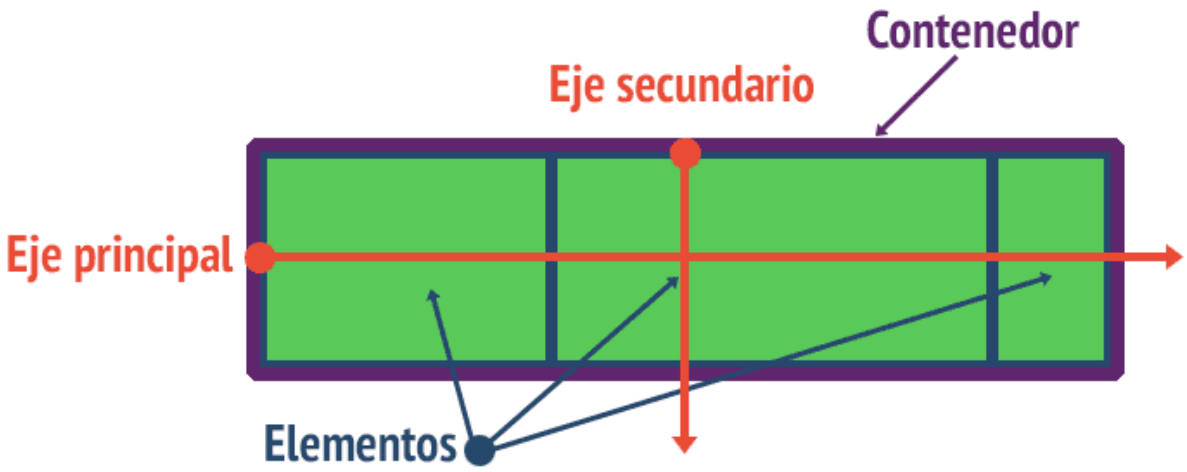
Contenedor e items

Las cosas que se pueden hacer con Flexbox involucran a dos partes fundamentales. Una es el contenedor y otra los ítems que están dentro de él.

Los ejes en Flexbox

En flexbox vamos a tener dos ejes. El eje principal, por defecto, es el eje horizontal y el eje secundario que es el eje vertical. Por defecto significa que nosotros como diseñadores podremos cambiar este comportamiento, de modo que el eje principal sea el vertical y el secundario el horizontal.

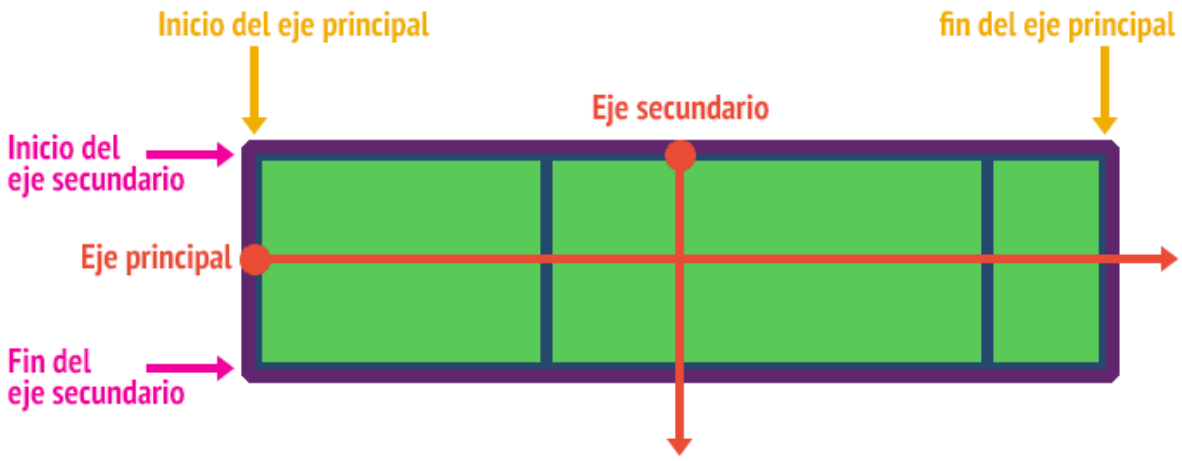
Estos ejes, principal y secundario, implican el modo en el que los ítems se van a posicionar. Todo es configurable, pero para que nos hagamos una idea con un ejemplo, si el eje principal es la horizontal, los ítems se pondrán uno al lado del otro. Si el eje principal fuera la vertical, los ítem se colocarían uno debajo (o arriba) del otro.



Nota: El atributo flex-direction es el que nos permitirá intercambiar el eje principal y secundario. Más adelante lo veremos con ejemplos. Inicio y fin de los ejes.

A la hora de alinear un texto, por ejemplo, con CSS tenemos los conceptos left y right, indicando que queremos una alineación a izquierda o derecha. Esto no funciona justamente igual en Flexbox. En este caso tenemos los conceptos de inicio y fin (start / end).

Como se ha dicho, en flex tenemos dos ejes: principal y secundario, pues existirá un inicio y un fin para cada uno de los ejes.



Dimensiones del contenedor

Aquí las dimensiones continúan siendo la altura y la anchura, definidas con width y height. Sin embargo, la principal de estas dimensiones dependerá de cuál de sus ejes sea el principal. Date cuenta que, si la dirección es de arriba a abajo, entonces la dimensión principal será la altura.

Ejercicio práctico con Flexbox

No quiero todavía entrar en detalle con todas las propiedades disponibles en Flexbox, pero estoy seguro que querrás comenzar a plasmar estas ideas en algo de código con el que aclarar estos conceptos.

Vamos a partir de un HTML sencillo y aplicaremos varios cambios con Flexbox para ver la transformación a la hora de representarse en la página.

```
<section>
  <article>1</article>
  <article>2</article>
```

```
  <article>3</article>
  <article>4</article>
</section>
```

Partimos también de un CSS básico solamente para darle un poco de color a los elementos y poder verlos un poco separados en la página.

```
body {
  font-size: 2em;
  font-family: sans-serif;
  color: #666;
}
article {
  background-color: #9ef;
  margin: 5px;
  padding: 3px;
}
```

Tal cual están estos elementos, su representación en el navegador sería más o menos esta:

Ahora vamos a empezar a aplicar algo de flexbox. Lo primero que tenemos que hacer es decirle al contenedor principal (el SECTION) que debe comportarse como

un elemento "flex". Esto lo conseguimos con el atributo "display", aplicando el valor "flex".

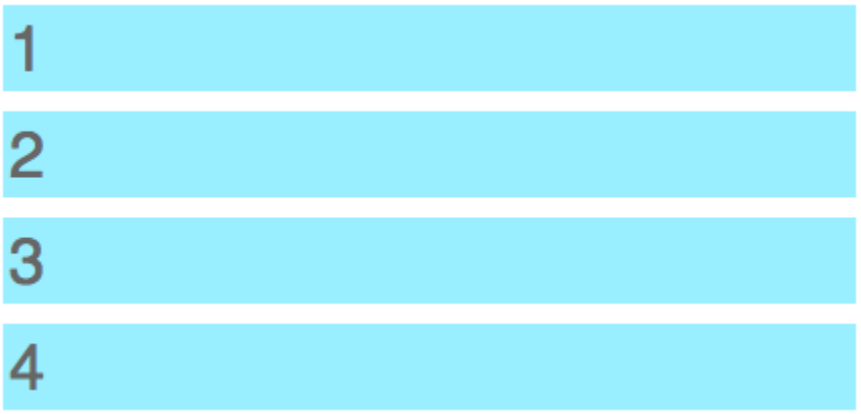
```
section {
  display: flex;
}
```

Solo por haber añadido este comportamiento, nuestros elementos van a colocarse en la página de otra manera totalmente distinta.

Como no hemos indicado nada, el eje predeterminado es el horizontal y por ello es que aparecen uno al lado del otro. Pero podríamos indicar que se colocasen uno debajo del otro si cambiamos el atributo "flex-direction".

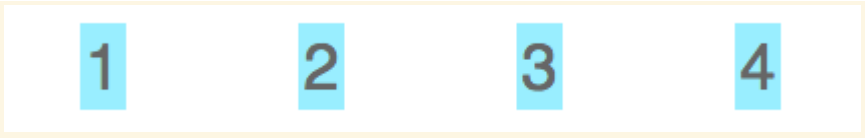
```
section {
  display: flex;
  flex-direction: column;
}
```

Ahora los elementos se muestran como puedes ver en la siguiente imagen:



Vamos a hacer una pequeña alteración en nuestro código para conseguir que, estando dispuestos en el eje horizontal, los elementos tengan espacio entre ellos de modo que se distribuyan uniformemente en todo el contenedor.

```
section {
  display: flex;
  flex-direction: row;
  justify-content: space-around;
}
```



Ahora para acabar este primer acercamiento a Flexbox, vamos a ver cómo podríamos conseguir que los ítem tengan una anchura que permita ocupar todo el espacio disponible en el eje principal. Lo conseguimos con "flex-grow", pero ten en cuenta que esta propiedad ya no depende del contenedor, sino de los item, elementos internos, en nuestro caso los ARTICLE.

> **Nota:** Creo que es obvio, pero lo menciono por si acaso. En mi ejemplo he decidido usar etiquetas SECTION y ARTICLE, pero estas propiedades de flexbox puedes aplicarlas a todo tipo de elementos, independientemente de la etiqueta escogida.

```
article {
  background-color: #9ef;
  margin: 5px;
  padding: 3px;
  flex-grow: 1;
}
```

Habiendo colocado a todos los elementos el mismo valor de flex-grow (1) estamos produciendo que el tamaño que van a ocupar sea idéntico para todos. Se verán como la siguiente imagen.

Con esto terminamos este artículo. Hemos visto un poco de teoría de Flexbox y lo hemos complementado con un poco de práctica para ayudarnos a cristalizar las ideas. En próximos artículos abordaremos con más detalle ya las propiedades de los elementos "flex", tanto contenedores como ítems.

Flexbox CSS: propiedades

Display flex

Al contenedor principal en un esquema Flexbox es al que le asignamos "display: flex". Esta propiedad hace que cambien las reglas con las cuales sus hijos van a ser representados en la página.

```
.contendor-flex {

  display: flex;

}
```

Tan sencillo como eso! a partir de este momento, todos los elementos en la página con la clase "contenedor-flex" se comportarán según las reglas de Flexbox. Ésto implica que sus hijos se van a posicionar de una manera distinta a la habitual. Por lo tanto, debe quedar claro que el propio contenedor no se va a ver afectado, sólo sus hijos.

Display inline-flex

Además del display flex tenemos también el valor "inline-flex". Si conocemos los elementos "inline-block", la diferencia fundamental es la misma que tienen respecto a los elementos "block" normales, que se comportan como un bloque, pero no se exanden para ocupar todo el espacio en la horizontal.

```
.contendor-flex {

  display: inline-flex;

}
```

En resumen, con inline-flex es como si tuviéramos un elemento **inline-block**, donde sus hijos se comportan con las reglas de Flexbox.

Una vez el contenedor es "flex" o "inline-flex" puedo aplicarle toda una serie de propiedades adicionales para personalizar todavía más su comportamiento. Las veremos a continuación.

Propiedad flex-direction

Esta propiedad nos sirve para definir la dirección del flujo de colocación de los elementos. Tiene que ver con los ejes que conocimos en el artículo anterior, pudiendo marcar si los elementos se van a colocar todos en la misma fila, o si se van a colocar en una columna, pero además también permite indicar el orden de los item, normal o reverso.

Permite usar estos valores:

- row (valor predeterminado): Indica que los elementos se colocan en una fila, uno al lado del otro, de izquierda a derecha.
- row-reverse: se colocan en una fila, pero con orden de derecha a izquierda.
- column: se colocan uno debajo del otro, en orden los primeros arriba.

- column-reverse: se colocan en una columna, pero los primeros aparecerán abajo.

Podemos experimentar con esta propiedad con un HTML como este. (De hecho verás que el HTML es muy parecido al del ejercicio anterior y es que con esto nos es suficiente para probar las distintas facetas de Flexbox).

```
<div class="flex-container">

  <div class="item">1</div>

  <div class="item">2</div>

  <div class="item">3</div>

  <div class="item">4</div>

</div>
```

Ahora vamos a aplicar unos estilos. Principalmente lo que nos interesa es aplicar el CSS al elemento con class="flex-container", pero aplicaremos estilos a todos los elementos para que podamos apreciar mejor la posición de las cajas.

```
body {

  font-size: 2.5em;

  font-family: sans-serif;
```

```
  color: #ddd;
}
.flex-container {
  display: flex;
  flex-direction: column-reverse;
}
.item {
  background-color: #369;
  margin: 2px;
  padding: 5px;
  flex-grow: 1;
}
```

Como al contenedor flex le hemos colocado flex-direction: column-reverse; observarás que los elementos se colocan uno debajo del otro y con orden contrario al que aparecen en el código HTML.

Propiedad flex-wrap

Sirve para indicar si queremos que haya saltos de línea en los elementos que se colocan en el contenedor, si es que éstos no caben en el espacio disponible.

De manera predeterminada con Flexbox los elementos se colocan en el eje de la horizontal, en una fila. Si los elementos tienen unas dimensiones tales que no quepan en el contenedor, el comportamiento flex hará que se intenten agrupar en la fila de manera que quepan bien sin saltar de línea, pero también podemos configurarlo para hacer que, si no caben, se pasen a la línea siguiente.

- nowrap (predeterminado): hace que nunca se produzcan saltos de línea.
- wrap: hace que si no caben, entonces se coloquen en la siguiente línea.
- wrap-reverse: El salto de línea se producirá al contrario, o sea, hacia arriba.

> **Nota:** Si tenemos configurado "nowrap" el navegador hará lo que pueda para hacer que los elementos quepan en la fila (si es que flex-direction es row o row-reverse), llegando a alterar las dimensiones definidas en el width de los ítem si fuera necesario. Sin embargo, si por mucho que el navegador lo intente, siguen sin caber los elementos ahí dependerá de otros estilos CSS lo que podrá ocurrir. Con otras propiedades de CSS podremos conseguir que los elementos desborden el contenedor, que no se vean los que no quepan, etc. Por ejemplo usarás la propiedad "overflow" de toda la vida.

```
.flex-container {

  display: flex;

  flex-direction: row;

  flex-wrap: nowrap;

}

.item {

  background-color: #369;

  width: 30%;

  padding: 5px;
```

```
}
```

En este caso hemos colocado "flex-wrap: nowrap" y además cada uno de los ítem hemos indicado que debe tener una anchura de 30%. Recuerda que en nuestro HTML teníamos 4 ítem dentro del flex-container y sin embargo, cuando veamos el resultado obtendremos algo como esto:

Es obvio que 4 contenedores a 30% cada uno no cabrían en la horizontal, pero con flexbox el navegador hace un esfuerzo para ajustarse en una fila.

Ahora bien, si cambiamos para "flex-wrap: wrap" entonces sí se producirá el salto de línea:

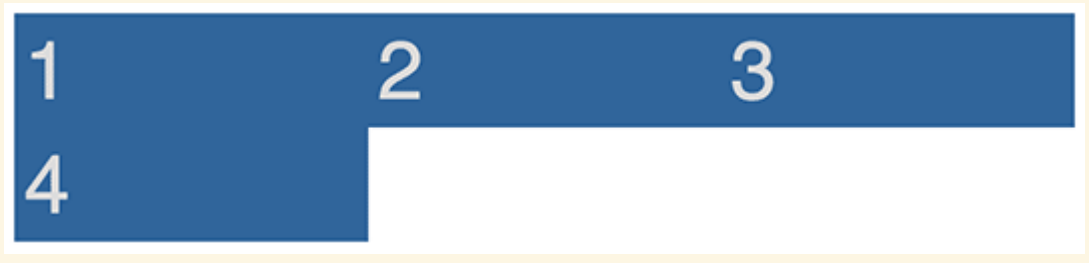
```
.flex-container {

  display: flex;

  flex-direction: row;

  flex-wrap: wrap;

}
```



> **Nota:** Si se te ocurre poner los contenedores a 33% puedes apreciar que aun así no caben 3 en la horizontal, cuando sí debería. Si es el caso, el problema no es Flexbox. Tendrías que ver si acaso por culpa de los margin los elementos no tengan espacio, o dependiendo de la combinación de box-sizing pueden afectar los padding o border. Recuerda la recomendación de usar "box-sizing: border-box".

Propiedad flex-flow

Esta propiedad no aporta nada nuevo, pues simplemente es un atajo para escribir de 1 sola vez flex-direction y flex-wrap. El valor predeterminado es "row nowrap"

```
.flex-container {

  display: flex;

  flex-flow: row wrap;

}
```

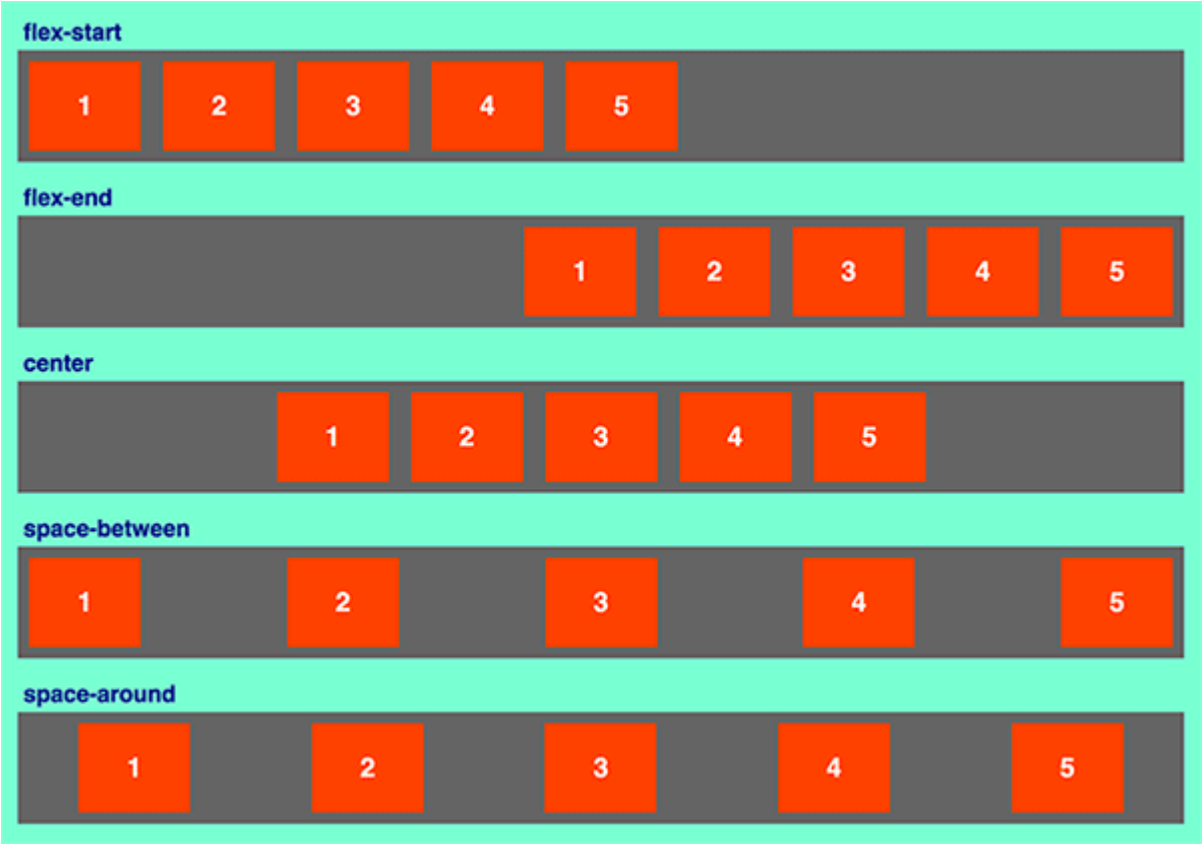
Propiedad justify-content

Esta propiedad es muy útil para indicar cómo se van a colocar los justificados y márgenes de los ítems. Puedes indicar que vayan a justificados al inicio del eje o al final del eje o que a la hora de distribuirse se coloque un espacio entre ellos o un espacio entre ellos y los bordes.

Es interesante como para tratarla de manera independiente y así poder ver varios ejemplos de ella. Veamos simplemente sus posibles valores:

- flex-start: Añade los elementos a partir del inicio del eje principal.
- flex-end: Añade los elementos a partir del final del eje principal.
- center: los elementos se centran en el espacio del contenedor, siempre con respecto al eje principal.
- space-between: hace que los elementos se distribuyan con un espacio proporcional entre ellos, siendo que los ítem de los extremos se sitúan en el borde del contenedor.
- space-around: es parecido a space-between en el sentido de dejar un espaciado proporcional, sin embargo, en esta ocasión se deja también espacio entre el borde del contenedor y los ítem de los extremos.

Lo veremos todo más claro observando la siguiente imagen:



Propiedad align-items

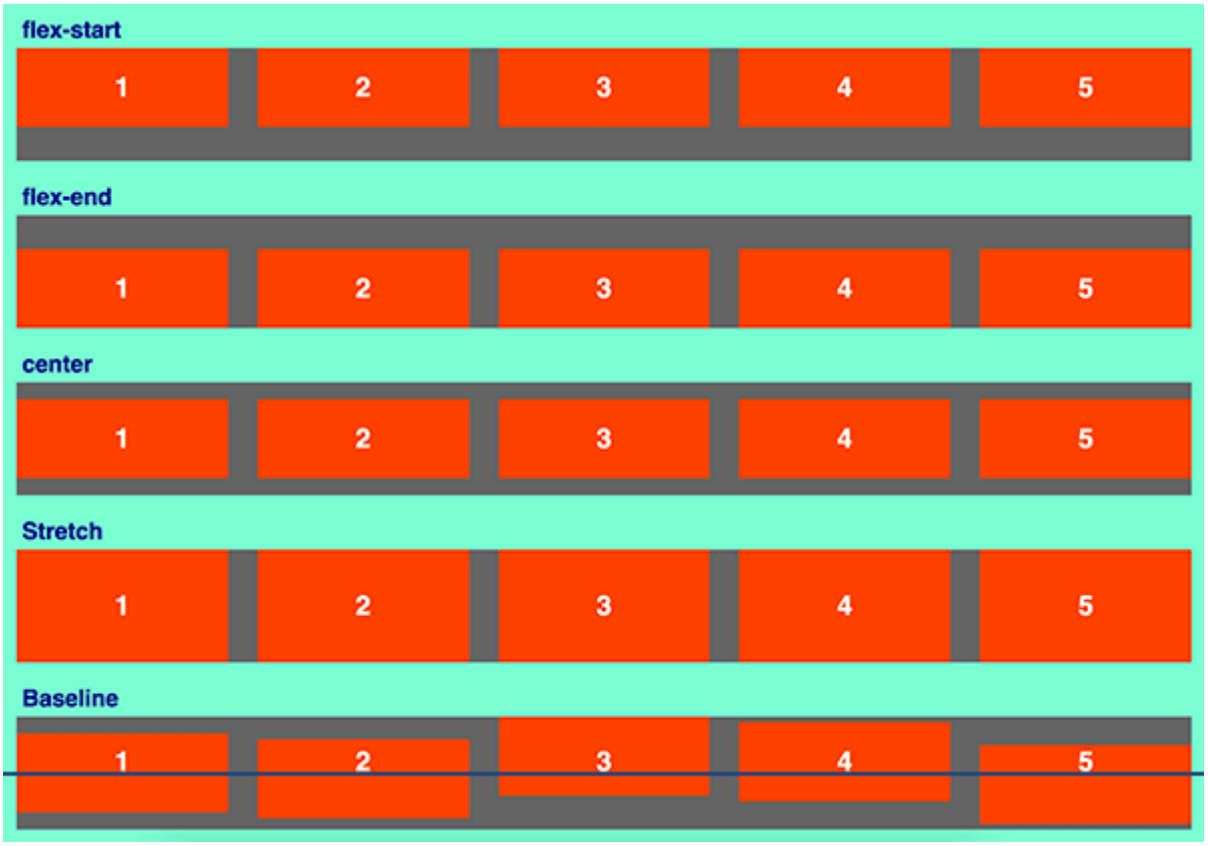
Esta propiedad es muy similar a la propiedad anterior, justify-content, solo que ahora estamos alineando con respecto al eje secundario y no el principal.

En el caso de un contenedor flex cuyo eje principal está en la horizontal, entonces align-items servirá para obtener el alineamiento en el otro eje (vertical, de arriba a abajo). En definitiva, align-items nos ofrece el tan deseado alineamiento vertical que hemos echado en falta en CSS históricamente.

También merece hacer ejemplos específicos para verlo con más detalle, por lo que ahora nos limitaremos a enumerar sus posibles valores con una breve descripción.

- flex-start: indica que se posicionarán al comienzo del eje secundario.
- flex-end: se posicionarán al final del eje secundario.
- center: se posicionarán en el centro del eje secundario.
- stretch: ocuparán el tamaño total del eje secundario (a no ser que hayamos marcado que esos elementos tengan un tamaño diferente).
- baseline: para el posicionamiento de los elementos se tendrá en cuenta el texto que hay escrito dentro.

Como una imagen vale más que mil palabras, se pueden ver aquí los distintos efectos con bastante claridad.



Propiedad align-content

Esta propiedad sólo aplica cuando dispones de varias líneas de elementos en el contenedor flexbox. El efecto que conseguiremos será una alineación y separación de las filas en el eje secundario.

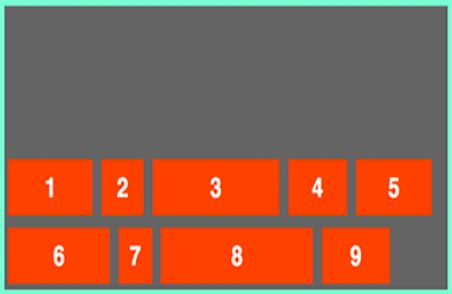
> **Nota:** Para conseguir varias líneas de elementos en el contenedor flex necesitarás aplicarles un "flex-flow: wrap" y por supuesto que haya suficientes elementos, con suficiente anchura, para que se necesiten varias filas para su distribución.

Al final, el efecto que conseguimos con align-content es parecido al que conseguimos con align-items, en el sentido que aplicará al eje secundario su efecto de distribución, solo que aquí no estamos indicando la colocación de una única fila, sino de todas las filas. Además se parece también a justify-content en el sentido que estamos definiendo la separación entre ítems, pero afectando a filas de ítems en vez de ítems sueltos.

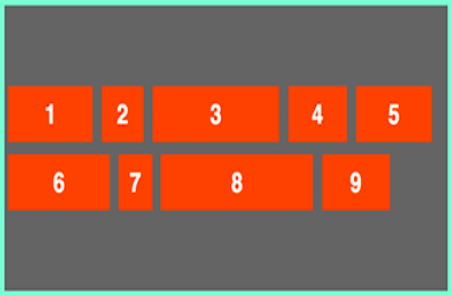
- flex-start: indica que las filas se colocarán todas pegadas entre sí (obviamente no aparecerán exactamente pegadas si le hemos colocado un margin), desde el inicio del eje secundario.
- flex-end: las filas se colocarán pegadas entre sí, pero esta vez pegadas al final del eje secundario.
- center: se posicionarán en el centro del eje secundario, pegadas entre sí.
- stretch: Sus dimensiones crecerán para ocupar todo el espacio disponible (a no ser que se haya colocado una dimensión diferente en los elementos).
- space-between: indica que las filas se separarán entre sí, dejando un espacio proporcional entre ellas.
- space-around: indica que las filas se separarán, dejando un espacio entre ellas proporcional, también con el borde.

flex-start

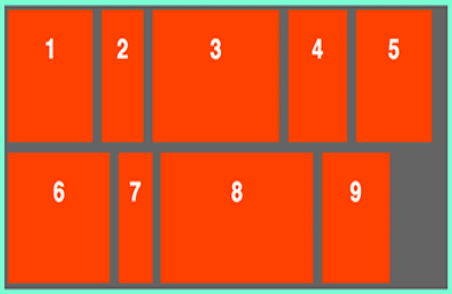
flex-end



center

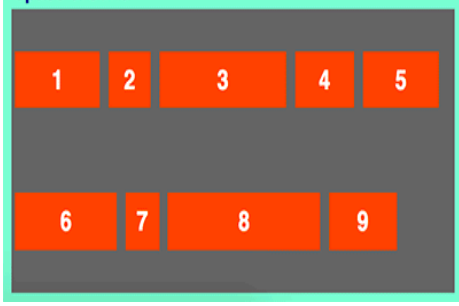


Stretch



space-between

space-around

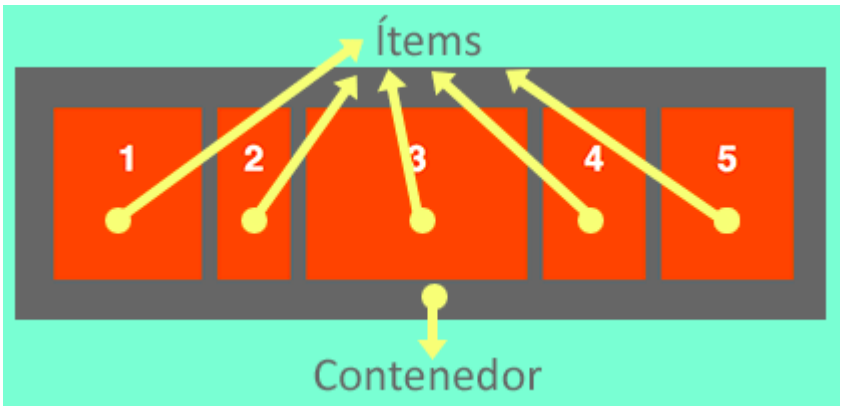


Veamos las propiedades que puedes poner a los hijos de los contenedores Flexbox para alterar su disposición.

En el Manual de Flexbox hemos podido dedicar mucho texto a ofrecer información acerca de los atributos disponibles en los contenedores "Display Flex". Se pueden hacer muchas cosas, algunas de ellas realmente destacadas y novedosas, pero el estándar de posicionamiento no queda ahí.

Ahora vamos a aprender cosas que tienen que ver con los ítem, es decir los hijos directos de los contenedores que se comportan bajo el estándar Flexbox. Veremos las propiedades o atributos que se pueden aplicar a los hijos, junto con descripciones y ejemplos. Son menos que las que se pueden aplicar a los contenedores, así que te será más fácil de acordarte de todo

Quizás resulta obvio, pero no está de más recordar a qué nos referimos con contenedores e ítems. Quedará claro a la vista de esta imagen:



Propiedad flex-grow

La propiedad flex-grow sirve para decir cómo deben crecer los elementos incluidos en el contenedor, es decir, cómo distribuir el espacio entre ellos, haciendo que ocupen más o menos espacio. El valor que acepta es numérico e indica la proporción de espacio que va a ocupar.

A mi me resulta la más útil de todas las propiedades de los ítem, que además es bastante versátil. El ejemplo más típico para poder entender flex-grow lo podemos ver en el siguiente ejemplo:

En esta imagen tenemos un contenedor y 5 ítem.

```
<div class="contenedor">

  <div class="item grow1">1</div>

  <div class="item grow2">2</div>

  <div class="item grow3">3</div>

  <div class="item grow2">2</div>

  <div class="item grow1">1</div>
```

```
</div>
```

El contenedor es por supuesto un flexbox:

```
.contenedor {

  display: flex;

  background-color: #666;

  font-size: 1.5em;

  font-weight: bold;

  text-align: center;

  color: #fff;

}
```

Y los ítem tienen varias clases, a las que les vamos a asociar distintos valores de flex-flow.

```
.grow1 {

  flex-grow: 1;

}
```

```
.grow2{

  flex-grow: 2;

}

.grow3{

  flex-grow: 3;

}
```

Pues como se puede ver en la imagen, los elementos tendrían dimensiones distintas, atendiendo a los valores de flex-grow. Automáticamente le pone unas anchuras que tienen la proporción indicada por el valor numérico de flex-flow.

- Los elementos con flex-grow: 2 ocupan el doble de espacio que los elementos con flex-grow: 1.
- El elemento con flex-grow: 3 ocupa el triple de espacio que los elementos con flex-grow: 1.

Las aplicaciones podrían ser por ejemplo que todos tuvieran la misma anchura, en cuyo caso colocaríamos en todos el mismo valor de flex-grow, pero hay un caso que me parece muy útil que es el que se puede ver en la siguiente imagen:

En este caso concreto tenemos varios ítem y hay uno de ellos que queremos que ocupe todo el espacio disponible sobrante.

Para conseguir ésto simplemente tenemos que aplicarle el flex-grow: 1 al elemento que queremos que ocupe el resto del espacio.

```
<div class="contenedor">

  <div class="item">Item</div>

  <div class="item">Item</div>

  <div class="item">Item</div>

  <div class="item grow1">1</div>

</div>
```

Recuerda que la clase "grow1" tenía el valor de flex-grow: 1. Pero en la práctica podría haber tenido cualquier otro valor en flex-grow, pues aquí la clave es que solamente esa etiqueta tiene definido un flex-grow, de modo que esa será la que se tomará todo el espacio disponible sobrante.

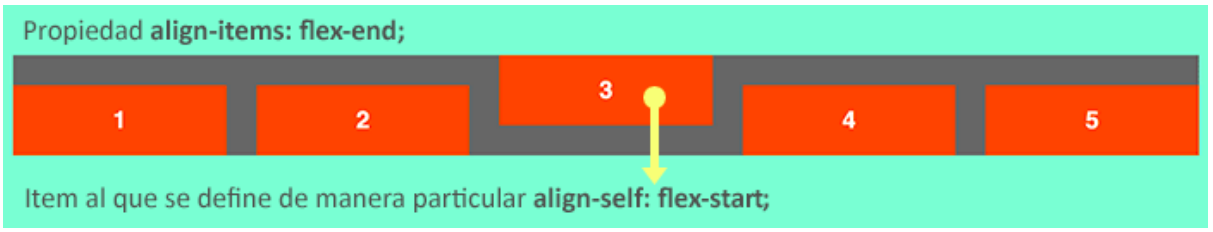
Propiedad order

No necesita muchas explicaciones. Admite un valor numérico entero y sirve para aplicar un orden puramente arbitrario en la disposición de los elementos.

Propiedad align-self

Esta propiedad sirve para modificar el valor de align-items marcado por el contenedor principal. Como ya sabes por el artículo anterior de Propiedades Flex para Contenedores, align-items nos permite definir el alineamiento en el eje secundario del contenedor (lo que se traduce por el alineamiento vertical en el caso de elementos que se posicionan en una fila. Con align-items podemos definir el alineamiento de todos los elementos a la vez y sin embargo con align-self podemos sobreescribirlo para un ítem concreto.

Lo puedes seguramente entender mejor con la siguiente imagen.



Propiedad flex-shrink

Sirve para indicar que ciertos ítems deben encoger su tamaño. El valor predeterminado de flex-shrink es de 1. Cualquier valor superior indica que ese elemento se encogerá con respecto a lo que ocuparía si no tuviera esa propiedad. A mayor valor de flex-shrink, más reducido será el tamaño resultante del elemento.

Propiedad flex-basis

Esta propiedad sirve para modificar las dimensiones de los elementos atendiendo a varias posibilidades. La propiedad sobre el papel sirve para definir el tamaño predeterminado de un elemento, pero antes de que el espacio sobrante sea distribuido, cuando proceda, por causa de otras propiedades como flex-grow.

Los valores que soporta son los siguientes:

- Numéro, unidad CSS o porcentaje: lo que indica las dimensiones iniciales del elemento, antes de otorgar espacio sobrante.
- auto: es el valor predeterminado e indica que flex-basis no va a tener efecto, otorgando dimensionamiento en función de cualquier otro atributo que pueda haber en el elemento, o en función del contenido del propio elemento.

Nota: flex-bases es una propiedad menos obvia que otras conocidas anteriormente. Sabemos que las dimensiones de los elementos se pueden modificar directamente con width y height, que también tenemos flex-grow para otorgar más espacio y flex-shrink para encoger los elementos. Quizás pensarás que cuesta encajar este nuevo atributo, pero la clave es que permite indicar unas veces la anchura y otras la altura, en función del eje principal de los elementos flexbox definido en el contenedor. Cuando el eje está en la horizontal, flex-basis aplica a la anchura de los elementos. Cuando el eje está en la vertical, entonces flex-basis aplica a la altura de los elementos.

Propiedad flex

Esta no agrega nada nuevo. Es solo un atajo para escribir en una sola línea de código CSS las propiedades flex-grow, flex-shrink y flex-basis. El valor por defecto de esta propiedad es "0 1 auto".