Introducción (y un poco de historia)

Flex, Flexbox, o "Flexible Box" fue introducido por primera vez en el borrador de especificación CSS en 2009. Su objetivo principal era abordar los desafíos que surgían al tratar de crear diseños complejos y flexibles en las interfaces web utilizando los modelos de diseño tradicionales de CSS. La necesidad de un sistema más eficiente y fácil de usar que el tradicional con float llevó al desarrollo de Flexbox.

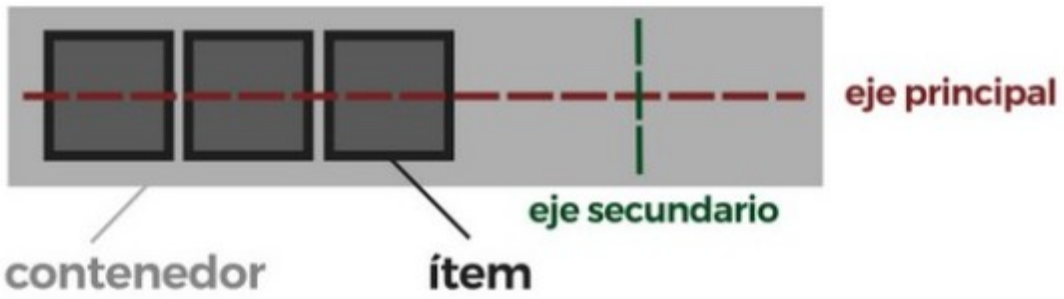
El trabajo inicial en Flexbox fue liderado por fantasai y Tab Atkins, dos miembros del Grupo de Trabajo de CSS (CSS Working Group) del W3C. La especificación pasó por varias iteraciones y ajustes antes de convertirse en una recomendación oficial del W3C en 2012. Desde entonces, Flexbox ha ganado amplia aceptación en la comunidad de desarrollo web debido a su capacidad para simplificar el diseño y la alineación de elementos en una dirección o en ambas, proporcionando mayor flexibilidad y control. Su adopción ha sido fundamental para mejorar la creación de diseños responsivos y dinámicos en sitios web modernos.

Hoy en día se usa, por ejemplo, en algunos de los diseños de Google, Facebook, X (antes Twitter), Microsoft, Netflix, GitHub, Amazon, Airbnb, Linkedin o Wordpress.

Para empezar

Flex nos permite realizar diseños de una única dimensión, ya sea ancho (fila) o alto (columna). Grid, sin embargo, nos va a permitir realizar diseños con dos dimensiones simultaneamente (filas y columnas)

Los elementos básicos en Flex son el contenedor y los items ¿te suena de los diseños que hemos hecho hasta ahora con float, verdad? La principal diferencia que vas a encontrar es que la parte del trabajo mas pesada y difícil que has hecho hasta ahora manualmente se realiza de forma automática.



El contenedor es el elemento padre que tendrá en su interior cada uno de los items. Los contenedores tendrán una orientación o eje principal que por defecto es horizontal (configuración de fila). La orientación o eje secundario será siempre perpendicular a la

principal.

Para activar el modo flex usamos en la caja contenedora la propiedad display especificando el valor flex:

HTML	CSS

  1
  2
  3
 	
	.contenedor {
	  display: flex;
	}

NOTA: con el valor flex el objeto se comporta como un elemento de bloque pero tenemos también la opción de usar inline-flex como valor. En ese caso el elemento se comporta como elemento de línea.

Contamos, una vez activado flex, con varias propiedades que nos permiten modificar la apariencia y comportamiento de nuestros items. Las iremos viendo a continuación de una en una y agrupadas según su utilidad:

Dirección de los ejes

flex-direction nos permite modificar la dirección del eje principal del contenedor para que sea horizontal o vertical con los valores row (horizontal, por defecto) o column (vertical). Añadiendo la partícula -reverse (row-reverse o column-reverse) coloca los items en orden inverso a como aparecen en el html

flex-wrap define el comportamiento del contenedor cuando los items no caben en el contenedor. Los posibles valores son los siguientes:

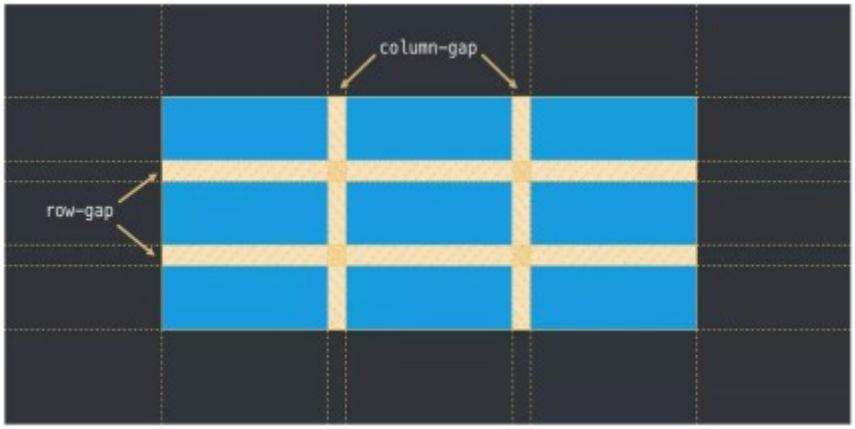
- nowrap – Los items estarán siempre en una sola línea impidiendo el desbordamiento
- wrap – Si los items no caben en una única linea desbordan el contenedor y ocupan una línea adicional
- wrap-reverse – Idem al anterior pero en este caso los elementos que no entran en el contenedor se desbordarían por encima y no por debajo

flex-flow es un atajo que nos permite definir de una sola vez las dos propiedades anteriores. Así por ejemplo:

flex-flow: row-reverse nowrap

Espaciado o separación entre items

La separación entre items (Gaps o Huecos según la terminología de Flex) se define mediante las propiedades row-gap y column-gap.



Puesto que Flex solo trabaja en una dimensión las dos propiedades no siempre tienen efecto. Definiremos row-gap cuando flex-direction esté configurado en modo column y con column-gap cuando flex-direction esté en modo row

NOTA: Salvo que tengamos definido el modo flex-wrap a wrap o wrap-reverse. En ese caso cuando, por ejemplo, una línea desborde tendremos separeción no sólo entre columnas sino también entre líneas

En ambos casos el valor de la propiedad puede ser normal (dejando que flex ajuste el espaciado entre items como crea conveniente) o el espaciado que queremos en cualquiera de las unidades de medida permitidas que conocemos:

```
row-gap: normal;
colum-gap: 10px;
```

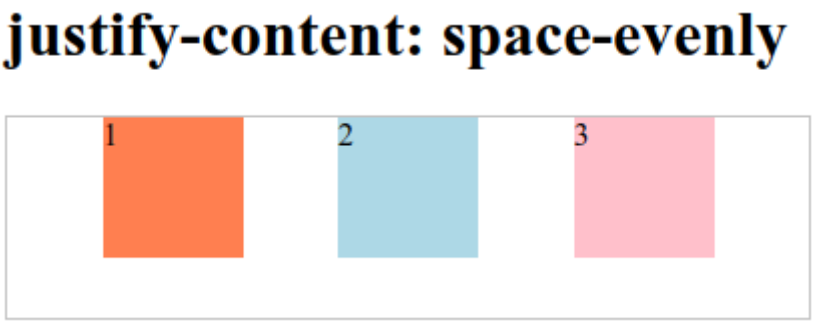
Tenemos también un atajo (gap) que nos permite especificar ambas propiedades de forma simultanea:

```
gap: 5px 10px; /* 5px para el espaciado entre filas y 10px entre columnas */
gap: 10px;     /* 10px en ambos casos */
```

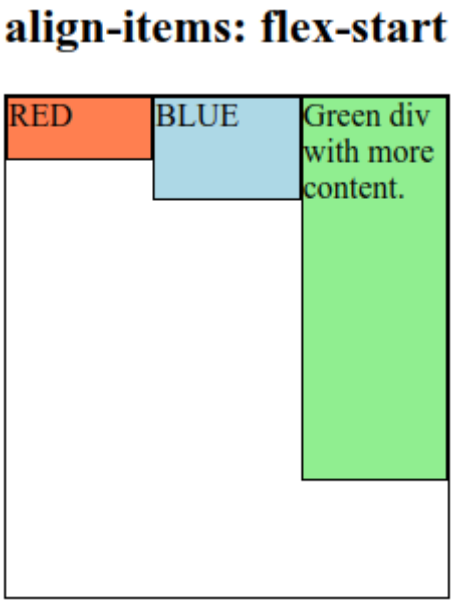
Alineación de elementos

Contamos con tres propiedades que nos permiten alinear los elementos de una u otra forma:

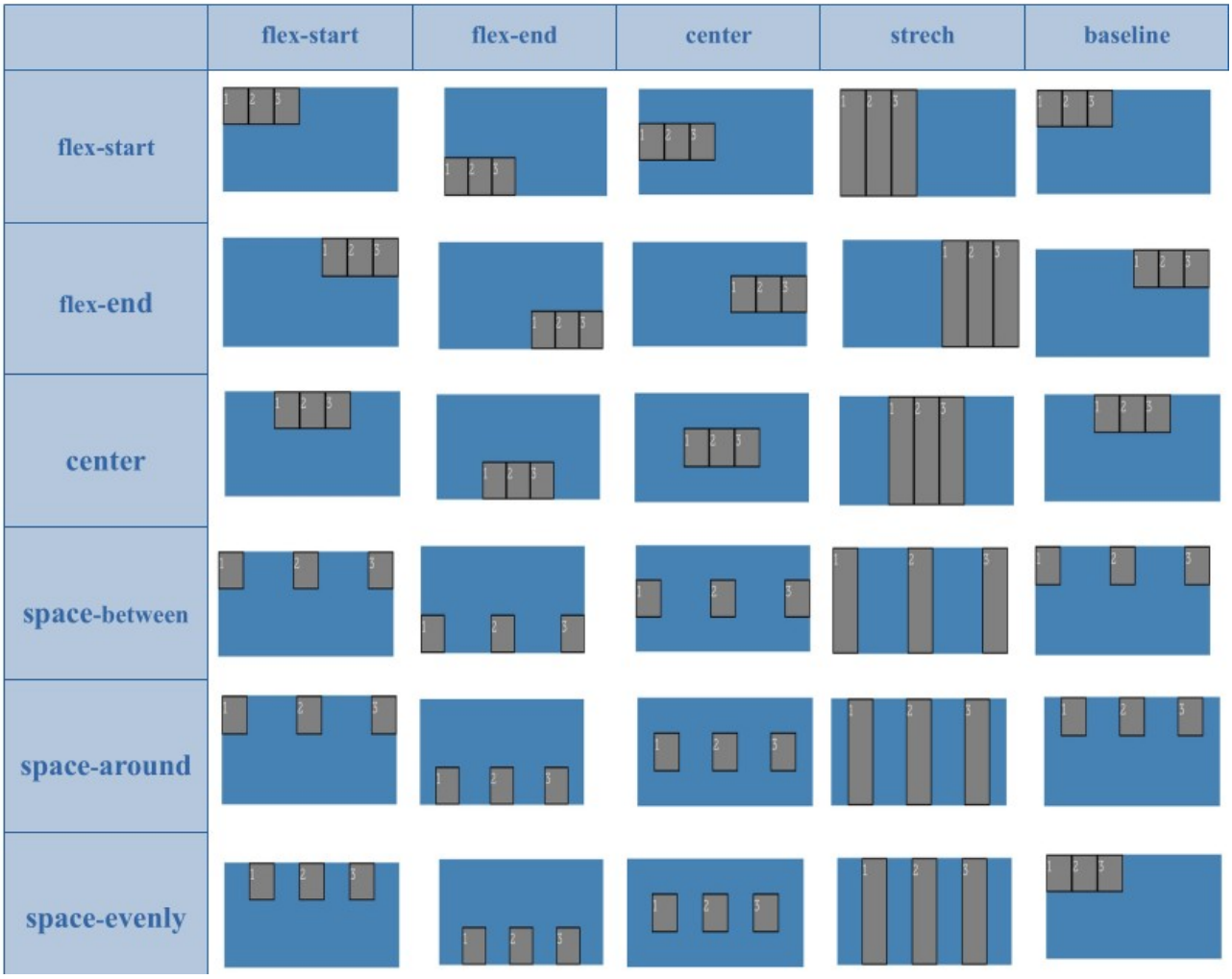
- **justify-content,** alinea los items usando el eje principal del contenedor. Los valores posibles son flex-start (por defecto al principio), flex-end (al final), center, space-between (repartiendo el espacio entre ellos), space-around (con espacio antes y después de los items lo cual hace que entre ellos el espacio sea doble) o space-evenly (repartiendo por igual el espacio antes, entre y después de los items, ver ejemplo)



- **align-items** alinea los items usando el eje secundario, normal (por defecto en el centro), stretch (llenan todo el espacio), center, flex-start, (arriba o al inicio del contenedor, ver ejempoo) flex-end (abajo o al final del contenedor). Otras opciones: start, end y baseline



A continuación tienes una tabla comparativa entre las diferentes opciones de justify-content y align-items que son las dos propiedades mas usadas

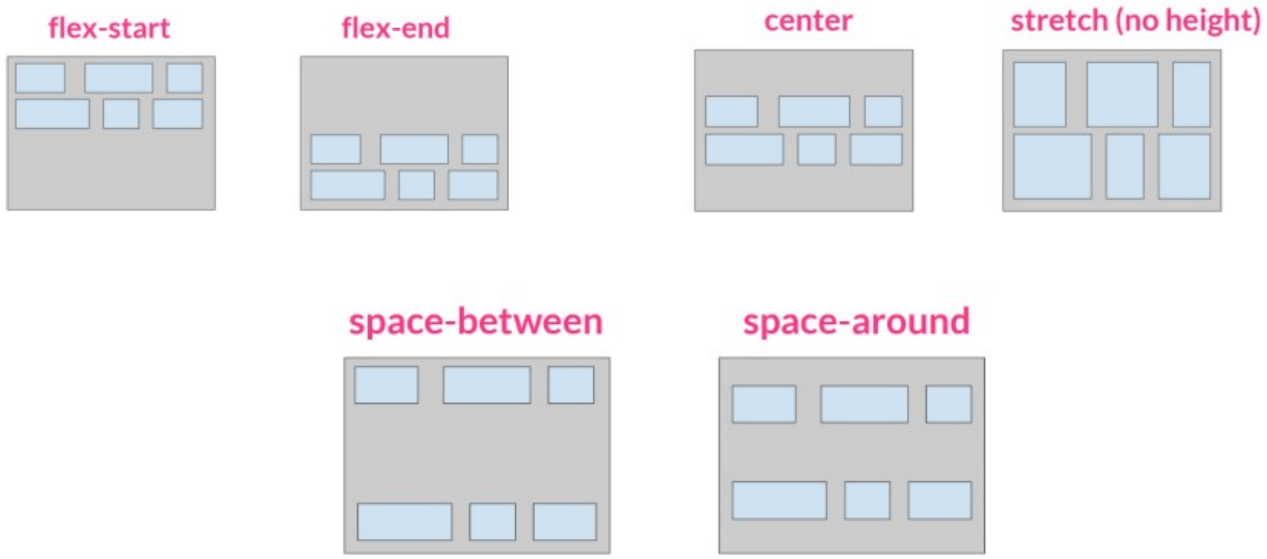


place-content es un atajo para las dos propiedades anteriores:

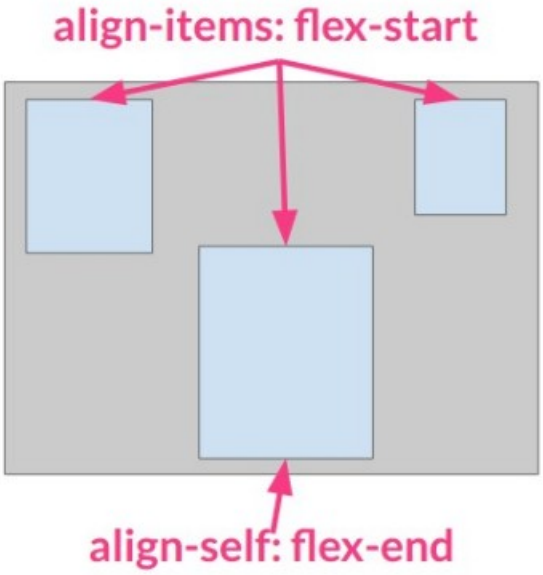
```
place-content: center flex-end;
```

El primer valor hace referencia a justify-content y el segundo a aling-items

- **align-content**, muy parecido (y con los mismos valores que align-items) aplicable cuando tenemos varias líneas (porque usamos wrap). align-items y align-content no solapan los valores que apliquemos. Si los elementos forman una sola línea se comportan como pongamos en aling-items y si desbordan y ocupan mas de una según pongamos en align-content:



- **align-self**, igual a los anteriores pero en este caso las opciones de alineación se aplican a los items de forma individual en lugar de al contenedor y al conjunto de los items.



Propiedades aplicables a los items

Salvo **align-self** (la última propiedad que hemos visto) todas las demás se aplican sobre el contenedor y tienen efecto sobre todos los elementos que contiene. Ahora nos centraremos en las propiedades que tienen efecto sobre los items de forma individual

Con la propiedad **order** podemos modificar el orden de los items independientemente de la forma en que aparezcan en el html. El valor de order debe de ser un número entero (positivo o negativo). Valores mas pequeños emplazaran a los items antes y valores mas altos los

emplazarán despues:

```
.item1{  order: 30; }
.item2{  order: -5; }
.item3{  order: 1; }
```

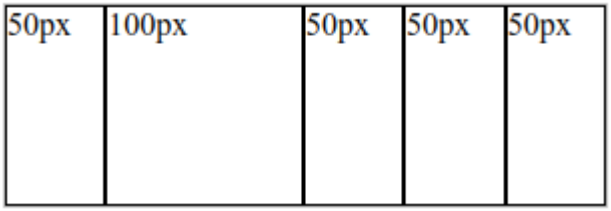
El anterior CSS colocaría en primer lugar el item2, en segundo el item3 y en tercero el item1, independientemente del orden en el que aparezcan en el html

flex-basis, flex-grow y **flex-shrink** son propiedades que nos van a permitir modificar el tamaño por defecto de los items y la forma en que estos crecen o decrecen en función del contenedor donde se ubican.

flex-basis nos permite seleccionar cual es el tamaño inicial de cada item. Su valor puede ser un porcentaje o cualquier unidad de medida válida. El valor auto dejará el tamaño en manos del navegador.
En el siguiente ejemplo todos los items tendrán inicialmente 50px salvo el segundo que tendrá 100px:

```
<div id="main">
   <div>50px</div>
   <div>100px</div>
   <div>50px</div>
   <div>50px</div>
   <div>50px</div>
</div>
```

```
#main {
   width: 300px;
   height: 100px;
   display: flex;
}
#main div {  flex-basis: 50px; }
div{  border: 1px solid black; }
#main div:nth-of-type(2) {  flex-basis: 100px; }
```



flex-grow y **flex-shrink** son propiedades opuestas y sus valores deberían de ser siempre enteros mayor que cero. Si el valor es n, la primera indica que el item crecerá siempre hasta ser dos veces mas grande que el resto mientras que en el segundo caso indica que decrecerá siendo 2 veces más pequeño que el resto.

Existe un atajo (flex) que sirve para configurar los tres valores con una única propiedad:

```
flex: 1 3 35% /* flex-grow: 1; flex-shrink: 3; flex-basis: 35%; */
```