



*¿QUÉ ES FLEXBOX?*

Flexbox es un modo de diseño que nos permite **crear estructuras para sitios web de una forma más fácil**. Si ya sabes de HTML y CSS probablemente alguna vez habrás visto que los sitios web se realizan utilizando la propiedad float, para desplazar contenedores. **Con Flexbox ya no necesitarás usar float para posicionar tus elementos**, al contrario, con Flexbox podrás posicionar y distribuir los elementos como tú quieras.

**Puedes posicionar un elemento en la posición que desees horizontalmente y por si fuera poco también en forma vertical**. Posicionar elementos de forma vertical no es tarea fácil sin Flexbox, incluso muchas veces teníamos que hacer uso de Javascript para hacer cálculos y saber cuánto margen poner a un elemento para centrarlo verticalmente.

No solo puedes posicionar elementos vertical y horizontalmente, sino que puede establecer cómo se distribuirán, el orden que tendrán e incluso el tamaño que tendrán en proporción a otros elementos. **Esto es perfecto para crear diseños adaptables a dispositivos móviles** (Responsive Design).

| .flex-container {  display: flex;  } |
| --- |
|

Flexbox no es una propiedad ni un conjunto de propiedades. **Flexbox es un nuevo modelo de layout que viene a incorporarse a los ya existentes en css (se está trabajando actualmente en nuevo modelo: el grid layout) y sustituye los modelados anteriores;**

* **Block** (los elementos aparecen uno debajo de otro ocupando todo el ancho disponible)
* **Inline** (los elementos aparecen uno al lado del otro en una línea y saltan a la línea siguiente al ocupar el espacio disponible)
* **Table** (los elementos imitan la distribución de una tabla HTML, con filas, encabezados y columnas).
* **Positioned** (los elementos pueden romper el flujo y posicionarse en cualquier lugar del documento).

**Un modelo de layout es un set de algoritmos que determinan el tamaño y la posición de los elementos con respecto a sus hermanos y ancestros.**

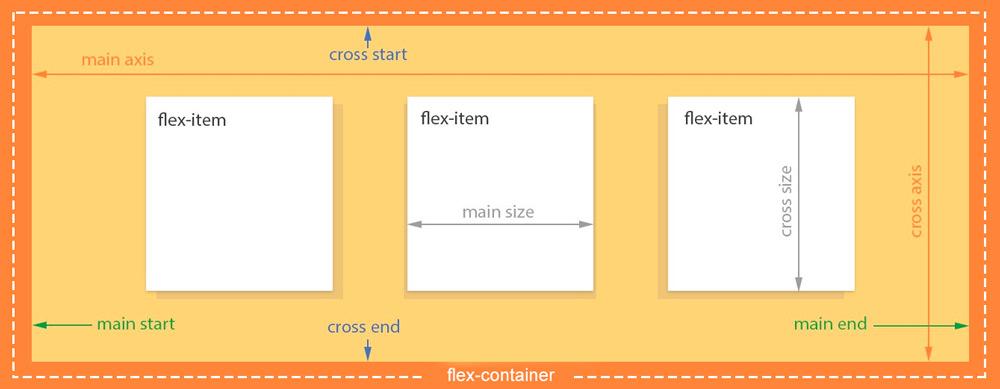
# **¿Qué se puede hacer con flexbox?**

Entendiendo lo que significa un modelo de layout, con flexbox podemos hacer lo siguiente:

* Distribuir los elementos en sentido vertical u horizontal.
* Reordenar la aparición de los elementos sobreescribiendo su aparición en el navegador.
* Ajustar dinámicamente las dimensiones de los elementos para evitar desbordamientos (overflow) respecto a su padre.
* Redefinir el sentido del flujo de los elementos (hacia arriba, hacia abajo, hacia la izquierda o hacia la derecha).
* Alinear los elementos respecto al padre o respecto a sus hermanos.

# **Conceptos básicos de Flexbox**

Para entender bien este modelo de Layout debemos de entender algunos conceptos básicos. Primero la disposición flex debe de estar constituido por elementos padres e hijos, **el padre será el contenedor Flexible “flex container” y los hijos inmediato serán los elementos Flexibles “flex item”.**

****

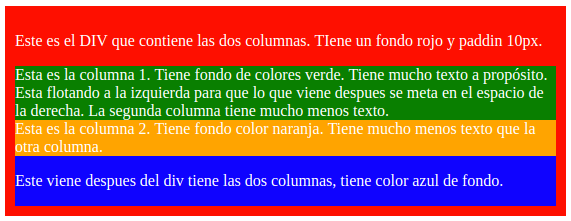
En la imagen anterior vemos como la W3C nos presenta gráficamente el modelo flexbox con sus diferentes propiedades y su pensado funcionamiento

**Propiedades del Flexbox:**

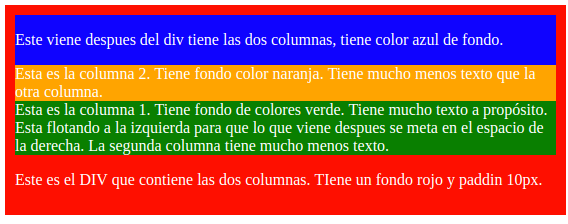
* flex-direction
* flex-wrap
* flex-flow
* justify-content
* align-items
* Align-content

**Flex-direction:** Indica la dirección en la que se posicionarán los elementos. Se tiene como opciones: *column, column-reverse, row, row-reverse.*

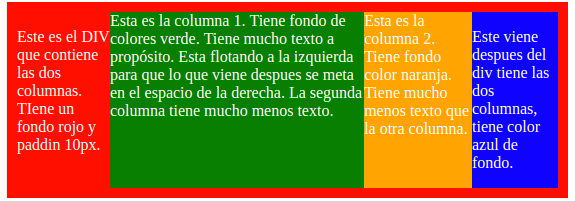
| .flex-container {  display: flex;  flex-direction: column;  } |
| --- |
|



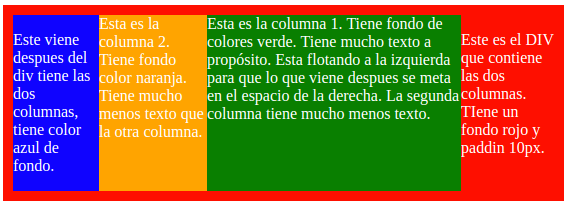
| .flex-container {  display: flex;  flex-direction: column-reverse;  } |
| --- |
|



| .flex-container {  display: flex;  flex-direction: row;  } |
| --- |
|



| .flex-container {  display: flex;  flex-direction: row-reverse;  } |
| --- |
|



*PROPIEDADES DE PADRES E HIJOS*

Propiedades para aplicar en el contenedor flexible

* *display: flex*
* *flex-direction*
* *flex-wrap*
* *flex-flow*
* *justify-content*
* *align-items*
* *align-content*

Propiedades para aplicar a los flex items

* *order*
* *flex-grow*
* *flex-shrink*
* *flex-basis*
* *flex*
* *align-self*

# ¿Cómo empezamos con Flexbox?

Para comenzar a utilizar las bondades de Flexbox lo primero que debemos hacer es establecer la propiedad display con el valor flex en el elemento padre.

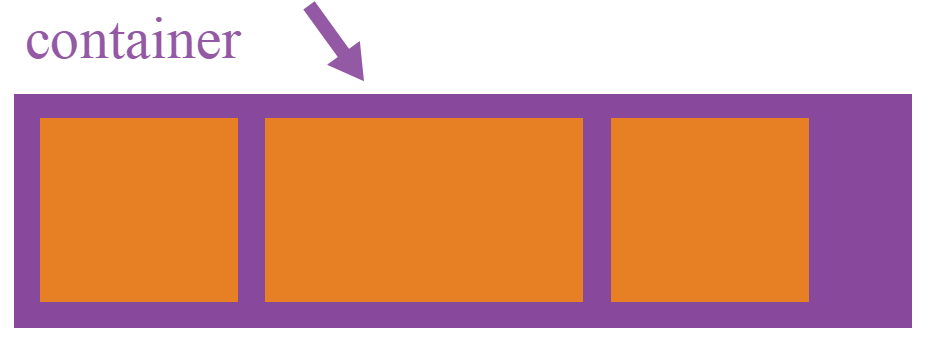
| .flex-container {  display: flex;  } |
| --- |
|

*display: flex* es la única propiedad que necesitamos para configurar el contenedor principal y de esta manera todos sus hijos inmediatos se convertirán en elementos flexibles de forma automática.



El conjunto de propiedades que nos presenta la especificación CSS3 en cuanto al módulo Flexbox las podemos dividir en dos grupos, **las propiedades para el elemento contenedor flex y las otras propiedades para los elementos hijos flexibles**.

Propiedades para el contenedor Flex

****

*(Fuente:* [*https://css-tricks.com/snippets/css/a-guide-to-flexbox/*](https://css-tricks.com/snippets/css/a-guide-to-flexbox/)*)*

# Flex-direction

Esta propiedad me va a **permitir manejar el direccionamiento de los flex items**, es decir, me va a permitir cambiar el flujo normal de los elementos flexibles en el main axis. Esta propiedad tiene múltiples valores que nos van a permitir variar de todas las formas el direccionamiento de los elementos hijos flexibles. **En conclusión esta propiedad nos va a permitir especificar si queremos que los flex items se dispongan en filas o columnas.**

Veamos los valores de flex-direction:

## ***flex-direction: row;***

| .flex-container {  display: flex;  flex-direction: row;  } |
| --- |
|

Con el valor *row* los flex items se apilan **en una fila de izquierda a derecha** tomando como eje principal el main axis. Este es el valor que se aplica por defecto en la propiedad *flex-direction*.



## ***flex-direction: row-reverse;***

| .flex-container {  display: flex;  flex-direction: row-reverse;  } |
| --- |
|

Con el valor *row-reverse* (fila inversa) **los flex items se apilan en una fila de derecha a izquierda.**

****

## ***flex-direction: column;***

| .flex-container {  display: flex;  flex-direction: column;  } |
| --- |
|

Con el valor *column* los flex items se apilan en **una columna de arriba hacia abajo**, esta vez tomando como eje principal el cross axis.



## ***flex-direction: column-reverse;***

| .flex-container {  display: flex;  flex-direction: column-reverse;  } |
| --- |
|

Con el valor column-reverse **los flex items se apilan en una columna de abajo hacia arriba**, tomando como eje principal el cross axis



# Flex-wrap

El comportamiento inicial del contenedor flexible es poder mantener **los flex items en su main axis o eje horizontal sin importar que las dimensiones de estos items cambien**, pero hay ocasiones donde vamos a querer controlar este alineamiento y hacer que los elementos puedan saltar de linea para poder mantener una apariencia deseada en estos flex items. Con *flex-wrap* vamos a poder especificar si queremos que los items puedan saltar a una nueva línea si el contenedor flexible se queda sin espacio.

Veamos los valores de la propiedad flex-wrap:

## ***flex-wrap: nowrap;***

| .flex-container {  display: flex;  flex-wrap: nowrap;  } |
| --- |
|

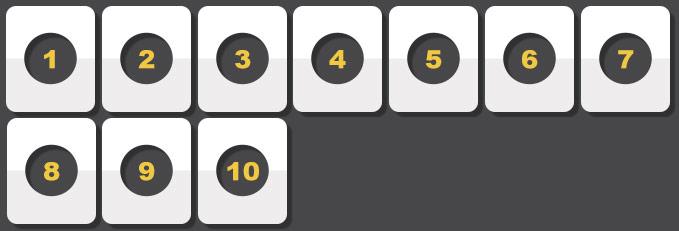
*nowrap* es el valor que se aplica por defecto en la propiedad *flex-wrap*, este valor **hace que los elementos encajen en el ancho del contenedor** flexible aun modificando la apariencia de estos, este es el valor que se aplica por defecto en la propiedad *flex-wrap*.



## ***flex-wrap: wrap;***

| .flex-container {  display: flex;  flex-wrap: wrap;  } |
| --- |
|

Con este valor en la propiedad *flex-wrap* **los flex items pueden romper la linea del eje horizontal si les es necesario** para así conservar las características de dimensiones de los flex items. Esto es de izquierda a derecha y de arriba a abajo.



## ***flex-wrap: wrap-reverse;***

| .flex-container {  display: flex;  flex-wrap: wrap-reverse;  } |
| --- |
|

Con el valor *wrap-reverse* la propiedad *flex-wrap* los flex items también pueden romper la linea del eje horizontal si les es necesario. Pero **esta vez el orden es de izquierda a derecha y de abajo a arriba.**

# Flex-flow

La propiedad *flex-flow* es simplemente **la forma shorthand o forma rápida para las propiedades *flex-direction* y *flex-wrap* vistas anteriormente**. Los valores que se aplican por defecto a esta propiedad son las mismas que se aplican a las propiedades que están actuando como valor. row nowrap.

Valores:

* valores de la propiedad flex-direction
* valores de la propiedad flex-wrap

| .flex-container {  display: flex;  flex-flow: row-reverse wrap;  } |
| --- |
|

# Justify-content

*Justify-content* nos va a permitir **alinear los elementos en el main axis de la linea actual del contenedor flexible**, esto puede ser de forma vertical o horizontal según lo especifiquemos con *flex-direction*, también nos va a ayudar a distribuir los flex items en el contenedor flexible cuando los items no utilicen todo el espacio disponible en su eje principal actual. **Esto es declarar la forma en que el navegador debe distribuir el espacio disponible entre los items flexibles.**

Veamos los valores de la propiedad justify-content:

## ***justify-content: flex-start***

| .flex-container {  display: flex;  justify-content: flex-start;  } |
| --- |
|

El valor *flex-start* de la propiedad *justify-content* lo que hace **es alinear los flex items en el main start del contenedor flexible es decir alineados al lado izquierdo**, este es el valor que se aplica por defecto en la propiedad *justify-content.*

**

## ***justify-content: flex-end;***

| .flex-container {  display: flex;  justify-content: flex-end;  } |
| --- |
|

El valor *flex-end* de la propiedad *justify-content* lo que hace es **alinear los flex items en el main end del contenedor flexible es decir alineados al lado derecho**.

****

## ***justify-content: center;***

| .flex-container {  display: flex;  justify-content: center;  } |
| --- |
|

El valor *flex-center* lo que hace es **alinear los flex items en el centro del contenedor flexible.**



## ***justify-content: space-between;***

| .flex-container {  display: flex;  justify-content: space-between;  } |
| --- |
|

*space-between* va a hacer que **los flex items se tomen la misma distancia o espaciado entre ellos dentro del contenedor** flexible quedando el primer y ultimo elemento alineados con los bordes del contenedor en el eje principal.



## ***justify-content: space-around;***

| .flex-container {  display: flex;  justify-content: space-around;  } |
| --- |
|

*space-around* muestra **los flex items con el mismo espacio de separación entre si, en diferencia del valor space-between los items primero y ultimo toman también el espaciado entre los bordes del contenedor flexible.**  


# Align-items

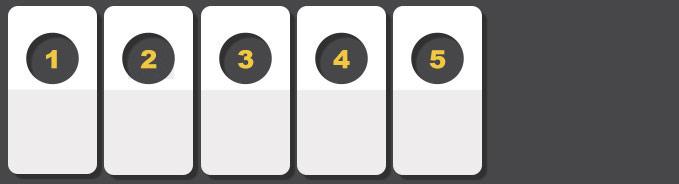
*Align-items* nos permite establecer la **alineación que tendrán por defecto los flex item** incluyendo los anónimos, es similar a la propiedad *justify-content* pero **esta vez la dirección es perpendicular**. Es decir *align-items* nos va a permitir los items en el eje secundario del contenedor flex.

Veamos los valores de la propiedad flex-wrap:

## ***align-items: stretch;***

| .flex-container {  display: flex;  align-items: stretch;  } |
| --- |
|

El valor stretch (estirar) para la propiedad *align-items* lo que va a hacer es **tratar de llenar toda la altura (o anchura) desde el cross start hasta el cross end del contenedor** flexible, siempre y cuando los items no tengan propiedades de dimensión definidas.



## ***align-items: flex-start;***

| .flex-container {  display: flex;  align-items: flex-start;  } |
| --- |
|

Con *flex-start* **los flex** **items se apilan en el cross start del contenedor** flexible, es decir en el inicio transversal, este es el valor que se aplica por defecto en la propiedad *align-items.*



## ***align-items: flex-end;***

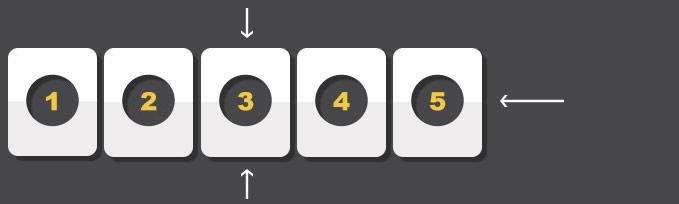
| .flex-container {  display: flex;  align-items: flex-end;  } |
| --- |
|

Con *flex-end* los **flex items se apilan en el cross end del contenedor** flexible

## ***align-items: center;***

| .flex-container {  display: flex;  align-items: center;  } |
| --- |
|

Con *center* los flex **items se apilan en el centro del cross axis del contenedor** flexible, osea en el eje transversal.



## ***align-items: baseline;***

| .flex-container {  display: flex;  align-items: baseline;  } |
| --- |
|

*baseline* **alinea los flex items sus propias lineas de base**, esta se define en base de los texto de cada item.



# Align-content

La propiedad *align-content* **alinea los flex items cuando estos no usan todo el espacio disponible en el cross axis del contenedor** flexible, que de forma predeterminada es verticalmente.

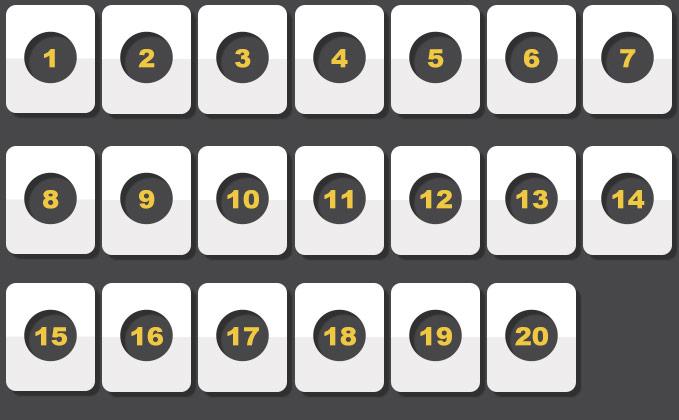
**Nota**: Esta propiedad **sólo tiene efecto cuando el contenedor flexible tiene varias líneas de flex items**. Si se colocan en una sola línea esta propiedad no tiene ningún efecto sobre el diseño.

Veamos los valores de la propiedad *align-content*:

## ***align-content: stretch;***

| .flex-container {  display: flex;  align-content: stretch;  } |
| --- |
|

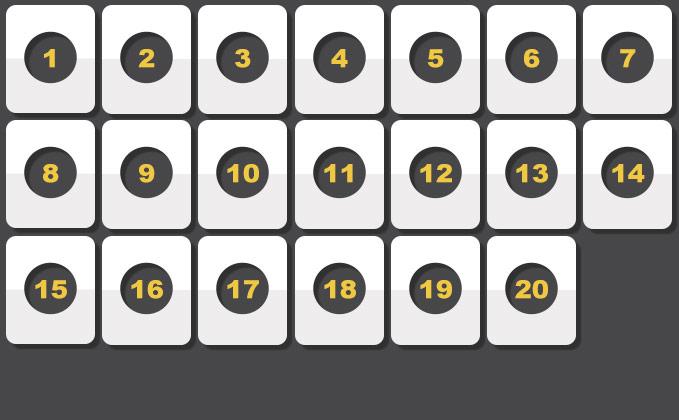
Con el valor *stretch***los flex items son mostrados con espacios distribuidos después de cada fila** compuestas elementos flexibles dentro de su contenedor, este es el valor que se aplica por defecto en la propiedad *align-content*.



## ***align-content: flex-start;***

| .flex-container {  display: flex;  align-content: flex-start;  } |
| --- |
|

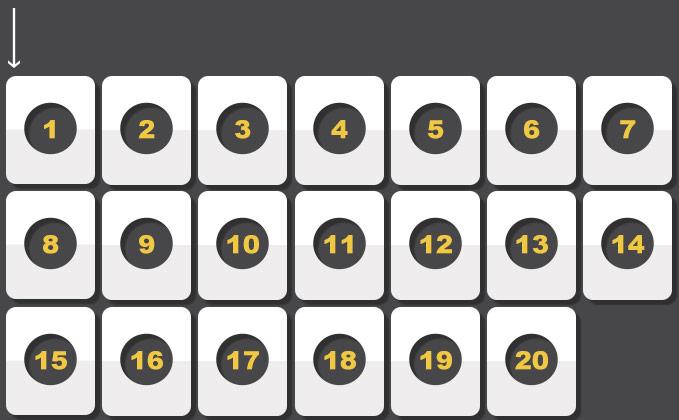
El valor *flex-start* lo que hace es **apilar los flex items partiendo en el inicio transversal (cross start) del contenedor** flexible.



## ***align-content: flex-end;***

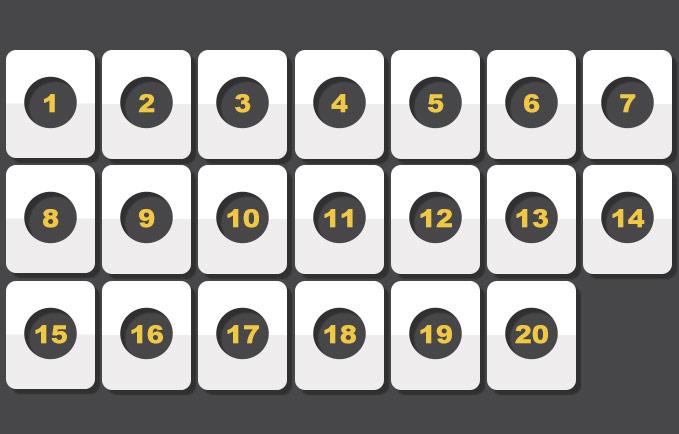
| .flex-container {  display: flex;  align-content: flex-end;  } |
| --- |
|

El valor *flex-end* **apila los flex items hacia el cross end del contenedor** flexible.



## ***align-content: flex-center;***

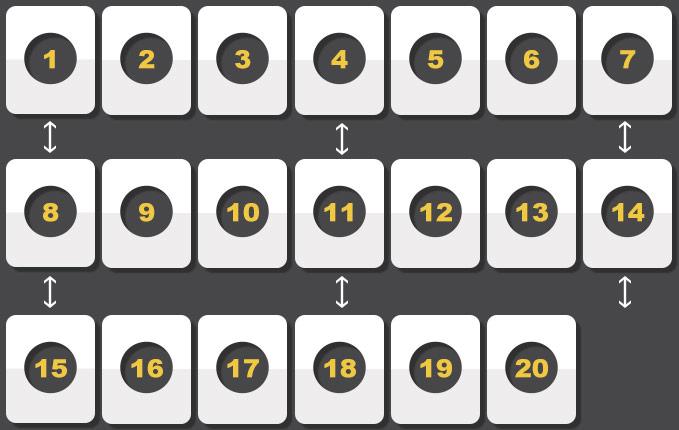
| .flex-container {  display: flex;  align-content: center;  } |
| --- |
|

El valor *flex-end* ajusta **los items hacia el centro del contenedor** flexible.  
  


## ***align-content: space-between;***

| .flex-container {  display: flex;  align-content: space-between;  } |
| --- |
|

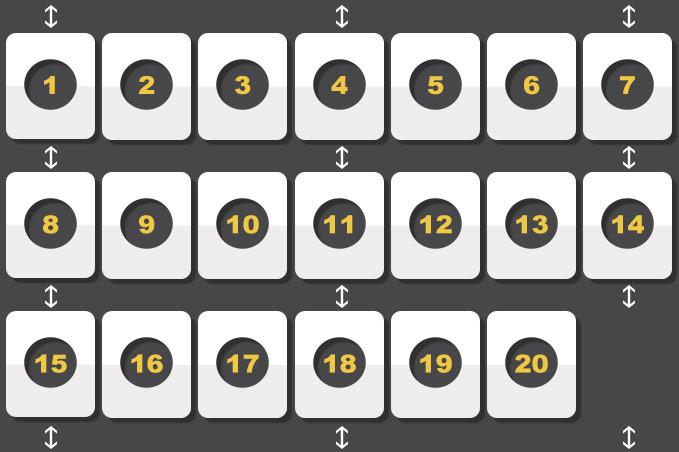
*space-between* en la propiedad *align-content* va a hacer que **los flex items se tomen la misma distancia o espaciado entre ellos dentro del contenedor** flexible quedando el primer y último elemento alineados con los bordes del contenedor en el eje principal (cross axis predeterminado).



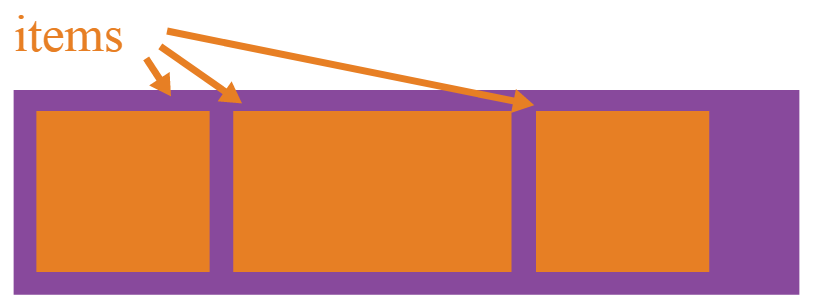
## ***align-content: space-around;***

| .flex-container {  display: flex;  align-content: space-around;  } |
| --- |
|

*space-around* en la propiedad *align-content* muestra **los flex items con el mismo espacio de separación entre si, en diferencia del valor space-between los items primero y ultimo toman también el espaciado entre los bordes del contenedor** flexible.



Propiedades para los Flex items

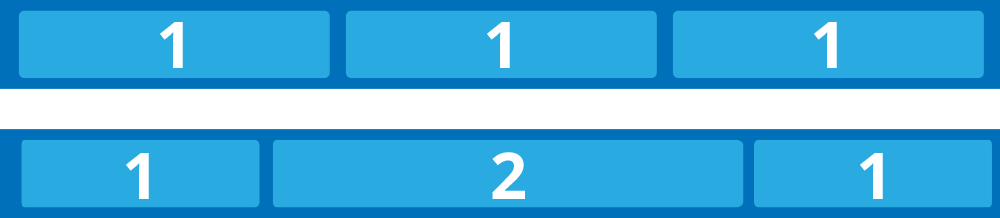


*(Fuente:* [*https://css-tricks.com/snippets/css/a-guide-to-flexbox/*](https://css-tricks.com/snippets/css/a-guide-to-flexbox/)*)*

Esta sección describe las **propiedades que se le pueden asignar a los items** como tal. Algunas de ellas sobreescriben propiedades que se configuran en el container.

# Flex-grow

Esta propiedad define la **capacidad de un elemento de crecer cuando en el contenedor todavia hay espacio sobrante**. Esta propiedad se configura con un valor numérico entero natural (no acepta negativos). Por defecto el valor viene configurado en "0" por lo tanto el elemento no crecerá de manera horizontal. Si este valor es configurado en 1 para todos los items, todos ellos crecerán de igual manera por lo que ocuparan la misma cantidad de espacio dentro del contenedor, si a uno de ellos el valor se configura en "2" dicho elemento ocupará el doble de los demás.



| .item {  flex-grow: <numero>; /\* por defecto 0 \*/  } |
| --- |
|

# Flex-shrink

El shrink configura **la posibilidad de un elemento de encogerse en caso del máximo ancho del contenedor sea alcanzado por todos los elementos.**

| .item {  flex-shrink: <numero>; /\* por defecto 1 \*/  } |
| --- |
|

# Flex-basis

Define el **ancho de un elemento inicial**. Este valor por defecto viene configurado en 0.

| .item {  flex-basis: <ancho> | auto; /\* default auto \*/  } |
| --- |
|

# Flex

El *flex* es la **manera rápida de configurar a un ítem las tres propiedades anteriores**, es decir, en una sola propiedad se configura el *"flex-grow", "flex-shrink" y "flex-basis"*, es ese orden exacto.

| .item {  flex: none | flex-grow flex-shrink flex-basis  } |
| --- |
|

# Order

Esta propiedad **permite modificar el orden de aparición de un elemento**. Recibe como valor numeros enteros.



| #item2 {  order: <integer>;  } |
| --- |
|

Páginas de Interés:

* [Demo de flexbox](https://demo.agektmr.com/flexbox/)
* [Herramienta para probar propiedades de flexbox](http://the-echoplex.net/flexyboxes/)
* [A complete guide to Flexbox](https://css-tricks.com/snippets/css/a-guide-to-flexbox/)
* [How Flexbox works — explained with big, colorful, animated gifs](https://medium.freecodecamp.org/an-animated-guide-to-flexbox-d280cf6afc35)
* [Aprendiendo jugando](https://flexboxfroggy.com/#es)
* <https://codepen.io/JustSharkieCodes/pen/qBEEzWa>