FLEXBOX

Vamos a continuar el [Manual de Flexbox](https://desarrolloweb.com/manuales/css3.html#manual203) abordando un nuevo tema básico como son las propiedades disponibles en los contenedores "flex". Veremos cómo podemos conseguir que un contenedor se comporte con las reglas de Flexbox y cómo configurarlo para personalizar su comportamiento.

Aunque ya lo vimos en el artículo anterior, con [conceptos básicos de Flexbox](https://desarrolloweb.com/articulos/conceptos-principales-flexbox.html), recordamos que el contenedor flex, o contenedor principal, es aquel en el que vamos a situar los flex items. Si tienes alguna duda sobre esto te recomendamos estudiar el mencionado artículo, porque ahí encontrarás explicado con detenimiento éste y otros conceptos en los que vamos a apoyarnos..



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:

https://desarrolloweb.com/archivoimg/general/4189.png

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"](https://desarrolloweb.com/articulos/estilos-css-rwd.html).

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;

}

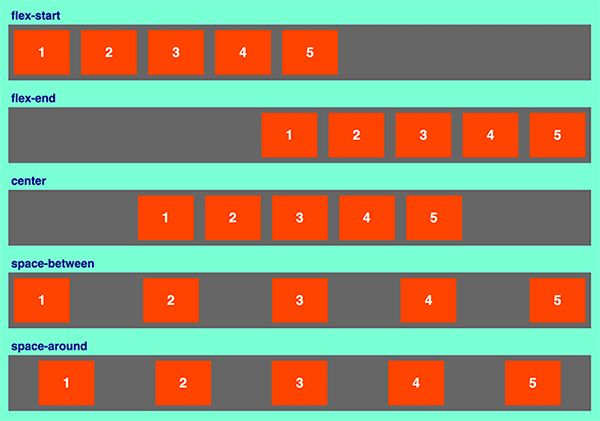
Propiedad justify-content

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:



En próximos artículos abordaremos [más prácticas de uso de justify-content](https://desarrolloweb.com/articulos/practica-justifycontent-flexbox.html).

Propiedad align-items

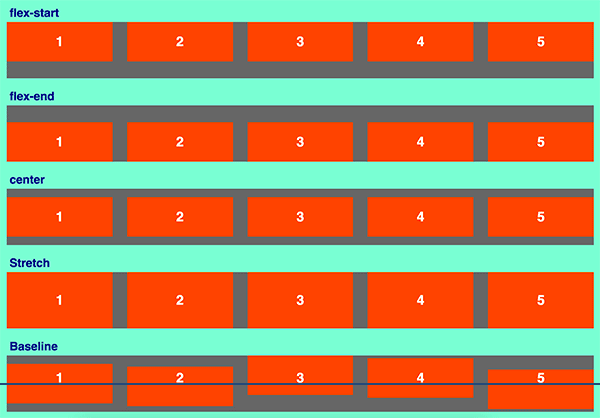
Esta propiedad es muy similar a la propiedad anterior, justify-content, solo que ahora estamos alineando con respecto al eje secundario y no el principal.

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

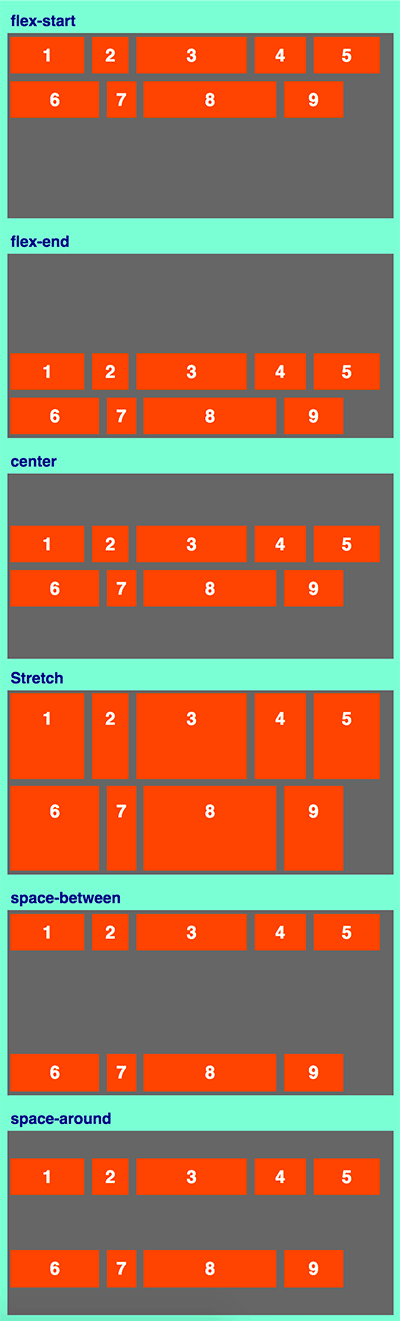
Esta propiedad sólo aplica cuando dispones de varias líneas de elementos en el contenedor flexbox. El efecto que conseguiremos será una alineación y separación de las filas en el eje secundario.

**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.

Creo que se entenderá mejor a la vista de la siguiente imagen.



Conclusión

Con esto hemos visto ya una buena cantidad de la teoría que tienes que aprender cuando vayas a trabajar con Flexbox, y de las posibilidades de este estándar. Sin duda habrás apreciado que permite mucho juego y variantes muy interesantes, que resultarán muy prácticas en muchas ocasiones del día a día.

En el siguiente artículo veremos propiedades que afectan a los ítems, en vez del contenedor, al que hemos dedicado todo este artículo. Además en adelante también podrás encontrar artículos diversos donde podremos practicar con casos de maquetación más prácticos de la vida real.