



Flexbox

El Módulo de Caja Flexible, comúnmente llamado flexbox, fue diseñado como un modelo unidimensional de layout, y como un método que pueda ayudar a distribuir el espacio entre los ítems de un contenedor y mejorar las capacidades de alineación.

Cuando describimos a flexbox como unidimensional destacamos el hecho que flexbox maneja el layout en una sola dimensión a la vez -ya sea como fila o como columna-. Esto contrasta con el modelo bidimensional del Grid Layout de CSS, el cual controla columnas y filas a la vez.

Los dos ejes de flexbox

Cuando trabajamos con flexbox necesitamos pensar en términos de dos ejes — el eje principal y el eje cruzado.

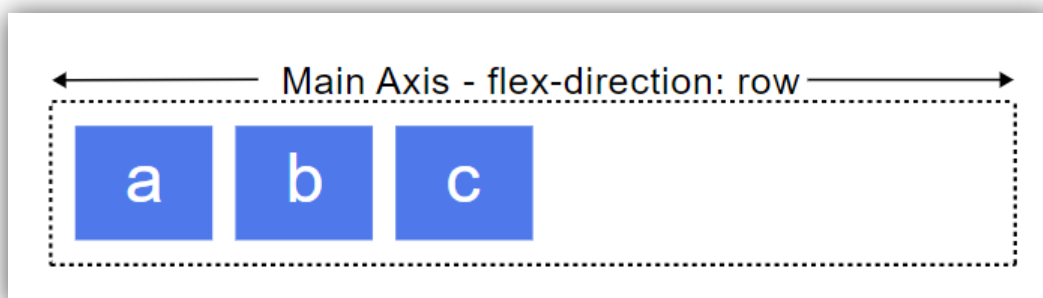
El eje principal está definido por la propiedad [flex-direction](#), y el eje cruzado es perpendicular a este. Todo lo que hacemos con flexbox está referido a estos dos ejes, por lo que vale la pena entender cómo trabajan desde el principio.

El eje principal

El eje principal está definido por `flex-direction`, que posee cuatro posibles valores:

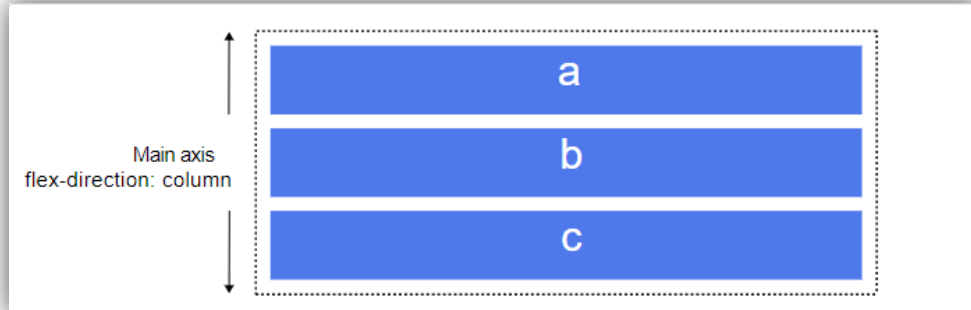
- `row`
- `row-reverse`
- `column`
- `column-reverse`

Si elegimos `row` o `row-reverse`, el eje principal correrá a lo largo de la fila según la dirección de la línea.



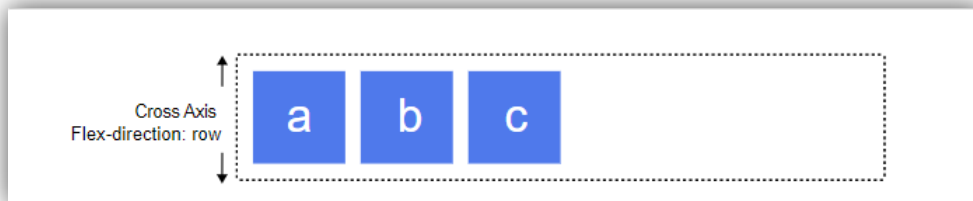


Al elegir **column** o **column-reverse** el eje principal correrá desde el borde superior de la página hasta el final — según la dirección del bloque.

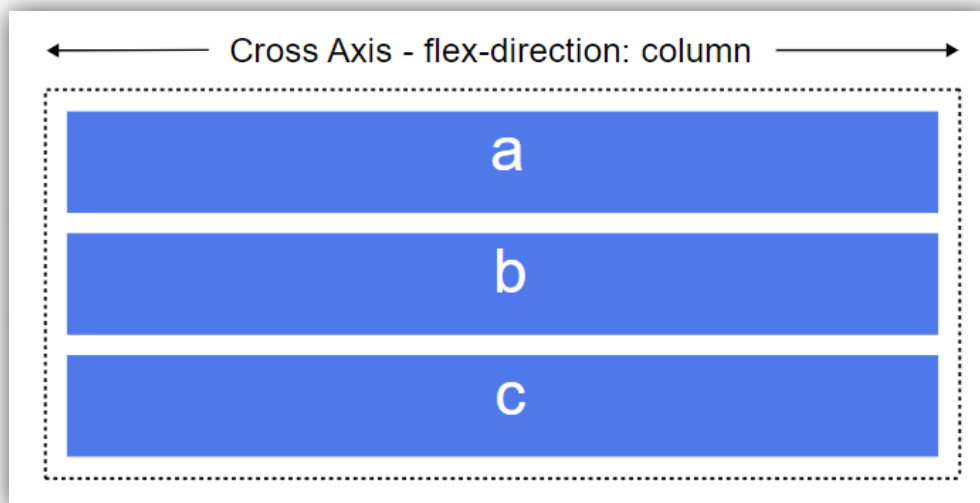


El eje cruzado

El eje cruzado va perpendicular al eje principal, y por lo tanto si flex-direction (del eje principal) es **row** o **row-reverse** el eje cruzado irá por las columnas.



Si el eje principal es **column** o **column-reverse** entonces el eje cruzado corre a lo largo de las filas.





Entender cuál eje es cuál es importante cuando empezamos a mirar la alineación y justificación flexible de los ítems; flexbox posee propiedades que permiten alinear y justificar el contenido sobre un eje o el otro.


El contenedor flex

Un área del documento que contiene un flexbox es llamada contenedor flex. Para crear un contenedor flex, establecemos la propiedad del área del contenedor `display` como `flex` o `inline-flex`. Tan pronto como hacemos esto, los hijos directos de este contenedor se vuelven ítems flex. Como con todas las propiedades de CSS, se definen algunos valores iniciales, así que cuando creamos un contenedor flex todos los ítems flex contenidos se comportarán de la siguiente manera.

- Los ítems se despliegan sobre una fila (la propiedad `flex-direction` por defecto es `row`).
- Los ítems empiezan desde el margen inicial sobre el eje principal.
- Los ítems no se ajustan en la dimensión principal, pero se pueden contraer.
- Los ítems se ajustarán para llenar el tamaño del eje cruzado.
- La propiedad `flex-basis` es definida como `auto`.
- La propiedad `flex-wrap` es definida como `nowrap`.

El resultado es que todos los ítems se alinearán en una solo fila, usando el tamaño del contenedor como su tamaño en el eje principal. Si hay más ítems de los que caben en el contenedor, estos no pasarán más abajo si no que sobrepasarán el margen. Si hay ítems más altos que otros, todos los ítems serán ajustados en el eje cruzado para alcanzar al mayor.

Se puede ver en el ejercicio de abajo cómo luce. Intente editar el ítem o añadir ítems adicionales para así probar el comportamiento inicial de flexbox.




```
.box {  
  display: flex;  
}
```

```
<div class="box">  
  <div>One</div>  
  <div>Two</div>  
  <div>Three  
    <br>has  
    <br>extra  
    <br>text  
  </div>  
</div>
```



El ejemplo de abajo tiene `flex-direction` puesto como `row-reverse`.



```
CSS
```

```
.box {  
  display: flex;  
  flex-direction: row-reverse;  
}
```

```
HTML
```


```
<div class="box">  
  <div>One</div>  
  <div>Two</div>  
  <div>Three</div>  
</div>
```

Contenedores flex Multi-línea con flex-wrap

Si bien flexbox es un modelo unidimensional, es posible lograr que nuestros ítems flex sean repartidos en varias líneas. Haciendo esto, se deberá considerar cada línea como un nuevo contenedor flex. Cualquier distribución del espacio solo sucederá dentro de esa línea, sin referenciar las líneas colaterales.

Para lograr repartirse en varias líneas añade la propiedad `flex-wrap` con el valor `wrap`. Cuando los ítems sean demasiados para desplegarlos en una línea, serán repartidos en la línea siguiente. El ejemplo de abajo contiene ítems que se les ha asignado un ancho, donde el ancho total de los ítems excede al del contenedor flex. Cuando `flex-wrap` se coloca como `wrap`, los ítems se repartirán.

Cuando `flex-wrap` se coloca como `nowrap`, el cual es el valor inicial, estos se contraerán para calzar con el contenedor ya que usan los valores iniciales de flexbox que permiten que los ítems se contraigan. Al usar `nowrap` los ítems podrían salirse del margen si estos no pudieran contraerse, o no contraerse lo suficiente para ser calzados.



```
CSS
```

```
.box {  
  display: flex;  
  flex-wrap: wrap;  
}
```

```
HTML
```

```
<div class="box">  
  <div>One</div>  
  <div>Two</div>  
  <div>Three</div>  
</div>
```



Propiedades aplicadas a los ítems flex

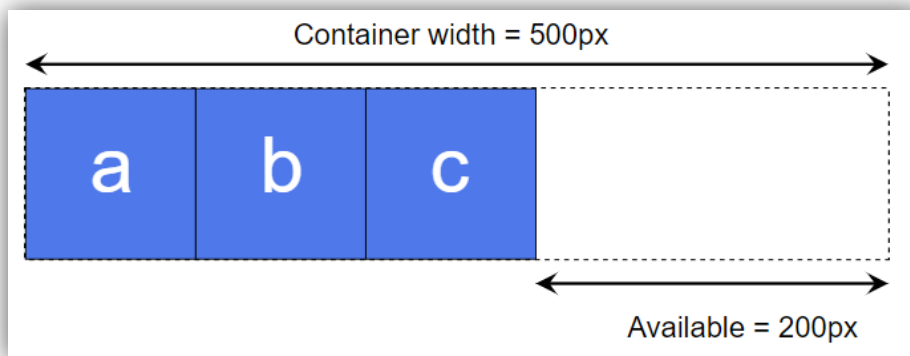
Para obtener más control sobre los ítems flex podemos apuntarlos directamente. Hacemos esto a través de tres propiedades:

- `flex-grow`
- `flex-shrink`
- `flex-basis`

Daremos un breve vistazo a estas propiedades en este resumen, y en un próximo artículo ahondaremos sobre su comportamiento.

Antes de darle sentido a estas propiedades debemos considerar el concepto de espacio disponible. Lo que hacemos cuando cambiamos el valor de alguna de estas propiedades es cambiar la forma que se distribuye el espacio disponible entre nuestros ítems. Este concepto de espacio disponible es también importante cuando veamos la alineación de ítems.

Si tenemos tres ítems con un ancho de 100 píxeles en un contenedor de 500 píxeles de ancho, entonces el espacio que se necesita para colocar nuestros ítems es de 300 píxeles. Esto deja 200 píxeles de espacio disponible. Si no cambiamos los valores iniciales entonces flexbox colocará ese espacio después del último ítem.



Si en cambio quisiéramos que los ítems crecieran para llenar ese espacio, entonces necesitaremos un método para distribuir el espacio sobrante entre los ítems. Es justo lo que harán las propiedades flex que aplicaremos a dichos ítems.

La propiedad `flex-basis`

Con `flex-basis` se define el tamaño de un ítem en términos del espacio que deja como espacio disponible. El valor inicial de esta propiedad es `auto` — en este caso el navegador revisa si los ítems



definen un tamaño. En el ejemplo de arriba, todos los ítems tienen un ancho de 100 píxeles así que este es usado como `flex-basis`.

Si los ítems no tienen un tamaño entonces el tamaño de su contenido es usado como `flex-basis`. Y por eso, apenas declarado `display: flex` en el padre a fin de crear ítems flex, todos estos ítems se ubicaron en una sola fila y tomaron solo el espacio necesario para desplegar su contenido.

La propiedad `flex-grow`

Con la propiedad `flex-grow` definida como un entero positivo, los ítems flex pueden crecer en el eje principal a partir de `flex-basis`. Esto hará que el ítem se ajuste y tome todo el espacio disponible del eje, o una proporción del espacio disponible si otro ítem también puede crecer.

Si le damos a todos los ítems del ejemplo anterior un valor `flex-grow` de 1 entonces el espacio disponible en el contenedor flex será compartido igualmente entre estos ítems y se ajustarán para llenar el contenedor sobre el eje principal.

Podemos usar `flex-grow` apropiadamente para distribuir el espacio en proporciones. Si otorgamos al primer ítem un valor `flex-grow` de 2 y a los otros un valor de 1, entonces 2 partes serán dadas al primer ítem (100px de 200px en el caso del ejemplo de arriba) y 1 parte para cada uno de los restantes (cada uno con 50px de los 200px en total).

La propiedad `flex-shrink`

Así como la propiedad `flex-grow` se encarga de añadir espacio sobre el eje principal, la propiedad `flex-shrink` controla como se contrae. Si no contamos con suficiente espacio en el contenedor para colocar los ítems y `flex-shrink` posee un valor entero positivo, el ítem puede contraerse a partir de `flex-basis`. Así como podemos asignar diferentes valores de `flex-grow` con el fin que un ítem se expanda más rápido que otros — un ítem con un valor más alto de `flex-shrink` se contraerá más rápido que sus hermanos que poseen valores menores.

El tamaño mínimo del ítem tendrá que ser considerado cuando se determine un valor de contracción que pueda funcionar, esto significa que `flex-shrink` tiene el potencial de comportarse menos consistentemente que `flex-grow`.

i Nota: Nótese que los valores de `flex-grow` y `flex-shrink` son proporciones. Típicamente si pusiéramos todos los ítems flex: `1 1 200px` y luego quisiéramos que un ítem creciera al doble, deberíamos ponerlo con flex: `2 1 200px`. Aunque igualmente podemos colocar flex: `10 1 200px` y flex: `20 1 200px` si quisiéramos.



Valores abreviados para las propiedades flex

Difícilmente veremos la propiedades `flex-grow`, `flex-shrink` y `flex-basis` usadas individualmente; si no que han sido combinadas en la abreviación `flex`.

La abreviación flex permite establecer los tres valores en este orden: flex-grow, flex-shrink, flex-basis.

El ejemplo de más abajo permite probar los diferentes valores de la abreviación `flex`; recuerde que el primer valor es `flex-grow`. Dándole un valor positivo significa que el ítem puede crecer. El segundo es `flex-shrink` -con un valor positivo los ítems pueden contraerse-. El valor final es `flex-basis`; este es el valor que los ítems usan como valor base para crecer y contraerse.



CSS

```
.box {  
  display: flex;  
}  
  
.one {  
  flex: 1 1 auto;  
}  
  
.two {  
  flex: 1 1 auto;  
}  
  
.three {  
  flex: 1 1 auto;  
}
```

HTML

```
<div class="box">  
  <div class="one">One</div>  
  <div class="two">Two</div>  
  <div class="three">Three</div>  
</div>
```

Alineación, justificación y distribución del espacio libre entre ítems

Una característica clave de flexbox es la capacidad de alinear y justificar ítems sobre los ejes principal y cruzado, y distribuir el espacio entre los ítems flex.



align-items

La propiedad `align-items` alineará los ítems sobre el eje cruzado.

El valor inicial para esta propiedad es `stretch` razón por la cual los ítems se ajustan por defecto a la altura de aquel más alto. En efecto se ajustan para llenar el contenedor flex -el ítem más alto define la altura de este-.

En cambio definimos `align-items` como `flex-start` para que los ítems se alineen al comienzo del contenedor flex, `flex-end` para alinearlos al final, o `center` para alinearlos al centro.

Intente esto en el ejemplo -He definido en el contenedor flex una altura para que se aprecie que se pueden mover libremente dentro del contenedor-.

Vea lo que sucede si se coloca el valor `align-items` como:

- `stretch`
- `flex-start`
- `flex-end`
- `center`

justify-content

La propiedad `justify-content` es usada para alinear los ítems en el eje principal, cuyo `flex-direction` define la dirección del flujo. El valor inicial es `flex-start` que alineará los ítems al inicio del margen del contenedor, pero también se podría definir como `flex-end` para alinearlos al final, o `center` para alinearlos al centro.

También podemos usar `space-between` para tomar todo el espacio sobrante después de que los ítems hayan sido colocados, y distribuir de forma pareja los ítems para que haya un espacio equitativo entre cada ítem. O bien, usamos el valor `space-around` para crear un espacio equitativo a la derecha e izquierda de cada ítem.

Pruebe con los siguientes valores de `justify-content` en el ejemplo:

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