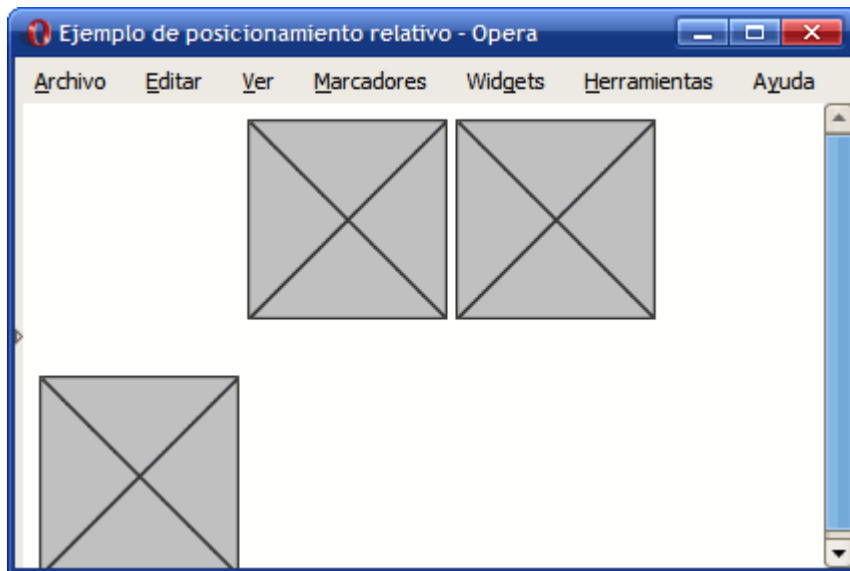
Aplicando el posicionamiento relativo, se desplaza la primera imagen de forma descendente:

```
img.desplazada
{ position: relative;
  top: 8em;
}



```

El aspecto que muestran ahora las imágenes es el siguiente:



El resto de imágenes no varían su posición y por tanto no ocupan el hueco dejado por la primera imagen, ya que el posicionamiento relativo no influye en el resto de elementos de la página. El único problema de posicionar elementos de forma relativa es que se pueden producir solapamientos con otros elementos de la página.

5. Posicionamiento absoluto

El desplazamiento en este caso dependen del posicionamiento del elemento contenedor.

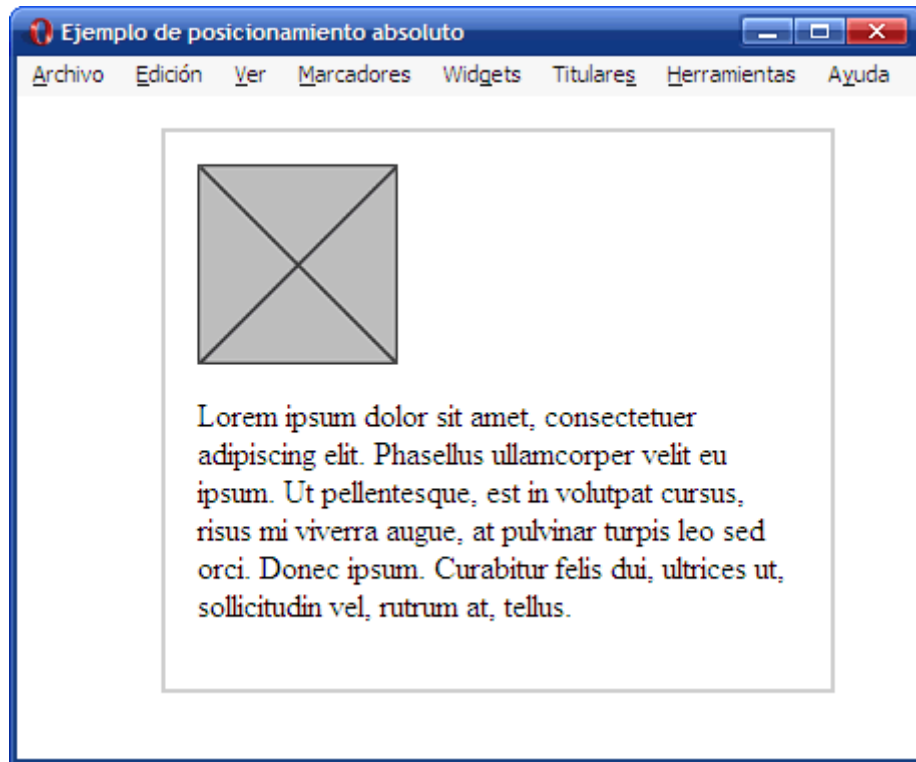
Cuando una caja se posiciona de forma absoluta, el resto de elementos de la página la ignoran y ocupan el lugar original ocupado por la caja posicionada. Al igual que en el posicionamiento relativo, cuando se posiciona de forma absoluta una caja es probable que se produzcan solapamientos con otras cajas.

En el siguiente ejemplo, se posiciona de forma absoluta la caja 2:



La caja 2 está posicionada de forma absoluta, lo que implica que el resto de elementos ignoran que esa caja exista. Por este motivo, la caja 3 deja su lugar original y pasa a ocupar el hueco dejado por la caja 2.

Partiendo de la siguiente imagen con posicionamiento normal:



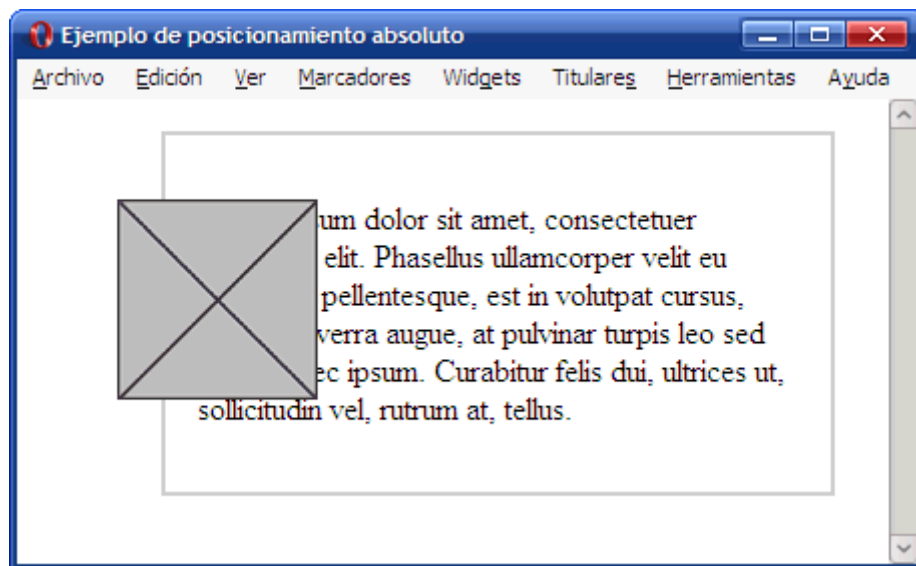
El código HTML y CSS de la página original es:

```
div {  
  border: 2px solid #CCC;  
  padding: 1em;  
  margin: 1em 0 1em 4em;  
  width: 300px;  
}  
  
<div>  
    
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus  
    ullamcorper velit eu ipsum. Ut pellentesque, est in volutpat cursus, risus  
    mi viverra augue, at pulvinar turpis leo sed orci. Donec ipsum. Curabitur  
    felis dui, ultrices ut, sollicitudin vel, rutrum at, tellus.</p>  
</div>
```

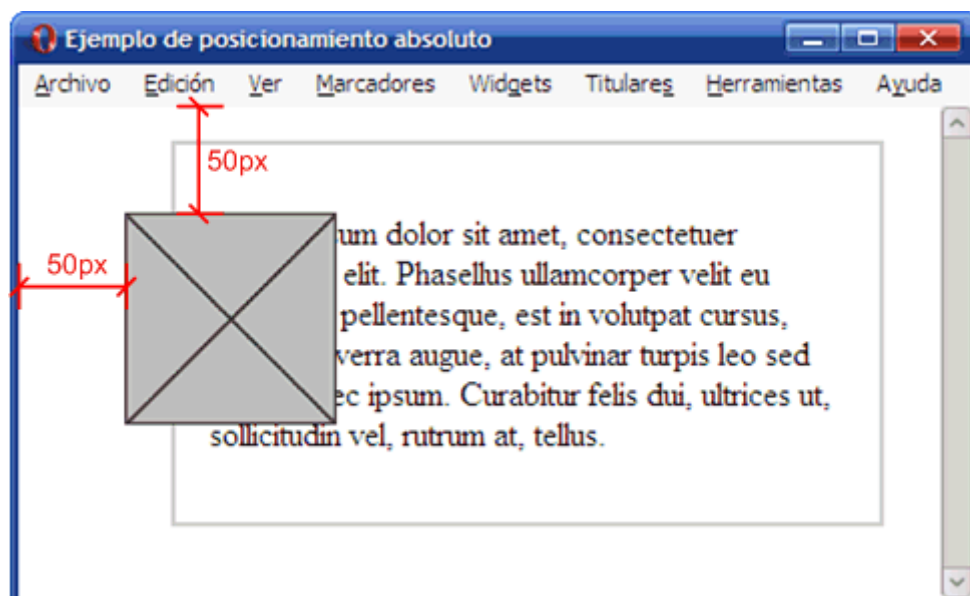
Se posiciona de forma absoluta la imagen mediante la propiedad position y se indica su nueva posición mediante las propiedades top y left :

```
div img {
    position: absolute;
    top: 50px;
    left: 50px;
}
```

El resultado visual se muestra en la siguiente imagen:



La imagen posicionada de forma absoluta no toma como origen de coordenadas la esquina superior izquierda de su elemento contenedor <div> , sino que su referencia es la esquina superior izquierda de la página:



6.

Posicionamiento fijo

Cuando una caja se posiciona de forma fija, la forma de obtener el origen de coordenadas para interpretar su desplazamiento es idéntica al posicionamiento absoluto. De hecho, si el usuario no mueve la página HTML en la ventana del navegador, no existe ninguna diferencia entre estos dos modelos de posicionamiento.

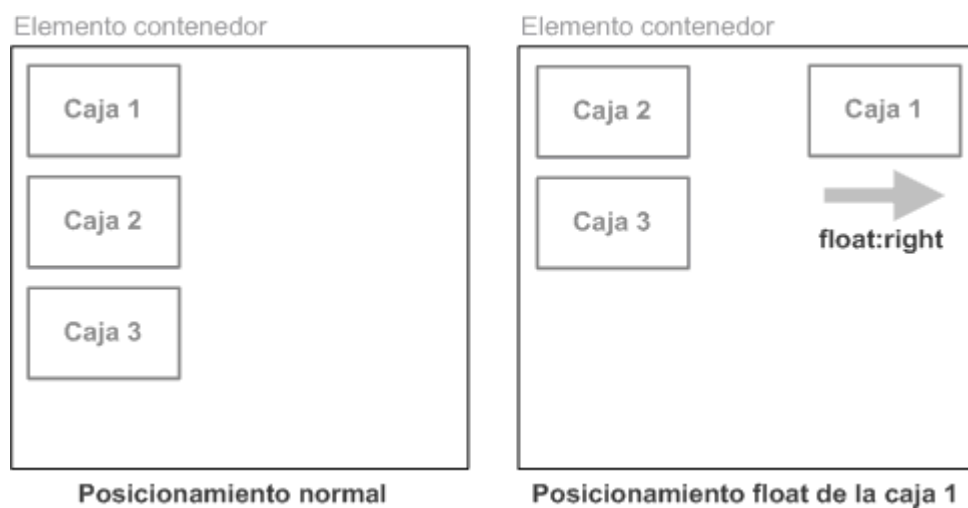
La principal característica de una caja posicionada de forma fija es que su posición es inamovible dentro de la ventana del navegador. El posicionamiento fijo hace que las cajas no modifiquen su posición ni aunque el usuario suba o baje la página en la ventana de su navegador.

7. Posicionamiento flotante

El posicionamiento flotante es el más utilizado.

Cuando una caja se posiciona con el modelo de posicionamiento flotante, automáticamente se convierte en una caja flotante, lo que significa que se desplaza hasta la zona más a la izquierda o más a la derecha de la posición en la que originalmente se encontraba.

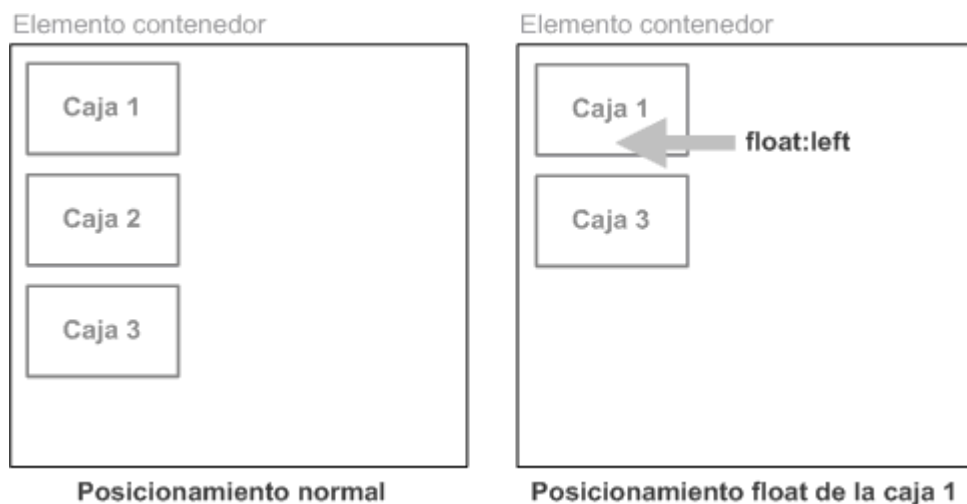
La siguiente imagen muestra el resultado de posicionar de forma flotante hacia la derecha la caja 1:



Cuando se posiciona una caja de forma flotante:

- La caja deja de pertenecer al flujo normal de la página, lo que significa que el resto de cajas ocupan el lugar dejado por la caja flotante.
- La caja flotante se posiciona lo más a la izquierda o lo más a la derecha posible de la posición en la que se encontraba originalmente.
- No existe solapamiento de cajas

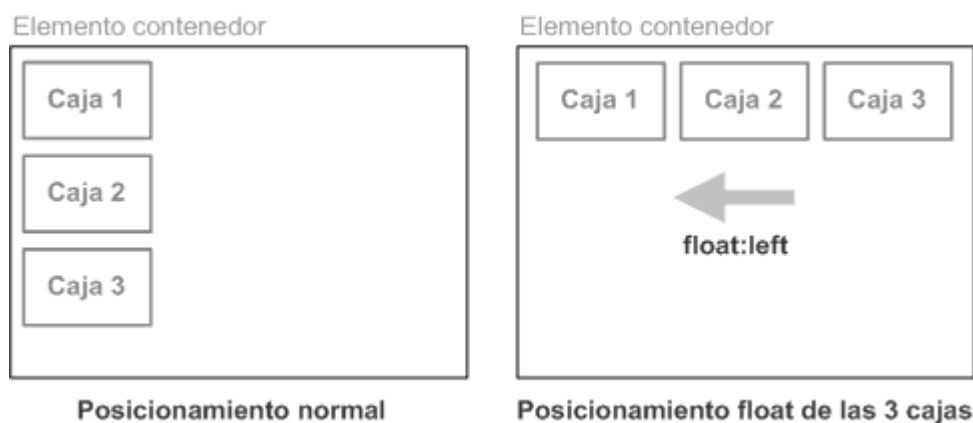
Si en el anterior ejemplo la caja 1 se posiciona de forma flotante hacia la izquierda, el resultado es el que muestra la siguiente imagen:



La caja 1 es de tipo flotante, por lo que desaparece del flujo normal de la página y el resto de cajas ocupan su lugar. El resultado es que la caja 2 ahora se muestra donde estaba la caja 1 y la caja 3 se muestra donde estaba la caja 2.

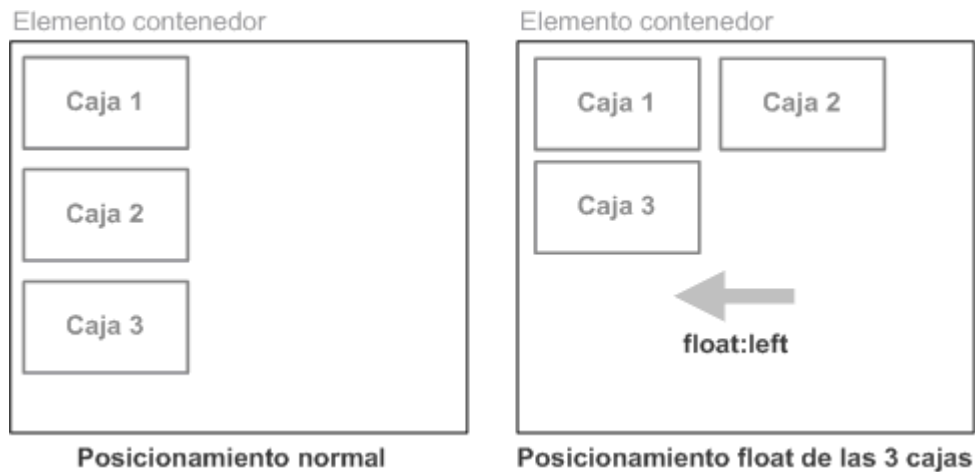
Al mismo tiempo, la caja 1 se desplaza todo lo posible hacia la izquierda de la posición en la que se encontraba. El resultado es que la caja 1 se muestra encima de la nueva posición de la caja 2 y tapa todos sus contenidos.

Si existen otras cajas flotantes, al posicionar de forma flotante otra caja, se tiene en cuenta el sitio disponible. En el siguiente ejemplo se posicionan de forma flotante hacia la izquierda las tres cajas:



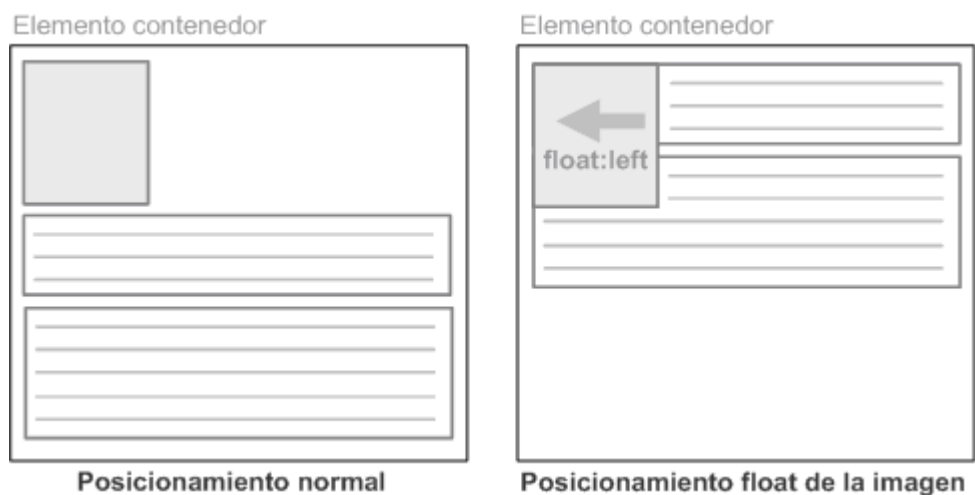
En el ejemplo anterior, las cajas no se superponen entre sí porque las cajas flotantes tienen en cuenta las otras cajas flotantes existentes. Como la caja 1 ya estaba posicionada lo más a la izquierda posible, la caja 2 sólo puede colocarse al lado del borde derecho de la caja 1, que es el sitio más a la izquierda posible respecto de la zona en la que se encontraba.

Si no existe sitio en la línea actual, la caja flotante baja a la línea inferior hasta que encuentra el sitio necesario para mostrarse lo más a la izquierda o lo más a la derecha posible en esa nueva línea:

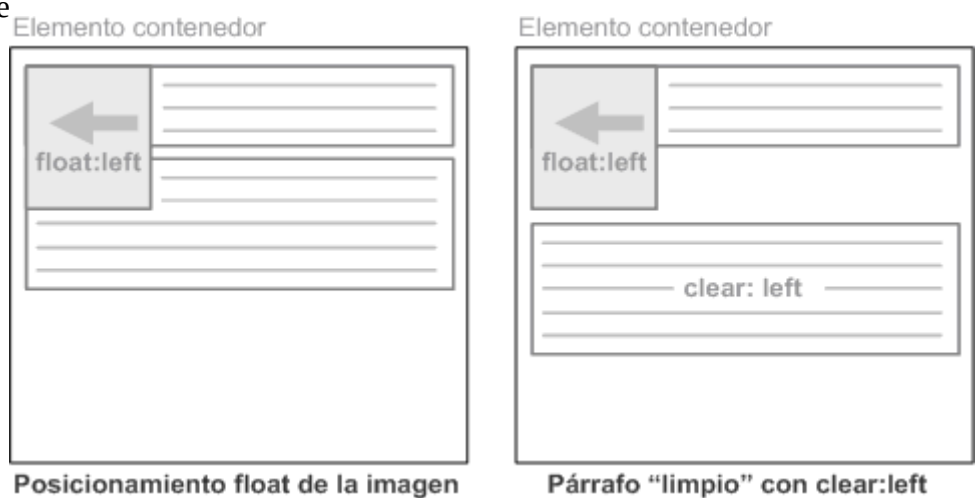


La propiedad CSS que permite posicionar de forma flotante una caja se denomina **float** y los valores que puede tomar son **left** y **right**.

Los elementos que se encuentran alrededor de una caja flotante adaptan sus contenidos para que fluyan alrededor del elemento posicionado:



CSS permite controlar la forma en la que los contenidos fluyen alrededor de los contenidos posicionados mediante float . De hecho, en muchas ocasiones es admisible que algunos contenidos fluyan alrededor de una imagen, pero el resto de contenidos deben mostrarse en su totalidad sin fluir alrededor de la imagen:



La propiedad `clear` permite modificar el comportamiento por defecto del posicionamiento flotante para forzar a un elemento a mostrarse debajo de cualquier caja flotante. Esta propiedad puede tener los valores `left` | `right` | `both`.

La propiedad `clear` indica el lado del elemento HTML que no debe ser adyacente a ninguna caja posicionada de forma flotante. Si se indica el valor `left`, el elemento se desplaza de forma descendente hasta que pueda colocarse en una línea en la que no haya ninguna caja flotante en el lado izquierdo.

Si se indica el valor `right`, el comportamiento es análogo, salvo que en este caso se tienen en cuenta los elementos desplazados hacia la derecha.

El valor `both` despeja los lados izquierdo y derecho del elemento, ya que desplaza el elemento de forma descendente hasta que el borde superior se encuentre por debajo del borde inferior de cualquier elemento flotante hacia la izquierda o hacia la derecha.

8. Visualización

CSS define otras cuatro propiedades para controlar su visualización: `display`, `visibility`, `overflow` y `z-index`.

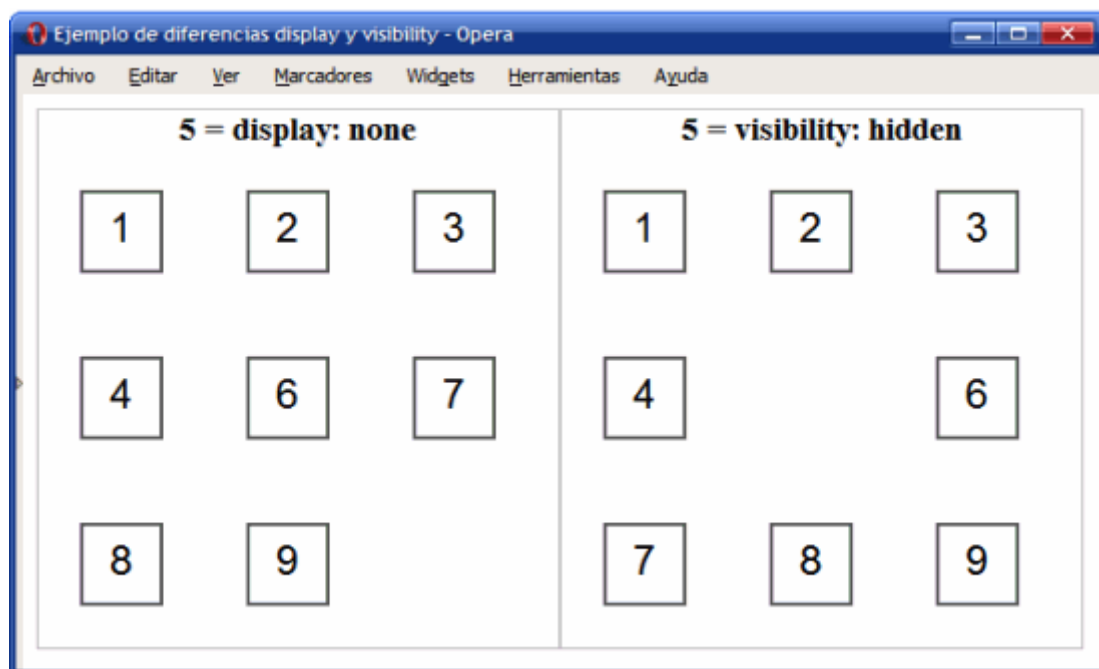
8.1. Propiedades `display` y `visibility`

Las propiedades `display` y `visibility` controlan la visualización de los elementos. Las dos propiedades permiten ocultar cualquier elemento de la página.

La propiedad `display` permite ocultar completamente un elemento haciendo que desaparezca de la página. Como el elemento oculto no se muestra, el resto de elementos de la página se mueven para ocupar su lugar.

La propiedad `visibility` permite hacer invisible un elemento, lo que significa que el navegador crea la caja del elemento pero no la muestra. En este caso, el resto de elementos de la página no modifican su posición, ya que aunque la caja no se ve, sigue ocupando sitio.

La siguiente imagen muestra la diferencia entre ocultar la caja número 5 mediante la propiedad `display` o hacerla invisible mediante la propiedad `visibility`:



En general, cuando se oculta un elemento no es deseable que siga ocupando sitio en la página, por lo que la propiedad **display** se utiliza mucho más que la propiedad **visibility**.

A continuación se muestra una tabla con los valores que se pueden utilizar con la propiedad **display** (algunos de ellos los veremos con mas detalle en otros temas)

Valor	Características
block	Se apila en vertical. Ocupa todo el ancho disponible de su etiqueta contenedora.
Versión en línea	
inline	Se coloca en horizontal. Se adapta al ancho de su contenido. Ignora width o height .
inline-block	Combinación de los dos anteriores. Se comporta como inline pero no ignora width o height .
Versión flexible	
flex	Utiliza el modelo de cajas flexibles de CSS. Ideal para estructuras de 1 dimensión.
inline-flex	Versión en línea (ocupa sólo su contenido) del modelo de cajas flexibles de CSS.
Versión de cuadrículas	
grid	Utiliza cuadrículas o rejillas con el modelo de cajas Grid CSS.
inline-grid	La versión en línea (ocupa sólo su contenido) del modelo de cajas grid css.
Versión de listas y tablas	
list-item	Actúa como un ítem de una lista. Es el comportamiento de etiquetas como .
table	Actúa como una tabla. Es el comportamiento de etiquetas como <table> .
table-cell	Actúa como la celda de una tabla. Es el comportamiento de etiquetas como <th> o <td> .
table-row	Actúa como la fila de una tabla. Es el comportamiento de etiquetas como <tr> .
Otros	
contents	Ignora la caja del elemento. Útil para mantener Grid/Flex aún teniendo un wrapper intermedio.

Valor	Características
none	Oculto el elemento visualmente, como si no existiera en el HTML.

La propiedad visibility solo admite los valores visible y hidden, ya que sólo permite hacer visibles o invisibles a los elementos de la página.

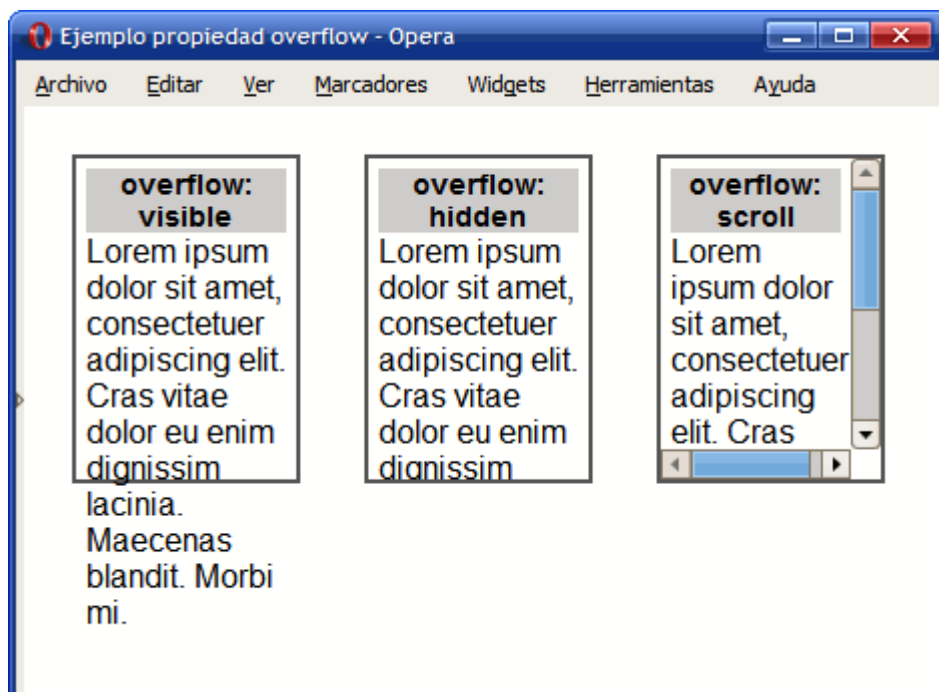
8.2. Propiedad overflow

En algunas ocasiones el contenido de un elemento no cabe en el espacio reservado para ese elemento y se desborda. La situación más habitual en la que el contenido sobresale de su espacio reservado es cuando se establece la anchura y/o altura de un elemento mediante la propiedad width y/o height.

CSS define la propiedad overflow para controlar la forma en la que se visualizan los contenidos que sobresalen de sus elementos. Los valores que puede tomar son: visible | hidden | scroll | auto

- **visible** : el contenido no se corta y se muestra sobresaliendo la zona reservada para visualizar el elemento. Este es el comportamiento por defecto.
- **hidden** : el contenido sobrante se oculta y sólo se visualiza la parte del contenido que cabe dentro de la zona reservada para el elemento.
- **scroll** : solamente se visualiza el contenido que cabe dentro de la zona reservada para el elemento, pero también se muestran barras de scroll tanto horizontales como verticales que permiten visualizar el resto del contenido.
- **auto** : el comportamiento depende del navegador, aunque normalmente es el mismo que la propiedad scroll . Colocando las barras de desplazamiento necesarias.

La siguiente imagen muestra un ejemplo de los tres valores típicos de la propiedad overflow :



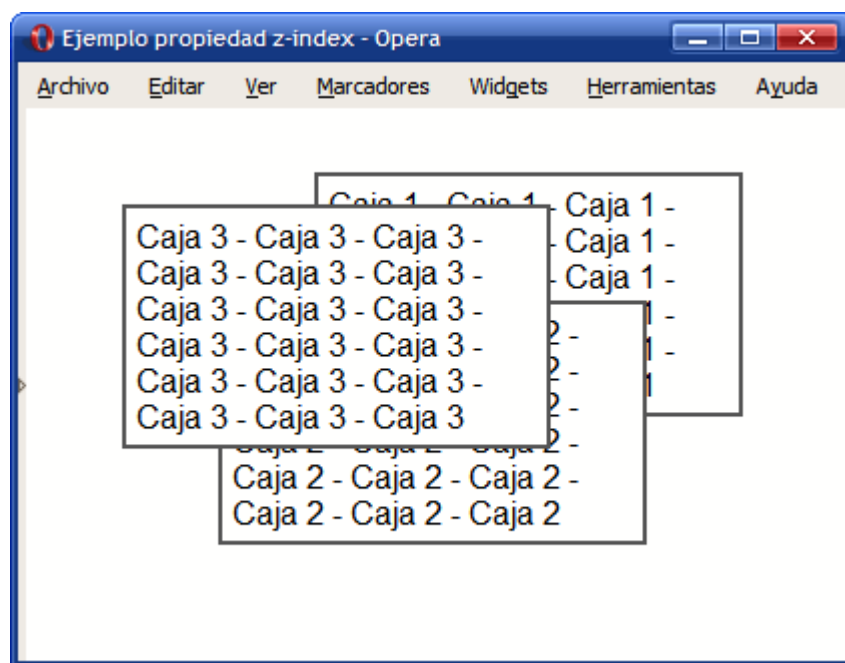
8.3. Propiedad z-index

CSS permite controlar la posición tridimensional de las cajas posicionadas. De esta forma, es posible indicar las cajas que se muestran delante o detrás de otras cajas cuando se producen solapamientos.

La posición tridimensional de un elemento se controla mediante la propiedad z-index que tomara como valor un número.

Cuanto más alto sea el valor numérico, más cerca del usuario se muestra la caja. Un elemento con z-index: 10 se muestra por encima de los elementos con z-index: 8 o z-index: 9 , pero por debajo de elementos con z-index: 20 o z-index: 50.

La siguiente imagen muestra un ejemplo de uso de la propiedad z-index :



La propiedad z-index sólo tiene efecto en los elementos posicionados, por lo que es obligatorio que la propiedad z-index vaya acompañada de la propiedad position.

9. Flex (flexbox)

Lo anterior cada vez se usa menos, z-index hay que usarlo siempre en posición absoluta.

Flex (también llamado flexbox) es un sistema de elementos flexibles que llega con la idea de olvidar estos mecanismos y acostumbrarnos a una mecánica más potente, limpia y personalizable, en la que los elementos HTML se adaptan y colocan automáticamente y es más fácil personalizar los diseños de una página web.

Para empezar a utilizar flex lo primero que debemos hacer es conocer algunos de los elementos básicos de este nuevo esquema, que son los siguientes:



- ❑ **Contenedor:** Es el elemento padre que tendrá en su interior cada uno de los ítems flexibles. Observa que al contrario que muchas otras estructuras CSS, por norma general, en Flex establecemos las propiedades al elemento padre.
- ❑ **Eje principal:** Los contenedores flexibles tendrán una orientación principal específica. Por defecto, el eje principal del contenedor flex es en horizontal (en fila).
- ❑ **Eje secundario:** De la misma forma, los contenedores flexibles tendrán una orientación secundaria, perpendicular a la principal. Si la principal es en horizontal, la secundaria será en vertical (y viceversa).
- ❑ **Ítem:** Cada uno de los hijos que tendrá el contenedor en su interior.

9.1 Modalidades de flex

Para activar el modo flex, utilizaremos sobre el elemento contenedor la propiedad display, y especificaremos el valor flex o inline-flex (dependiendo de como queramos que se comporte el contenedor):

Tipo de elemento	Descripción
inline-flex	Establece un contenedor en línea, similar a inline-block (ocupa solo el contenido).
flex	Establece un contenedor en bloque, similar a block (ocupa todo el ancho del padre).

Por defecto, y sólo con esto, observaremos que los elementos se disponen todos sobre una misma línea. Esto ocurre porque estamos utilizando el modo flex y estaremos trabajando con ítems flexibles básicos, garantizando que no se desborden ni mostrarán los problemas que, por ejemplo, tienen los porcentajes sobre elementos que no utilizan flex.

Sobre el siguiente código si aplicamos una de la modalidad flex a la clase container y a los ítems obtendremos los ítems de la siguiente forma:

```

<div class="container"> <!-- Flex container -->
  <div class="item item-1">1</div> <!-- Flex items -->
  <div class="item item-2">2</div>
  <div class="item item-3">3</div>
</div>

```



Si en lugar de aplicar flex aplicamos inline-flex, el resultado sería el siguiente:







9.2 Dirección de los ejes

Con las propiedad *flex-direction* podemos modificar la dirección del eje principal del contenedor para que se oriente en horizontal (valor por defecto) o en vertical. Además, también podemos incluir el sufijo *-reverse* para indicar que coloque los ítems en orden inverso.

Propiedad	Valor	Significado
flex-direction	row row-reverse column column-reverse	Cambia la orientación del eje principal.

Valor	Descripción
row	Establece la dirección del eje principal en horizontal.
row-reverse	Establece la dirección del eje principal en horizontal invertido.
column	Establece la dirección del eje principal en vertical.
column-reverse	Establece la dirección del eje principal en vertical invertido.

Resultado de aplicar los diferentes valores a flex-direction:

valor	Resultado
row	
row-reverse	
column	
column-reverse	

9.3 Contenedor flex multilínea

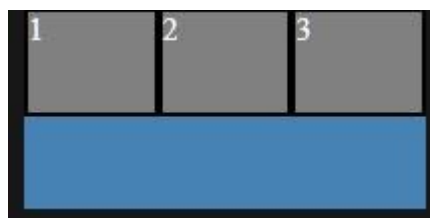
Por defecto, si un elemento no cabe dentro de nuestro contenedor flex, los elementos se harán más pequeños (son flexibles) para ajustarlos al contenedor. Este es el comportamiento por defecto de un contenedor flex. Sin embargo, con la propiedad flex-wrap podemos cambiar este comportamiento y permitir que nuestro contenedor flex se desborde, convirtiéndose en un contenedor flex multilínea.

Propiedad	Valor	Significado
flex-wrap	nowrap wrap wrap-reverse	Evita o permite el desbordamiento (multilínea).

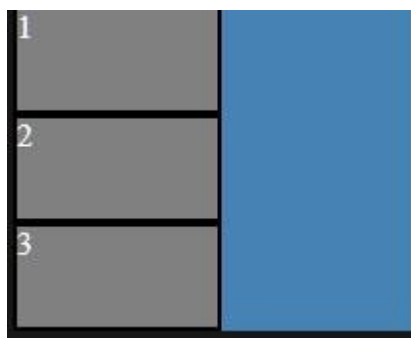
Los valores que puede tomar esta propiedad, son las siguientes:

Valor	Descripción
nowrap	Los ítems se ajustan para ocupar el tamaño del contenedor (no permite desbordamiento en múltiples líneas).
wrap	Establece los ítems en modo multilínea (permite que se desborde el contenedor).
wrap-reverse	Establece los ítems en modo multilínea, pero en dirección inversa.

Si especificamos la anchura del contenedor en 200px y establecemos la anchura de cada ítem (suponiendo que tenemos 3) en 100px, no caben en el contenedor y flex reajusta los ítems para que quepan en el contenedor.



Si especificamos *wrap* en la propiedad *flex-wrap*, lo que permitimos es que el contenedor se pueda desbordar, pasando a ser un contenedor multilínea, que mostraría los ítems que quepan en la primera línea y el resto en las líneas siguientes. En este caso cada ítem tendría un ancho de 100px.



9.4 Huecos

Existen dos propiedades de flexbox que han surgido recientemente: *row-gap* y *column-gap*. Dichas propiedades, permiten establecer el tamaño de un «hueco» entre ítems desde el elemento padre contenedor, y que eliminan la necesidad de estar utilizando *padding* o *margin* en los elementos hijos.

Propiedad	Valor	Descripción
<i>row-gap</i>	Normal <medida>	Espacio entre filas (sólo funciona con <i>flex-direction: column</i>)
<i>column-gap</i>	Normal <medida>	Espacio entre columnas (sólo funciona con <i>flex-direction: row</i>)

Como flex es un sistema para diseños de una sola dimensión, sólo una de las dos propiedades tendrá efecto. Si la propiedad flex-direction está establecida en column, podrás utilizar row-gap, y en el caso de que la propiedad flex-direction se encuentre en row, podrás utilizar el column-gap.

Eso sí, es posible usar ambas si tenemos la propiedad flex-wrap definida a wrap y, por lo tanto, disponemos de multicolumnas flexbox, ya que en este caso si podemos separar elementos por filas y por columnas.

Los huecos sólo se aplican entre elementos, y no entre un elemento hijo y su contenedor padre.

Podemos utilizar la propiedad atajo para huecos gap. Si se indica un valor se aplicara ese valor a ambos ejes y si se indican dos valores, el primero se aplicara al eje X y el segundo al eje Y.

Por ejemplo:






```
.container {  
  /* 2 parámetros: <row> <column> */  
  gap: 4px 8px;  
  /* Equivalente a */  
  row-gap: 4px;  
  column-gap: 8px;  
  
  /* 1 parámetro: usa el mismo para ambos */  
  gap: 4px;  
  /* Equivalente a */  
  row-gap: 4px;  
  column-gap: 4px;  
}
```

9.5 Alineación con flex


Justify-content: **Start | end | center | space-between | space-around | space-evenly**

Se utiliza para alinear los ítems del eje principal

Los valores que puede tomar esta propiedad son los siguientes:

Valor	Descripción
start	Agrupar los ítems al inicio del eje principal. 
end	Agrupar los ítems al final del eje principal. 
center	Agrupar los ítems al centro del eje principal. 
space-between	Distribuye los ítems dejando espacio entre ellos. 
space-around	Distribuye los ítems dejando espacio alrededor de ellos. 

Valor	Descripción
space-evenly	Distribuye como space-around, pero con un espacio exactamente igual alrededor de ellos.



Align-items: **start | end | center | stretch | baseline**

Se encarga de alinear los ítems en el eje secundario del contenedor flex

Los valores que puede tomar align-items son los siguientes:

Valor	Descripción
start	Alinea los ítems al inicio del eje secundario.
end	Alinea los ítems al final del eje secundario.
center	Alinea los ítems al centro del eje secundario.
stretch	Alinea los ítems estirándolos de modo que cubran desde el inicio hasta el final del contenedor.
baseline	Alinea los ítems en el contenedor según la base del contenido de los ítems del contenedor.

Align-content: **start | end | center | space-between | space-around | space-evenly | stretch**

Es similar al anterior pero se utiliza cuando tenemos un contenedor flex-wrap.

Los valores que puede tomar la propiedad align-content son los siguientes:

Valor	Descripción
start	Agrupar los ítems al inicio del eje principal.
end	Agrupar los ítems al final del eje principal.
center	Agrupar los ítems al centro del eje principal.
space-between	Distribuye los ítems desde el inicio hasta el final.
space-around	Distribuye los ítems dejando el mismo espacio a los lados de cada uno.
stretch	Estira los ítems para ocupar de forma equitativa todo el espacio.

Align-self: **auto | start | end | center | stretch | baseline**

Actúa exactamente igual que align-items pero en lugar de aplicar la propiedad a todos los items se lo aplica a un ítem en específico. (le añadiremos una clase a ese ítem)

La propiedad align-self puede tomar los siguientes valores:

Valor	Descripción
start	Alinea los ítems al inicio del contenedor.
end	Alinea los ítems al final del contenedor.
center	Alinea los ítems al centro del contenedor.
stretch	Alinea los ítems estirándolos al tamaño del contenedor.
baseline	Alinea los ítems en el contenedor según la base de los ítems.
auto	Hereda el valor de align-items del padre (si no se ha definido, es stretch).

9.6 Orden de los elementos

Order: 0 | <medida> Número que indica el orden de aparición de los ítems.

Por defecto, todos los elementos hijos de un contenedor flex tienen establecido un orden por defecto al valor 0. Si indicamos una propiedad order con un valor numérico diferente, irá recolocando los ítems según dicho número, colocando antes los elementos con un número order más pequeño (incluso valores negativos) y después los elementos con números más altos.

9.7 Propiedades de flexibilidad

Hasta ahora, excepto align-flex, todas las propiedades actúan sobre el contenedor. Las siguientes actúan sobre los elementos hijos.

Flex-basis: <medida> |content **Tamaño base de los ítems antes de aplicar una variación.**

Define el tamaño base por defecto que tendrán los ítems (es el width). Se suele aplicar un tamaño específico (unidades, porcentajes, etc...), pero también se puede aplicar la palabra clave content que ajusta automáticamente el tamaño al contenido del elemento. Este es el valor por defecto de la propiedad.

Si al contenedor le aplicamos la propiedad flex-direction: column, esta propiedad actuará como height.

Flex-grow: 0 | <número> **Número que indica el factor de crecimiento del ítem (si no ocupa el 100% del padre).**

La propiedad flex-grow actúa en situaciones donde:

- ☐ Hay un flex-basis definido.
- ☐ Los ítems cubren el tamaño total del contenedor flex padre.

En esas situaciones, la propiedad flex-grow indica el factor de crecimiento de los ítems en el caso de que no tengan un ancho o alto específico

Por ejemplo, si tenemos el siguiente código html:

```
<div class="container">
  <div class="item item1">Elemento 1</div>
  <div class="item item2">Elemento 2</div>
  <div class="item">Elemento 3</div>
</div>
```

y el siguiente CSS:

```
.container {
  display: flex;
  justify-content: center;
  align-items: center;
  background: grey;
  height: 150px;
  gap: 20px;
}
```

```
.item {
  flex-basis: 200px;
  background: red;
}
```

```
.item1 {flex-grow: 1;}
```

Todos los elementos tienen un `flex-grow: 0` por defecto, de modo que no hay factor de crecimiento en el elemento, y tendrá el tamaño definido por la propiedad `flex-basis`. Sin embargo, si colocamos un `flex-grow: 1` al primer hijo (`item1`), este crecerá hasta que la suma de los hijos ocupen el 100% del contenedor, mientras que el resto de hijos tendrá el tamaño base definido por `flex-basis`.

Si al código CSS anterior le añadimos

```
.item2 {flex-grow: 2;}
```

El primer elemento tiene un factor de crecimiento de 1, mientras el segundo elemento tiene un factor de crecimiento de 2, el resto de elementos tendrán un factor de crecimiento de 0, o sea, no crecen. Así pues, tendremos que el elemento número 3 tiene un tamaño igual al `flex-basis`, mientras que el elemento número 1 crece un poco más y el elemento número 2 crece un poco más que el número 1.

flex-shrink 1| <número> **Número que indica el factor de decrecimiento del ítem (si ocupa el 100% del padre).**

Esta propiedad es la opuesta a la propiedad flex-grow. Mientras que la anterior indica un factor de crecimiento, esta hace justo lo contrario: aplica un factor de decrecimiento.

La propiedad flex-shrink actúa en situaciones donde:

- ❑ Hay un flex-basis definido.
- ❑ Los ítems no cubren el tamaño total del contenedor flex padre.

```
.container {  
  display: flex;  
  justify-content: center;  
  align-items: center;  
  background: grey;  
  height: 150px;  
  gap: 20px;  
}
```

```
.item {  
  flex-basis: 600px;  
  background: red;  
}
```

```
.item1 { flex-shrink: 2; }  
.item2 { flex-shrink: 6; }
```

Ahora, hemos definido un flex-basis más grande, para obligar que ocupe más del 100% del contenedor flex padre. Si al primer elemento le aplicamos un flex-shrink: 2 y al segundo elemento un flex-shrink: 6, veremos que el primer elemento va a ser ligeramente más pequeño que el tercero, mientras que el segundo elemento va a ser considerablemente más pequeño.

Página flexbox

<https://lenguajecss.com/css/maquetacion-y-colocacion/flex/>