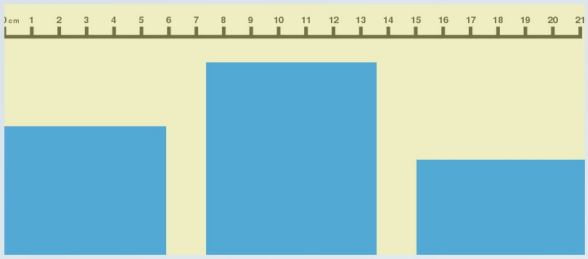
Manual de CSS Flexbox



Diana Aceves
Miguel Angel Alvarez

desarrolloweb.com
desarrolloweb.com/manuales/manual-flexbox-css.html

Introducción: Manual de Flexbox CSS

En este manual de Flexbox vamos a tratar con detalle una de las herramientas clave de CSS para poder realizar una maquetación de contenidos detallada y versátil: Flexbox.

Flexbox son un conjunto de nuevos atributos (propiedades) y valores de CSS que podemos aplicar a los elementos de una página y que permiten posicionarlos de maneras diferentes a las disponibles anteriormente. No quiere decir que antes no se pudiera hacer las cosas que hoy Flexbox te permite, sino que para conseguirlas tenías que usar varias técnicas en conjunto y que ahora obtienes de una manera muy sencilla.

En resumen, con flexbox podrás hacer casi inmediatamente cosas que con CSS tradicional son muy difíciles de conseguir, lo que te ahorrará tiempo y dolores de cabeza. Eso sí, requiere un estudio detallado y cambiar la manera de pensar a la hora de realizar las interfaces de usuario o maquetar webs completas.

En el manual de Flexbox podrás aprender qué es este potente estándar y cómo aplicarlo en ejemplos reales del diseño web.

Encuentras este manual online en:
http://desarrolloweb.com/manuales/manual-flexbox-css.html

Autores del manual

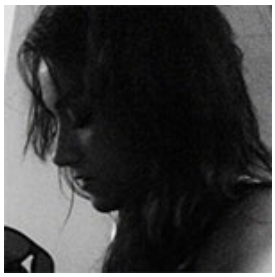
Las siguientes personas han participado como autores escribiendo artículos de este manual.

Miguel Angel Alvarez

Miguel es fundador de DesarrolloWeb.com y la plataforma de formación online EscuelaIT. Comenzó en el mundo del desarrollo web en el año 1997, transformando su hobby en su trabajo.

Diana Aceves

Licenciada en química, Diana comenzó con la programación con Java y finalmente su vocación la llevó a convertirse en desarrolladora front-end



CSS3 Flexbox

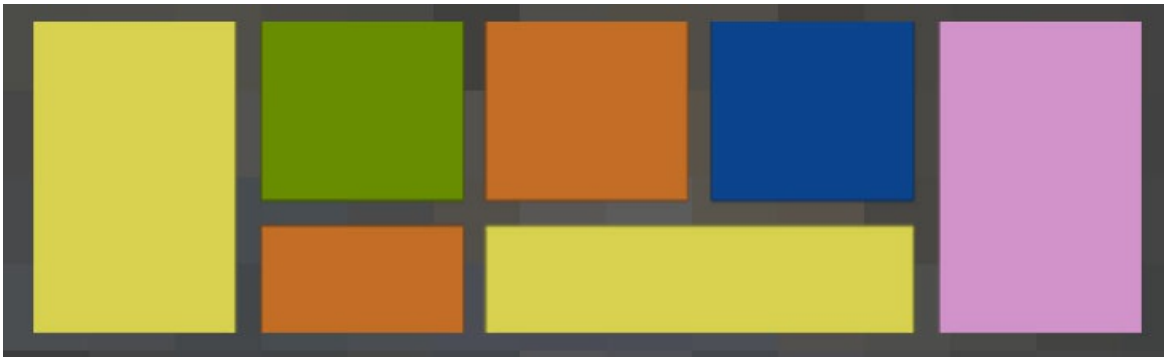
Qué es Flexbox y por qué tienes que comenzar a usar este potente mecanismo de CSS3 para componer layouts de una manera rápida, sencilla y poderosa.

Este es un artículo que nos sirve de introducción a Flexbox. Nos vamos a quedar en un conocimiento un tanto teórico, pero al final de este texto encontrarás un vídeo con poco más de dos horas, donde también lo conocerás por la práctica.

Antes de comenzar con Flexbox queremos que nos respondas una pregunta ¿Cuántas veces has sentido que CSS no era suficiente para resolver las necesidades de un proyecto? o quizás, ¿Cuántas veces has tenido que modificar el HTML para hacer posible la maquetación de unos elementos de una manera determinada? O peor, recurrir a Javascript para conseguir que se coloquen como deseas.

Si llevas un tiempo en el mundo del desarrollo para la web habrás observado que, a pesar que CSS ofrece muchas características para maquetación de contenidos y creación de "layouts", determinados comportamientos eran difíciles de producir y te obligaban a ingeniártelas de diversos modos para conseguir tus objetivos.

En resumen, si lo que necesitas es "clavar un diseño", para que se vea muy bien, tenías que sufrir con las herramientas disponibles hasta ahora. Esas y otras cuestiones se pretenden solucionar con Flexbox.



Qué es Flexbox

Flexbox es un módulo completo de layout disponible en la especificación de CSS3. Define cómo se muestran los elementos y cómo se relacionan con el resto. Como concepto, puedes entender Flexbox como un modelo para la creación de layouts, que pretende mejorar los anteriores, aunque sin ser excluyente. Todas las técnicas disponibles antes de Flexbox tenían diversos problemas y limitaciones que se pretenden solucionar con esta especificación de CSS3. Flexbox es una herramienta muy avanzada para poder crear layouts de características avanzadas y necesarias en el día de hoy, donde es tan importante una estética cuidada y una gran adaptabilidad a distintos formatos de pantalla.

Nota: hasta la fecha hemos maquetado con técnicas como float o display table, pero estos mecanismos no fueron pensados para utilizarse como los utilizamos. Los diseñadores los usaban y conseguían sus objetivos, muchas veces usando lo que se concen como "hacks css". Flexbox sí es un estándar pensado para hacer layouts y por tanto soluciona la mayoría de las necesidades de los desarrolladores sin tener que emplear técnicas rebuscadas.

En la práctica, Flexbox agrega un nuevo tipo de "display CSS", con una completa gama de nuevas propiedades aplicables a ese tipo de display, a partir de los que puedes conseguir cosas extraordinarias. Por aclararnos, igual que tienes en CSS el display "block", "inline", "inline-block", etc. y sabías todos las propiedades que te acepta ese tipo de elementos, ahora dispones de "flex" e "inline-flex", siendo que los elementos que tienen ese nuevo display aceptan una cantidad enorme de nuevas propiedades de gran utilidad.

En resumen, lo que antes tenías que conseguir con una docena de reglas y estilos CSS, cálculos, etc. ahora lo vas a poder implementar mucho más fácilmente, a veces incluso con una única propiedad.

En Flexbox diferenciamos dos elementos principales: la caja contenedora y los elementos que situamos dentro. Al aplicar un display flex o display inline-flex hacemos que una caja se comporte mediante este nuevo estándar y eso produce que los elementos que tiene como contenido se puedan distribuir con las propiedades de este estándar de maquetación.

El contenedor va a poder modificar las dimensiones y el orden de los items, para acomodarlos de distintas maneras controladas por el desarrollador. Podremos repartir el espacio entre ellos de diversas formas, para distribuirlo a nuestro antojo, permitir que los items se estiren para ocupar todo el contenedor, o se encojan para que quepan en él sin desbordar, de colocarse distribuidos en filas o en columnas, etc.

Nota: para aclarar posibles dudas tenemos que advertir que Flexbox es una especificación de CSS3. No se trata de una librería de estilos como podría ser Bootstrap, sino de un estándar de la web. No necesitas instalar nada para que lo entiendan los navegadores, igual como tampoco necesitas instalar nada para para trabajar con cualquier característica actual de CSS. Usar Flexbox no es excluyente de usar cualquier cosa que ya estás usando de CSS. Es decir, si decides usar Flexbox en un momento dado, nada te impide seguir utilizando comportamientos anteriores como podría ser float o display table. Simplemente con Flexbox podrás hacer las cosas más rápidamente y llegar donde antes no era sencillo con otros atributos de CSS.

Qué soluciona Flexbox

Flexbox permite que se puedan posicionar elementos de una manera más concisa y distribuir los espacios entre ellos de forma más flexible. Permite gran cantidad de comportamientos, pero este sería un rápido resumen, si nos ceñimos a las cosas que antes eran muy difíciles de hacer mediante CSS anterior:

Alineación vertical:

Seguro que has sudado para conseguir alinear elementos en la vertical. A veces dado por imposible, hoy es algo que es muy sencillo con Flexbox.

Columnas de igual altura:

Conseguir que todos los items de un listado tengan la misma altura, independientemente de su contenido. También difícil de conseguir con CSS anterior, sobre todo cuando el contenido era variable e impredecible. Por supuesto, Flexbox también nos permite elementos con igual anchura, y aunque esto es algo que ya teníamos antes fácilmente, ahora resulta mucho más sencillo.

Cambiar el orden de los elementos:

Sin tener que cambiar el orden de los elementos en el HTML, con Flexbox conseguimos que se ordenen de maneras distintas al visualizarse en la página.

Con esto se consigue que los diseñadores, maquetadores y desarrolladores frontend en general tengan mucho mayor control sobre los elementos en la página y puedan, de una manera más detallada, especificar su apariencia y colocación, limitándose solamente a modificar los atributos CSS.

Flexbox y compatibilidad con navegadores

Flexbox hoy ya es una realidad, puesto que los navegadores modernos lo soportan y lo interpretan correctamente. Por tanto, está listo para usar en cualquier tipo de proyecto.

El único navegador que no lo soporta es Internet Explorer en versiones antiguas, como IE8 o IE9 (Navegadores que a día de hoy no están soportados ni por el propio fabricante). Si es requisito indispensable que tu web se vea igual en estos navegadores, tendrás que usar las técnicas de toda la vida, o fallbacks que enseguida comentamos. Pero antes de ello conviene ver en la siguiente tabla de compatibilidad, extraida de Caniuse:

Nota: ¿Por qué no debes preocuparte por IE8 o IE9? porque tienes cosas más importantes en las que pensar y decidir soportarlos producirá que tu web no pueda usar las herramientas más nuevas disponibles para hacer tu trabajo. La web probablemente acabe siendo peor en calidad y menos adaptada a los tiempos que corren y a ti te lleva mucho más tiempo el desarrollarla. Generalmente a los clientes hay que convencerles que no necesitan dar la mejor imagen para navegadores anticuados y uno de los mejores argumentos es cobrar más por hacer el trabajo. De todos modos, usar Flexbox no significa que el contenido no va a estar ahí. Simplemente que no va a lucir de la mejor manera. Hay

técnicas, englobadas en el conocimiento de Resposive Web Design, que nos ofrecen la vía para que una web se vea correctamente en todos los navegadores y, a medida que el navegador es más avanzado, luzca mejor porque se usen nuevas y más potentes características. Si quieres profundizar lee el artículo Sobre Progressive Enhancement Vs Graceful Degradation.

Realmente IE8 o IE9 no son tan importantes en la actualidad, pues un pequeño porcentaje de usuarios (menor del 1% cada) lo usa. Lo que sí es más importante son las versiones antiguas de móviles, con Android 4.3 o anteriores. Esas versiones tienen navegadores antiguos que daban un soporte parcial a Flexbox, donde estamos obligados a usar los prefijos de las propiedades CSS ("vendor prefixes"). Por eso te vendría bien contar con algún tipo de preprocesador, postCSS y autoprefixer.

Aunque no suele ser la mejor solución, existen fallbacks o polyfills que aportan un soporte parcial a Flexbox, instalando una librería Javascript adicional. Son una opción cuando necesitamos soportar Flexbox en algunos navegadores antiguos, aunque también hay que advertir que tendrá un coste en términos de rendimiento / peso, y quizás no todo se vea exactamente igual que en navegadores con soporte nativo. Un ejemplo de polyfill lo encuentras en Flexibility.

Conoce Flexbox en la primera clase del Taller Profesional de Flexbox

Si quieres aprender más y además ver cómo se comportan las propiedades principales de Flexbox en una serie de ejemplos prácticos, te recomendamos continuar viendo el siguiente vídeo.

Es una clase impartida en octubre de 2016, la primera del Taller de Flexbox de EscuelaIT. Es un taller en el que se va abordar el desarrollo con Flexbox por la práctica y con ejemplos reales.

Para ver este vídeo es necesario visitar el artículo original en: http://desarrolloweb.com/articulos/css3-flexbox.html

Este artículo es obra de *Diana Aceves*
Fue publicado por primera vez en *14/10/2016*
Disponible online en *http://desarrolloweb.com/articulos/css3-flexbox.html*

Entender los conceptos principales de Flexbox

Establecemos algunos conceptos de Flexbox que debes tener en cuenta para trabajar con soltura con este estándar y veremos una primera práctica ilustradora de algunas de sus posibilidades.

En este segundo artículo del Manual de Flexbox de DesarrolloWeb.com queremos seguir aclarando conceptos y comenzar con una pequeña práctica donde empecemos a experiementar con algunas de las propiedades disponibles.

Comenzaremos con algunas informaciones un tanto teóricas, pero importantes de saber antes de empezar a trabajar con Flexbox. Son cosas que nos vendrá bien tener claras y así en adelante poder ir más rápido en nuestro aprendizaje y el uso de Flexbox en el día a día.

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.

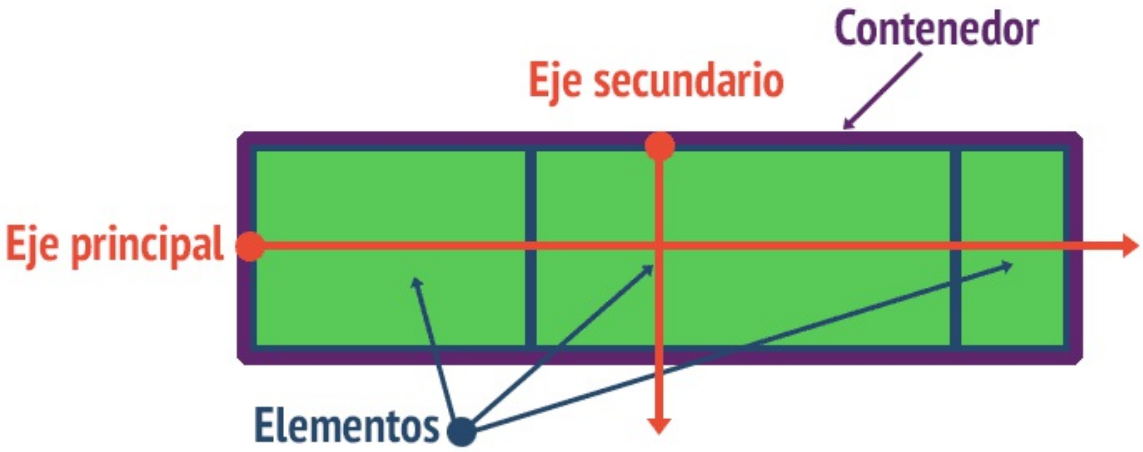
Más adelante detallaremos todo esto, pero puedes ir entendiendo que habrá propiedades CSS pensadas para alterar el modo en el que un contenedor se va a comportar y otras propiedades que sirven para indicar cómo se deben representar sus ítems internos. Por ejemplo, en el contenedor principal podremos decir que los elementos los queremos en la horizontal, o en la vertical. Mientras que en un item interno podremos indicar cosas como el tamaño que deben ocupar en relación a otros o el orden en el que deben aparecer.

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

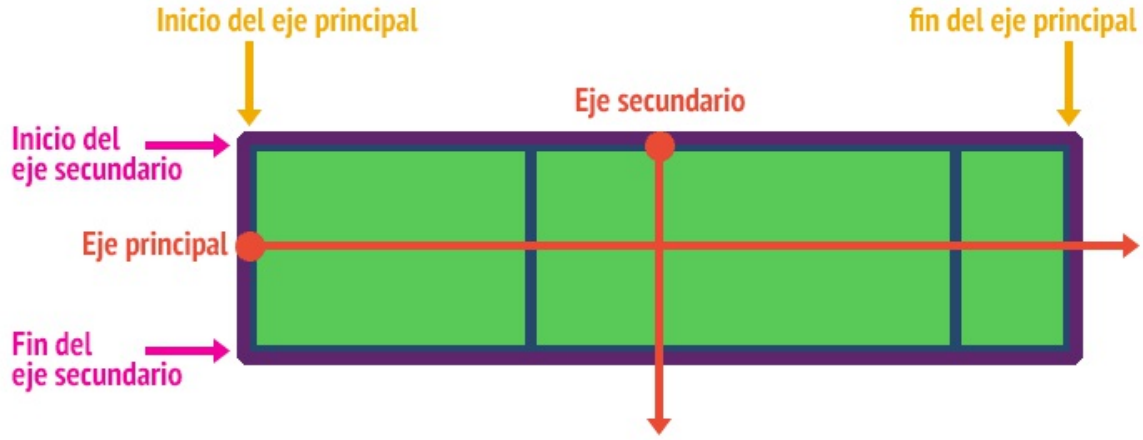
configurable, pero para que nos hagamos una idea con un ejemplo, si el eje principal es la horizontal, los ítems se pondrán uno al lado del otro. Si el eje principal fuera la vertical, los ítem se colocarían uno debajo (o arriba) del otro.



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.

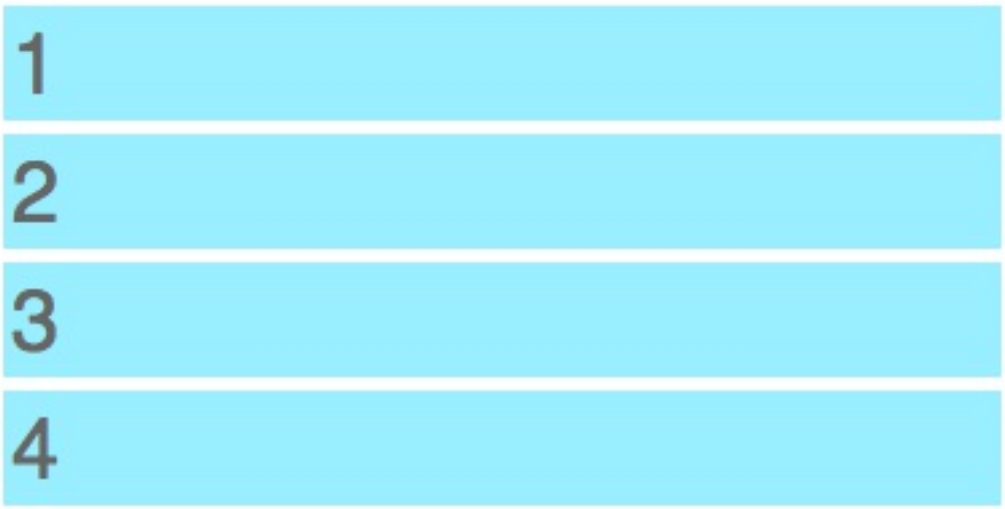
Vamos a partir de un HTML sencillo y aplicaremos varios cambios con Flexbox para ver la transformación a la hora de representarse en la página.

```
<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.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *18/04/2017*
Disponible online en *http://desarrolloweb.com/articulos/conceptos-principales-flexbox.html*

Propiedades para el contenedor Flexbox

Flexbox CSS: propiedades asignables a contenedores con display flex, con explicaciones y ejemplos de uso.

Vamos a continuar el Manual de Flexbox 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, 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:

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;
}
```

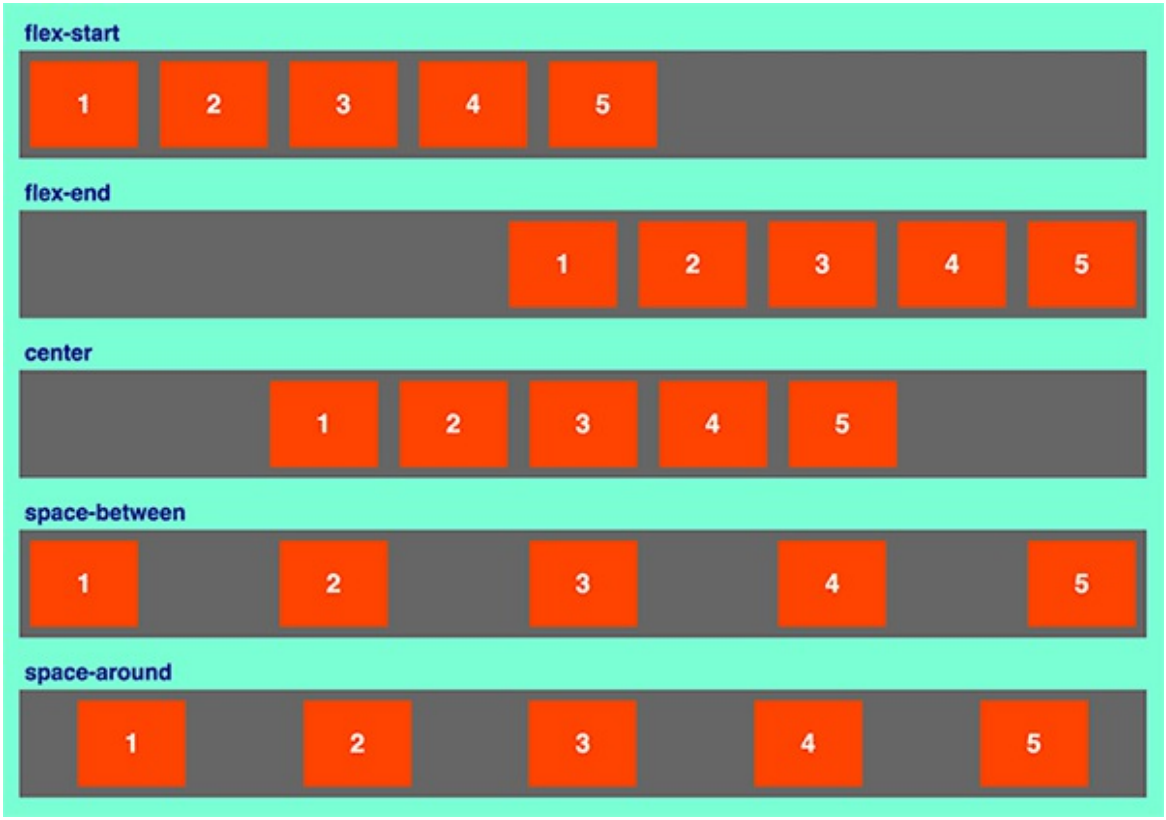
Propiedad justify-content

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:



En próximos artículos abordaremos más prácticas de uso de justify-content.

Propiedad align-items

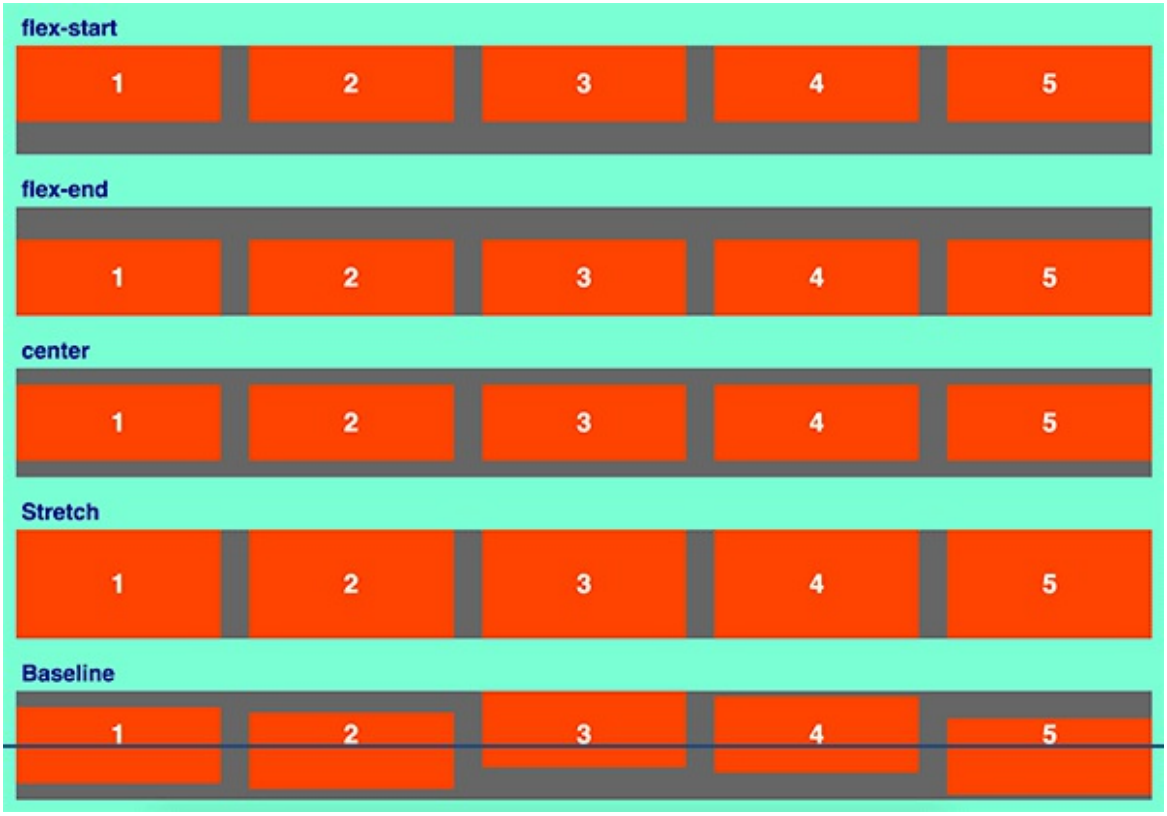
Esta propiedad es muy similar a la propiedad anterior, justify-content, solo que ahora estamos alineando con respecto al eje secundario y no el principal.

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

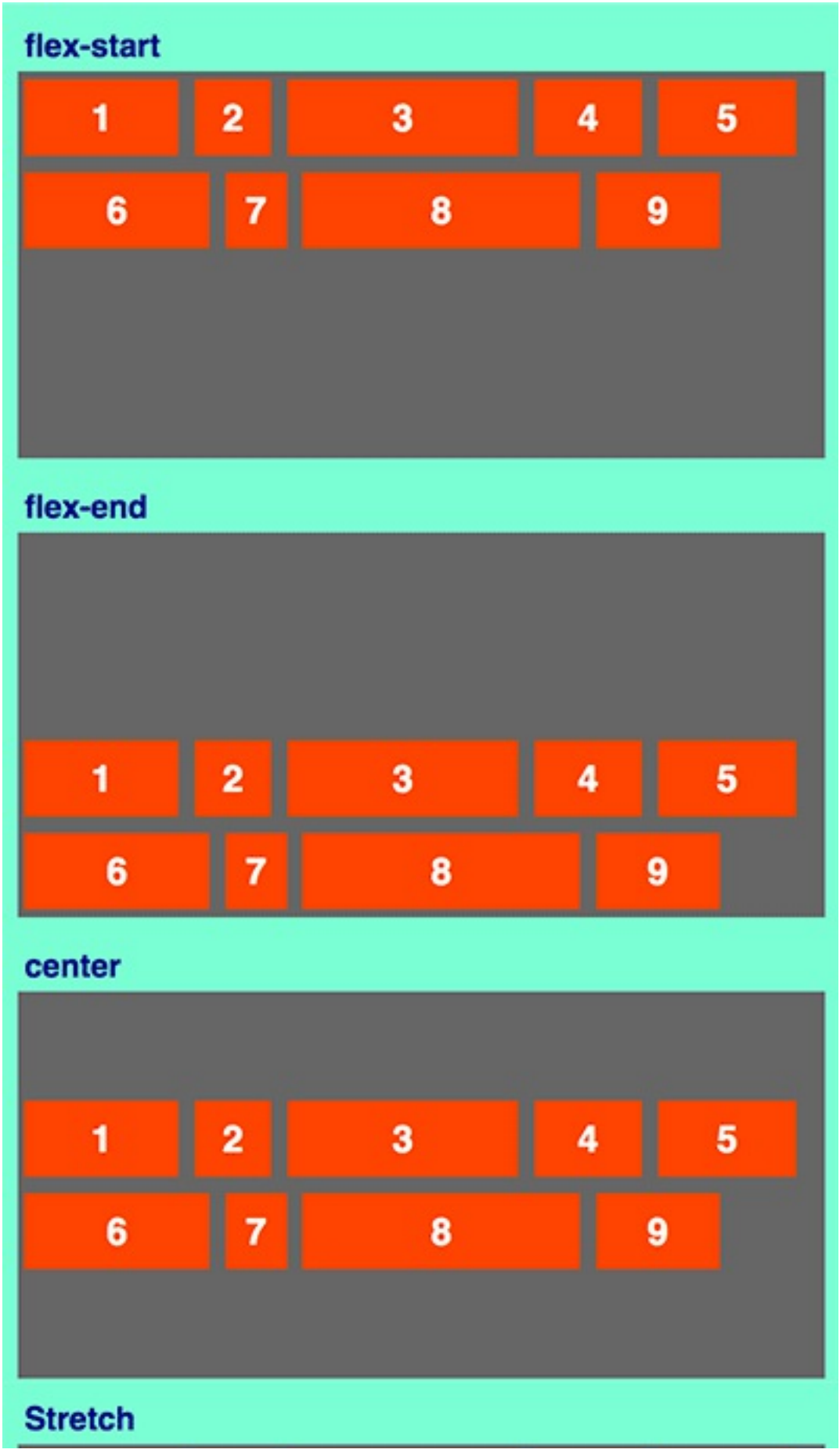
Esta propiedad sólo aplica cuando dispones de varias líneas de elementos en el contenedor flexbox. El efecto que conseguiremos será una alineación y separación de las filas en el eje secundario.

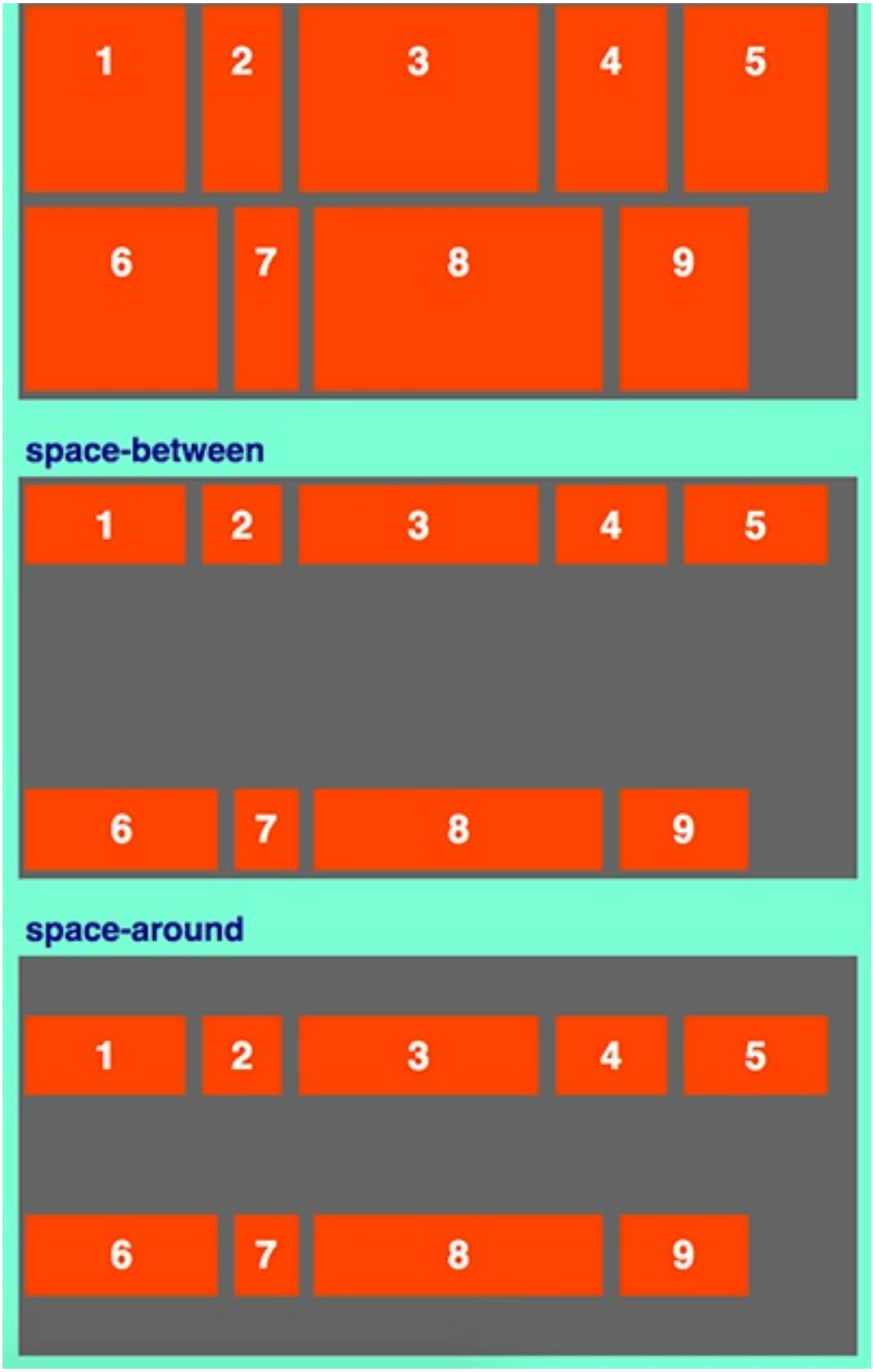
> **Nota:** Para conseguir varias líneas de elementos en el contenedor flex necesitarás aplicarles un "flex-flow: wrap" y por supuesto que haya suficientes elementos, con suficiente anchura, para que se necesiten varias filas para su distribución.

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.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *26/04/2017*
Disponible online en *http://desarrolloweb.com/articulos/propiedades-contenedor-flexbox.html*

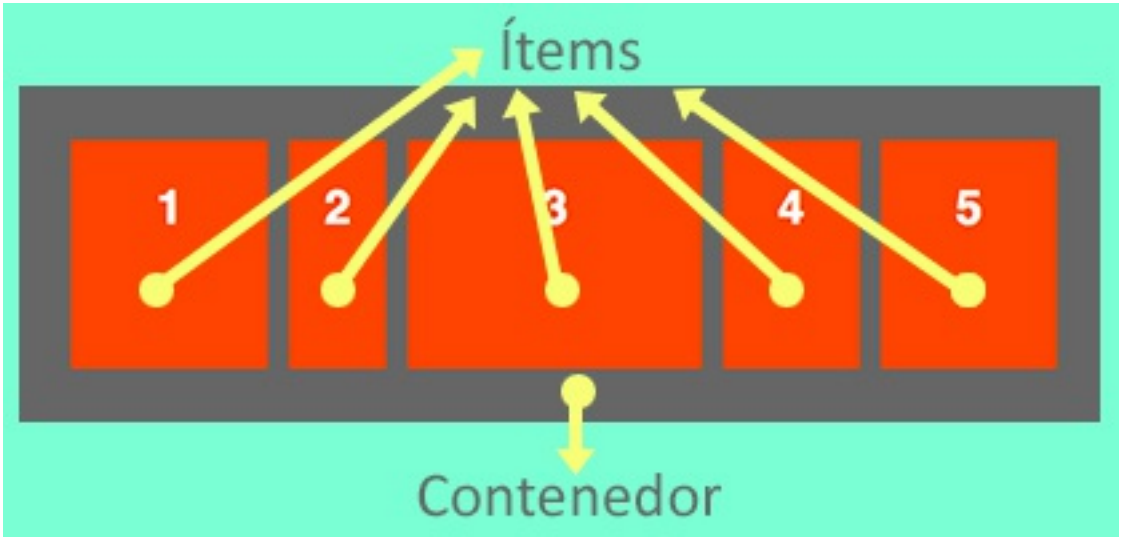
Propiedades de los ítem flexbox

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.

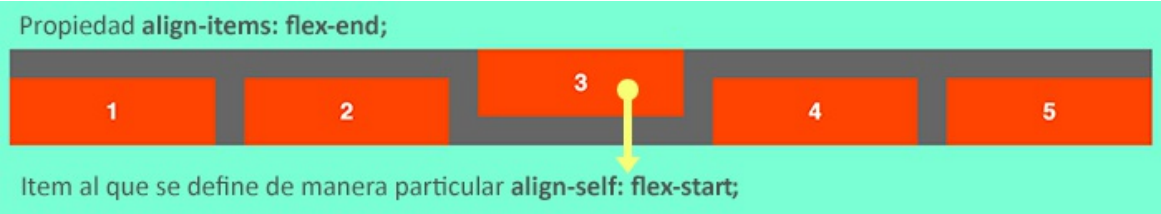
Propiedad order

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".

Conclusión

Con este resumen hemos conocido ya todas las propiedades que afectan a los ítems, que junto con las propiedades que afectan a los contenedores repasadas en la anterior entrega del manual, componen todo el material que debes conocer del estándar CSS Flexbox.

Ahora nos queda practicar bastante, revisando ejemplos completos en las siguientes entregas, con los que

dominar Flexbox con casos reales y útiles en el día a día.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *08/05/2017*
Disponible online en *http://desarrolloweb.com/articulos/propiedades-item-flexbox.html*

Práctica con justify-content en Flexbox

Veamos un ejemplo práctico de uso de la propiedad justify-content en Flexbox en el que experimentaremos con distintos valores posibles.

Mediante CSS3 Flexbox se hacen fáciles las cosas que antes eran mucho más complicadas. Con una sola propiedad como justify-content podemos hacer algo que, si bien no era tan difícil, sí que requería bastante trabajo, combinando diversas propiedades que había que montar en conjunto.

En este artículo observaremos su utilidad y podremos ver cómo con distintos valores generamos diversos comportamientos interesantes y útiles para maquetar sobre todo listas de ítems que se comporten de manera flexible ante diversas anchuras del elemento padre.

Recuerda que justify-content es una de las propiedades asignables a elementos Flexbox, de las que ya analizamos en el artículo sobre las propiedades Flexbox para el contenedor. En el pasado artículo ya ofrecimos una lista de posibles valores de justify-content, por lo que pasaremos directamente a la práctica.

Maquetar una lista de tarjetas

En diseño web habitualmente tienes listas de ítems que se deben organizar de una manera flexible, adaptándose a las pantallas de cualquier ordenador o dispositivo. Esos ítems pueden ser un elemento simple o tener a su vez diversos hijos para ofrecer varias informaciones e interacción con el usuario. Podrían ser fotos de un álbum, productos de una tienda, propiedades de una inmobiliaria, etc. A la hora de maquetarlos, usar flexbox representa mucha comodidad.

Tengo este código, con un contenedor y en ellos una serie de elementos ARTICLE, en los cuales hemos colocado únicamente una imagen (Podría haber un texto, una imagen, unos enlaces y otras cosas para cada uno de los article, en nuestro caso hay solo una imagen por simplificar, pero cada una de los item podría ser todo lo complejo que se quiera).

```
<div class="contenedor">
  <article class="item">
```

```
    <img src="img/1.png">
  </article>
  <article class="item">
    <img src="img/2.png">
  </article>
  <article class="item">
    <img src="img/3.png">
  </article>
  <article class="item">
    <img src="img/4.png">
  </article>
  <article class="item">
    <img src="img/5.png">
  </article>
  <article class="item">
    <img src="img/6.png">
  </article>
</div>
```

Si deseo que se muestren en un listado y que se adapten a las dimensiones de la pantalla, usar Flexbox es muy ágil.

> **Nota:** Con que "se adapten a la pantalla" me refiero a que, si tengo más espacio se pongan uno al lado del otro, y si no tengo espacio, uno debajo del otro.

Para aplicar Flexbox a este ejemplo tendríamos que comenzar por definir que el contenedor va a ser "display flex".

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

Display flex hace simplemente que sus elementos aparezcan uno al lado del otro, en una fila, que es el comportamiento predeterminado. Si no hay espacio para todos los elementos el navegador los colocará igualmente en la misma fila, pero lo que queremos es que, si no hay espacio, que los mande para la fila de abajo. Por ello tenemos que agregarle "flex-wrap: wrap".

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

Ahora bien, los elementos se colocarán en la fila, comenzando por el principio de la fila, que es el comportamiento predeterminado, pero igual queremos que se coloquen en el centro, dejando espacio a los lados, o a los extremos, dejando el espacio en el medio. Ese comportamiento es el que vamos a definir con

justify-content, cómo se van a justificar los elementos al situarse en la fila.

En la maquetación de este estilo de tarjetas que hemos señalado (resúmenes de productos de una categoría, fotos de un álbum, etc.) lo más normal es que queramos usar dos valores muy recurrentes: space-between y space-around. Te recomiendo que pruebes los dos y veas sus diferencias.

```
.contenedor {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
}
```

Y la otra alternativa…

```
.contenedor {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-around;
}
```

Con esta declaración en los estilos del contenedor ya es suficiente para que los item se adapten a diversos tamaños de pantalla. Pero igual necesitas colocar algo de CSS a los ítem, al menos para que no se peguen mucho los unos a los otros cuando el espacio comience a escasear por la pantalla estrecharse.

```
.item {
  margin: 10px;
}
```

En el siguiente vídeo tienes una muestra sobre cómo se comportaría esta maquetación con las dos alternativas que hemos señalado de justify-content:

Para ver este vídeo es necesario visitar el artículo original en:
http://desarrolloweb.com/articulos/practica-justifycontent-flexbox.html

Para el escaso CSS que hemos usado flexbox ya nos ofrece una solución bastante depurada y atractiva.

Test de los posibles valores de la propiedad justify-content

El ejercicio anterior ya de por sí resulta muy útil, pero no es la única posibilidad de justify-content.

Ahora voy a dejar un código HTML y su CSS que tengo para experimentar las posibilidades de esta propiedad de contenedores flexbox, aplicando sus distintos valores.

Observarás que tenemos un CSS básico y luego una serie de clases que se especializan para aportar distintos

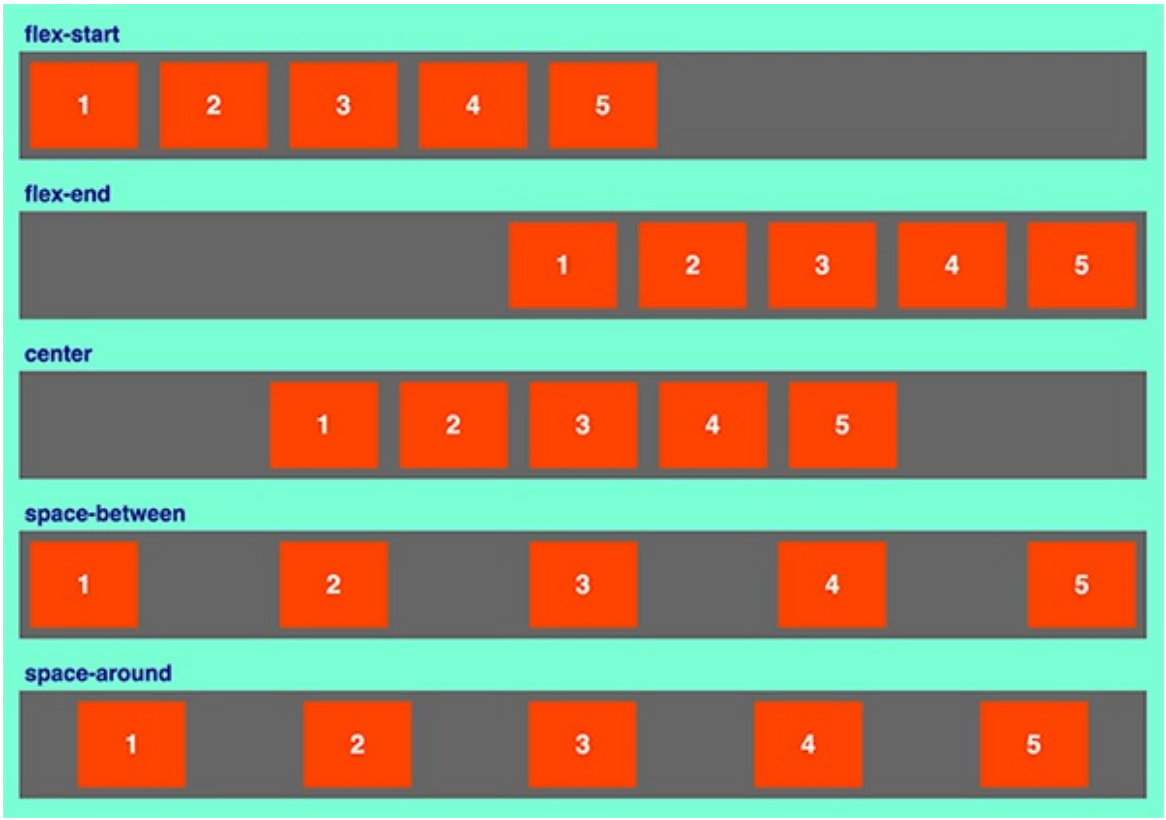
valores de justify-content. En el HTML luego tenemos una serie de listas (contenedor e ítems) con el mismo código, salvo que a cada alternativa le hemos aplicado una clase CSS distinta.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Test de justify-content</title>
  <style>
    body {
      background-color: aquamarine;
      font-family: sans-serif;
    }
    ul, li {
      margin: 10px;
      padding: 0;
      list-style: none;
      display: flex;
    }
    ul {
      margin-top: 0;
      background-color: #666;
    }
    li {
      display: block;
      width: 100px;
      height: 80px;
      background-color: orangered;
      line-height: 80px;
      font-size: 1.5em;
      font-weight: bold;
      text-align: center;
      color: #fff;
    }
    h1 {
      margin: 20px 20px 4px 15px;
      font-size: 1.3em;
      color: darkblue;
    }
    .flexstart {
      justify-content: flex-start;
    }
    .flexend {
      justify-content: flex-end;
    }
    .flexcenter {
      justify-content: center;
    }
    .between {
      justify-content: space-between;
    }
    .around {
```

```
      justify-content: space-around;
    }
  </style>
</head>
<body>
  <div class="test">
    <h1>flex-start</h1>
    <ul class="flexstart">
      <li>1</li>
      <li>2</li>
      <li>3</li>
      <li>4</li>
      <li>5</li>
    </ul>
  </div>
  <div class="test">
    <h1>flex-end</h1>
    <ul class="flexend">
      <li>1</li>
      <li>2</li>
      <li>3</li>
      <li>4</li>
      <li>5</li>
    </ul>
  </div>
  <div class="test">
    <h1>center</h1>
    <ul class="flexcenter">
      <li>1</li>
      <li>2</li>
      <li>3</li>
      <li>4</li>
      <li>5</li>
    </ul>
  </div>
  <div class="test">
    <h1>space-between</h1>
    <ul class="between">
      <li>1</li>
      <li>2</li>
      <li>3</li>
      <li>4</li>
      <li>5</li>
    </ul>
  </div>
  <div class="test">
    <h1>space-around</h1>
    <ul class="around">
      <li>1</li>
      <li>2</li>
      <li>3</li>
      <li>4</li>
      <li>5</li>
    </ul>
  </div>
```

```
</body>
</html>
```

Al visualizar este ejemplo en el navegador obtendrás un resultado como el de la imagen siguiente, demo de las alternativas posibles de justify-content.



Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *23/05/2017*
Disponible online en *http://desarrolloweb.com/articulos/practica-justifycontent-flexbox.html*

Ejercicios flexbox con align-items

Práctica con la propiedad align-items de los elementos con display flex, una de las principales posibilidades del modelo de maquetación flexbox css.

En el Manual de Flexbox hemos abordado diversas informaciones básicas sobre este modelo de posicionamiento CSS. Sin embargo nos quedan pendientes algunas prácticas para facilitar asimilar los conceptos explicados.

En este artículo vamos a practicar con otra de las principales posibilidades de Flexbox, que es la definida por el atributo align-items, que ofrece bastantes variantes para el posicionamiento de elementos.

Lo primero que se debe aclarar es que **align-items es uno de los atributos que modifican el comportamiento de los items**, pero que se **aplica sobre los contenedores flexbox**. Recuerda que el contenedor es el elemento que tiene "display: flex" o "display: inline-flex" y que los ítem son sus hijos directos. El atributo que nos ocupa, align-items, se aplica sobre los contendores flex, como se describió en el artículo Propiedades para el contenedor Flexbox.

Alineación vertical / horizontal de los ítem

Como ya se describió, el valor de **align-items afecta al eje contrario de disposición de los elementos flex**. Es decir, si los elementos se colocan en una fila (en la horizontal), align-items define el comportamiento en la vertical. Si los elementos se colocan en columnas (la vertical), align-items afecta a su posición en la horizontal.

Para explicar ésto vamos a estudiar cómo se comporta align-items dependiendo de si los elementos flex se distribuyen en filas o en columnas.

Elementos dispuestos en la fila

En el caso de un contenedor flexbox donde la dirección de los elementos es la fila, align-items afecta a su posicionamiento en la vertical. Por ejemplo tenemos un contenedor flex con este CSS:

```
ul {
  display: flex;
  flex-direction: row;
  justify-content: space-between;
  align-items: flex-end;
}
```

El atributo align-items afectará a la vertical, indicando que los elementos se coloquen en el final de la vertical. Quedarían más o menos así.

Como puedes ver, todos los elementos se ajustan abajo del todo, quedando alineados al final del eje secundario del contenedor flexbox.

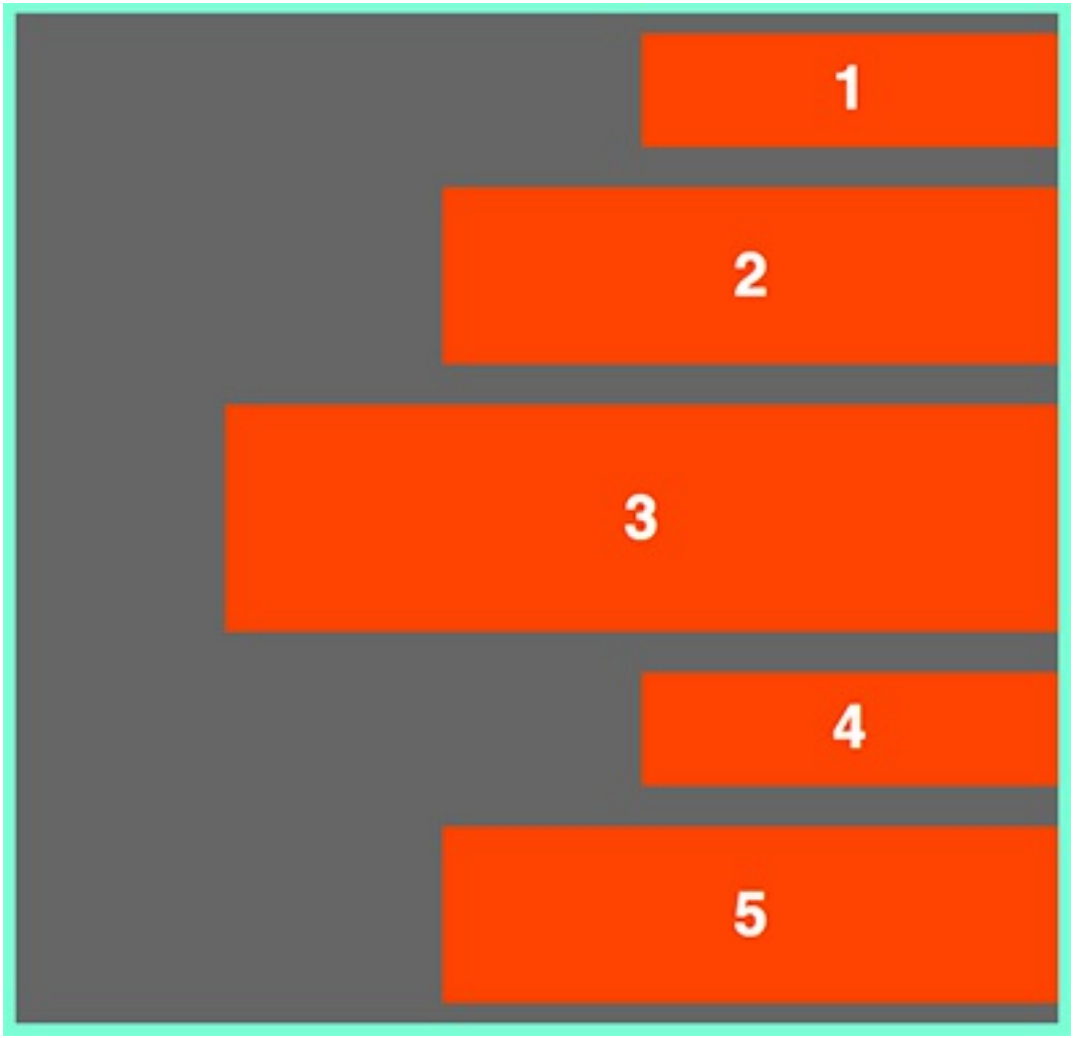
Elementos dispuestos en columnas

En el caso que los elementos se dispongan en columnas, el valor que apliquemos a align-items se referirá a la horizontal, el eje secundario. Por tanto, un mismo valor flex-end, provocará que todos los elementos se coloquen a la derecha. Veamos este CSS:

```
ul {
  display: flex;
  flex-direction: column;
  justify-content: space-between;
  align-items: flex-end;
}
```

Fíjate que con respecto al CSS anterior, lo único que se ha cambiado es flex-direction: column; usado para indicar que los ítem se coloquen uno debajo del otro.

Ahora puedes ver la disposición en la que se colocarían los elementos.



Ejercicio con todos los posibles valores de align-items

Entendido el efecto de este atributo, vamos a ver el código del ejercicio que podemos utilizar para experimentar con todos los valores de align-items.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>align-items</title>
  <style>
    body {
      background-color: aquamarine;
      font-family: sans-serif;
    }
    ul, li {
      padding: 0;
      list-style: none;
    }
    ul {
```

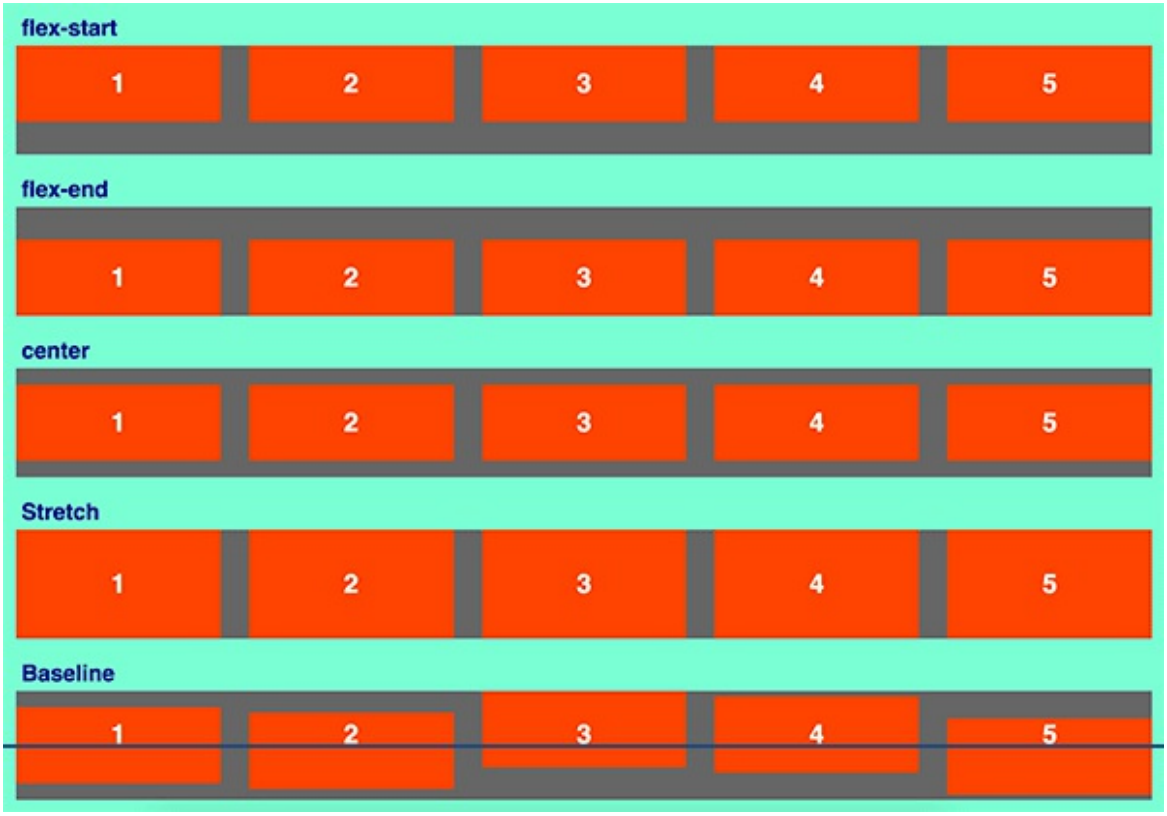
```
      display: flex;
      margin: 10px;
      margin-top: 0;
      background-color: #666;
      justify-content: space-between;
      height: 100px;
    }
    li {
      display: block;
      width: 18%;
      height: 70px;
      background-color: orangered;
      line-height: 70px;
      font-size: 1.5em;
      font-weight: bold;
      text-align: center;
      color: #fff;

    }
    h1 {
      margin: 20px 20px 4px 15px;
      font-size: 1.3em;
      color: darkblue;
    }
    .flexstart {
      align-items: flex-start;
    }
    .flexend {
      align-items: flex-end;
    }
    .flexcenter {
      align-items: center;
    }
    .stretch {
      align-items: stretch;
    }
    .stretch li {
      height: auto;
      line-height: 100px;
    }
    .baseline {
      align-items: baseline;
    }
  </style>
</head>
<body>
  <div class="test">
    <h1>flex-start</h1>
    <ul class="flexstart">
      <li>1</li>
      <li>2</li>
      <li>3</li>
      <li>4</li>
      <li>5</li>
    </ul>
```

```
  </div>
  <div class="test">
    <h1>flex-end</h1>
    <ul class="flexend">
      <li>1</li>
      <li>2</li>
      <li style="align-self: flex-start">3</li>
      <li>4</li>
      <li>5</li>
    </ul>
  </div>
  <div class="test">
    <h1>center</h1>
    <ul class="flexcenter">
      <li>1</li>
      <li>2</li>
      <li>3</li>
      <li>4</li>
      <li>5</li>
    </ul>
  </div>
  <div class="test">
    <h1>Stretch</h1>
    <ul class="stretch">
      <li>1</li>
      <li>2</li>
      <li>3</li>
      <li>4</li>
      <li>5</li>
    </ul>
  </div>
  <div class="test">
    <h1>Baseline</h1>
    <ul class="baseline">
      <li style="line-height: 50px;">1</li>
      <li style="line-height: 40px;">2</li>
      <li style="line-height: 80px;">3</li>
      <li>4</li>
      <li style="line-height: 30px;">5</li>
    </ul>
  </div>
</body>
</html>
```

Este ejercicio usa diversos bloques con los distintos valores posibles de align-items, que habíamos conocido en el artículo de Propiedades para el contenedor Flexbox.

La imagen resultando del posicionamiento de estos elementos la podemos ver a continuación.



Pero lo cierto es que este ejercicio no estaría completo si solo vemos cómo se aplica align-items cuando los elementos se sitúan en filas. Habría que verlo también para elementos que se coloquen en columnas pues entonces, tal como se ha explicado, la aplicación de align-items es sensiblemente distinta.

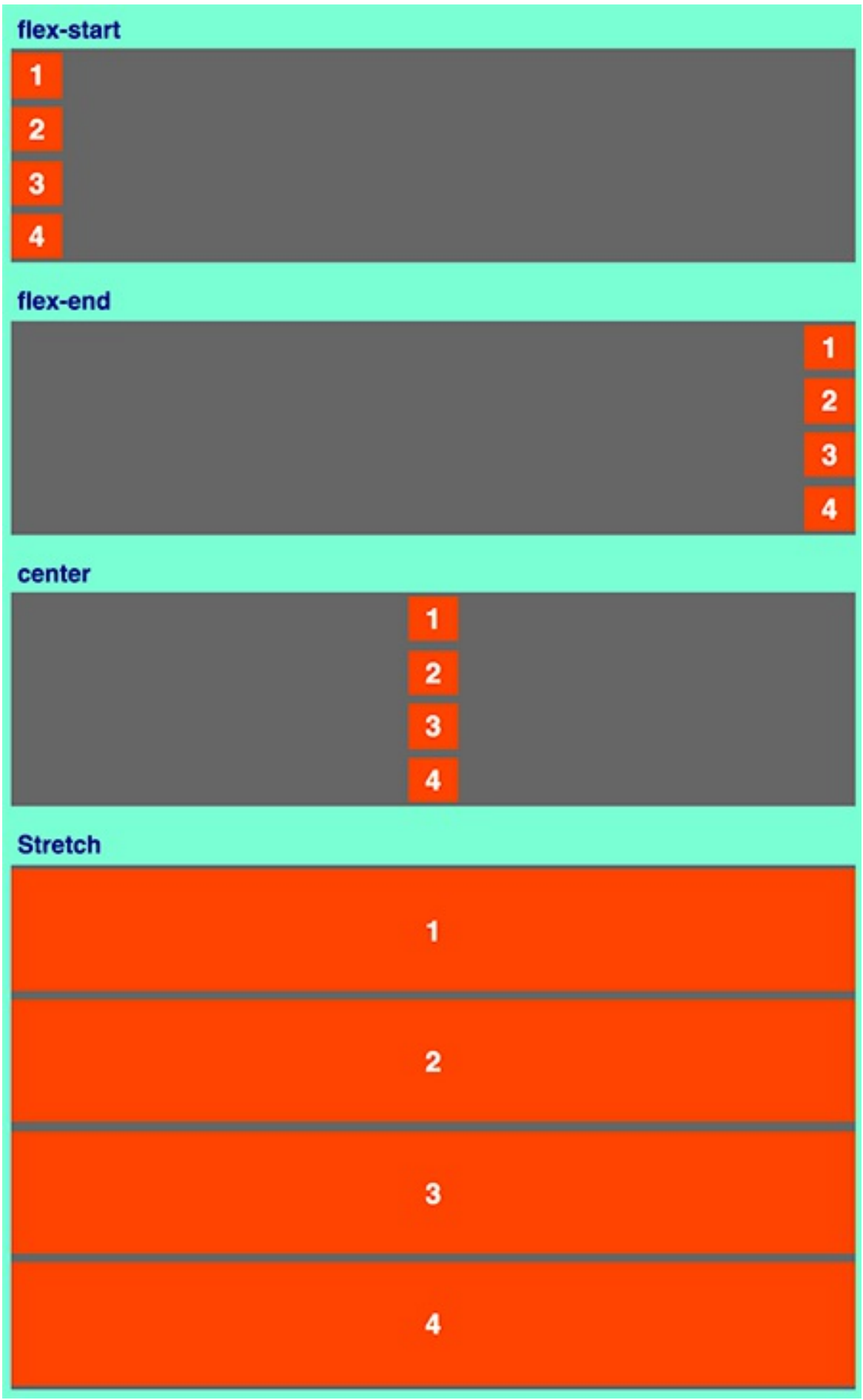
Podemos ver el código del siguiente ejemplo:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>align-items para columnas</title>
  <style>
    body {
      background-color: aquamarine;
      font-family: sans-serif;
    }
    ul, li {
      padding: 0;
      list-style: none;
    }
    ul {
      display: flex;
      flex-direction: column;
      margin: 10px;
      margin-top: 0;
      background-color: #666;
      justify-content: space-between;
```

```
    }
    li {
      display: block;
      background-color: orangered;
      line-height: 34px;
      font-size: 1.5em;
      font-weight: bold;
      text-align: center;
      color: #fff;
      margin: 4px 0;
      padding: 2px 15px;
    }
    h1 {
      margin: 20px 20px 4px 15px;
      font-size: 1.3em;
      color: darkblue;
    }
    .flexstart {
      align-items: flex-start;
    }
    .flexend {
      align-items: flex-end;
    }
    .flexcenter {
      align-items: center;
    }
    .stretch {
      align-items: stretch;
    }
    .stretch li {
      height: auto;
      line-height: 100px;
    }
    .baseline {
      align-items: baseline;
    }
  </style>
</head>
<body>
  <div class="test">
    <h1>flex-start</h1>
    <ul class="flexstart">
      <li>1</li>
      <li>2</li>
      <li>3</li>
      <li>4</li>
    </ul>
  </div>
  <div class="test">
    <h1>flex-end</h1>
    <ul class="flexend">
      <li>1</li>
      <li>2</li>
      <li>3</li>
      <li>4</li>
```

```
      </ul>
    </div>
    <div class="test">
      <h1>center</h1>
      <ul class="flexcenter">
        <li>1</li>
        <li>2</li>
        <li>3</li>
        <li>4</li>
      </ul>
    </div>
    <div class="test">
      <h1>Stretch</h1>
      <ul class="stretch">
        <li>1</li>
        <li>2</li>
        <li>3</li>
        <li>4</li>
      </ul>
    </div>
  </body>
</html>
```

Y a continuación el resultado que deberíamos apreciar para cada valor de align-items.


flex-start
1
2
3
4
flex-end
1
2
3
4
center
1
2
3
4
Stretch
1
2
3
4

Nota: he quitado en este caso el valor "baseline" de align-items, puesto que no tiene mucho sentido una alineación con respecto a la línea de texto, si cada elemento está abajo del anterior.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *08/09/2017*
Disponible online en *http://desarrolloweb.com/articulos/flexbox-align-items.html*