# CAJAS FLEXIBLES EN CSS

## 1.- INTRODUCCIÓN A FLEXBOX

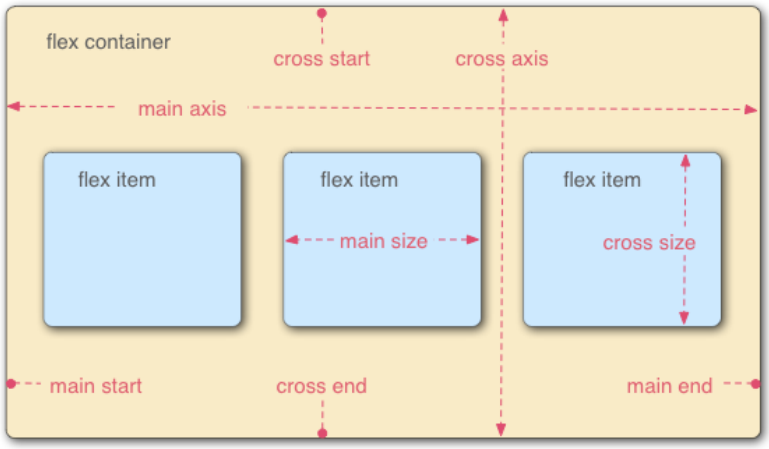
El módulo **FlexBox** pretende proporcionar un método eficiente para colocar, alinear y distribuir los espacios entre los elementos de un contenedor, incluso cuando su tamaño es desconocido o dinámico.

La idea principal detrás de Flex es proporcionar al contenedor la habilidad de modificar el ancho, alto y orden de sus elementos para rellenar de la mejor manera posible el espacio disponible. Un contenedor Flex expande los elementos que contiene para rellenar el espacio libre, o los comprime para evitar que sobresalgan.

Por otro lado Flex es **independiente de la dirección**, al contrario que los sistemas tradicionales de posicionamiento (donde los elementos en bloque se disponen verticalmente mientras que los elementos inline lo hacen de forma horizontal).

FlexBox no es una única propiedad sino que es un módulo completo que entre otras cosas incluye un conjunto completo de propiedades. Algunas de ellas se aplican al elemento contenedor (***contenedor flex***) mientras que otras se aplican a los elementos hijos de este contenedor (***elementos flex***).

Mientras que el posicionamiento que hemos visto hasta ahora está basado en el flujo de dirección dado por los elementos en bloque y en línea, Flex está basado en las *direcciones de flujo flex*.




Cuando hablamos de posicionamiento flexible hay una serie de términos que debemos conocer:

- **Contenedor flexible** (***flex container***): el elemento padre que contiene los elementos flexibles.
- **Elemento flexible** (***flex ítem***): cada hijo de un contenedor flex se convierte en un elemento flexible. Si hay texto directamente incluido en el contenedor flexible se envuelve automáticamente en un elemento flexible anónimo.
- **Ejes**: cada diseño de "caja flexible" sigue dos ejes:
  - **Eje principal**: es el eje a lo largo del cual los elementos se suceden unos a otros
  - **Eje secundario**: es el eje perpendicular al eje principal.

- **Direcciones:** los lados **inicio principal/fin principal** (***main start/main end***) e **inicio secundario/fin secundario** (***cross start/cross end***) del contenedor flexible describen el origen y final del flujo de los elementos flexibles.
- **Líneas:** los elementos flexibles pueden disponerse en una sola o varias líneas de acuerdo con la propiedad flex-wrap, que controla la dirección del eje secundario y la dirección en la que las nuevas líneas se apilan.
- **Dimensiones:** los elementos equivalentes a *altura* y *anchura* usados en los elementos flexibles son **tamaño principal** (***main size***) y **tamaño secundario** (***cross size***), que respectivamente siguen el eje principal y el secundario.

Algunas cuestiones que hay que tener en cuenta:

- Los hijos de un contenedor flexible que tengan posicionamiento absoluto se situarán de manera que su posición estática se determine en referencia a la esquina inicio principal de su contenedor flexible.
- Los márgenes de los elementos flexibles adyacentes no se colapsan.
- Para asegurar un tamaño mínimo de los elementos flexible se deben utilizar las propiedades `min-width: auto` y/o `min-height: auto`. Esto hace que su tamaño por lo menos sea el necesario para albergar su contenido.
- Si queremos mantener la compatibilidad con todos los navegadores debemos indicar todas las versiones prefijadas que se muestran en la captura de la derecha.

```
display: -webkit-box;
display: -moz-box;
display: -ms-flexbox;
display: -webkit-flex;
display: flex;
```

## 2.- PROPIEDADES DEL CONTENEDOR FLEX

### PROPIEDAD **DISPLAY**

La propiedad `display: flex;` indica que un elemento es un contenedor flex. Esto establece un contexto flex para todos sus hijos.

```
.page-wrap {
  display: -webkit-box;      /* OLD - iOS 6-, Safari 3.1-6 */
  display: -moz-box;         /* OLD - Firefox 19- (buggy but mostly works) */
  display: -ms-flexbox;      /* TWEENER - IE 10 */
  display: -webkit-flex;     /* NEW - Chrome */
  display: flex;             /* NEW, Spec - Opera 12.1, Firefox 20+ */
}
```

Una cosa que tienes que tener en cuenta es que cuando convertimos un elemento en flexible ya deja de comportarse como elemento en bloque o en línea. Esto quiere decir que los elementos en bloque ya no ocupan todo el ancho de su contenedor sino que **se limitarán a ocupar el ancho que necesiten** para su contenido a no ser que se indique lo contrario mediante alguna propiedad.

Por defecto los elementos se situarán a lo largo de la línea establecida por el eje principal pudiendo darse dos situaciones:

- Si los elementos no llegan a ocupar todo el espacio disponible simplemente quedará el espacio libre a la derecha o bien donde lo indique la propiedad `justify-content`.

- Si los elementos requieren más espacio del disponible se comprimirán de forma equitativa. Así, si por ejemplo tenemos un ancho de ventana de 500 píxeles y tenemos 5 elementos con la propiedad `width: 150px`; se comprimirán todos de forma que cada uno solo ocupará 100 píxeles de ancho, ignorando el valor de la propiedad `width`.
  Si en el mismo ejemplo redimensionamos la ventana del navegador hasta 900 píxeles de ancho veremos que cada elemento crecerá hasta sus 150 píxeles, pero no más. Por lo que quedarán 150 píxeles en blanco a la derecha.

## PROPIEDAD **FLEX-DIRECTION**

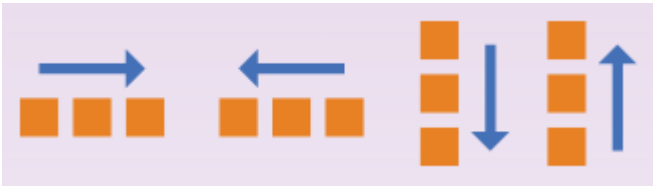
Indica **cuál es el eje principal** del contenedor flex, por lo que determina la dirección en la que se colocarán los elementos dentro de dicho contenedor.

```
.flex-v {
  -webkit-box-orient: vertical;
  -moz-box-orient: vertical;
  -ms-box-orient: vertical;
  -webkit-flex-direction: column;
  -moz-flex-direction: column;
  -ms-flex-direction: column;
  flex-direction: column;
}
```

Hay que tener en cuenta que la dirección de los elementos dentro del eje principal se verá afectada por la propiedad `direction`.

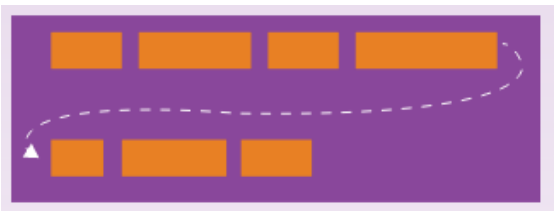
Los valores posibles son:

- `row`: izquierda a derecha en `ltr` y al revés en `rtl`.
- `row-reverse`: derecha a izquierda en `ltr` y al revés en `rtl`.
- `column`: igual que `row` pero de arriba abajo.
- `column-reverse`: igual que `row-reverse` pero abajo arriba.



## PROPIEDAD **FLEX-WRAP**

Por defecto, los elementos flex intentarán entrar en una única línea. Esto se puede cambiar y permitir a los elementos pasar a la siguiente línea. Con esta propiedad podemos evitar que los elementos se compriman cuando no caben en el ancho de la ventana, pasando a la siguiente línea una vez que se ha llegado al tamaño máximo.



La dirección también juega un papel importante aquí, indicando la dirección en la que se situarán las nuevas líneas.

Los valores posibles son:

- `nowrap`: Una única línea. Izquierda a derecha en `ltr` y de derecha a izquierda en `rtl`.
- `wrap`: múltiples líneas. Izquierda a derecha en `ltr` y de derecha a izquierda en `rtl`.
- `wrap-reverse`: múltiples líneas. Derecha a izquierda en `ltr` e izquierda a derecha en `rtl`.

## PROPIEDAD **FLEX-FLOW**

Es la propiedad *shorthand* que agrupa las propiedades `flex-direction` y `flex-wrap`. Ten en cuenta que, al igual que pasa con todas las propiedades *shorthand*, si los valores son palabras clave el orden en que se indican no es relevante.

```
.item {
    -webkit-flex-flow: row wrap;
    -moz-flex-flow: row wrap;
    -ms-flex-flow: row wrap;
    -ms-flex-direction: row;
    -ms-flex-wrap: wrap;
    flex-flow: row wrap;
}
```

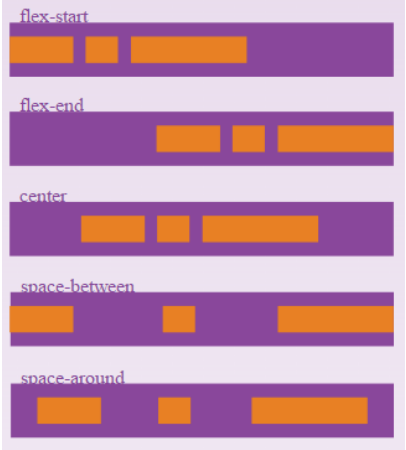
## PROPIEDAD **JUSTIFY-CONTENT**

Define el **alineamiento dentro del eje principal**. Ayuda a distribuir el espacio extra que queda libre cuando todos los elementos flex de una línea son inflexibles, o son flexibles pero han alcanzado su tamaño máximo.

```
.flex-center-v {
    -webkit-box-pack: center;
    -moz-box-pack: center;
    -ms-flex-pack: center;
    -webkit-justify-content: center;
    -moz-justify-content: center;
    justify-content: center;
}
```

Los posibles valores son:

- `flex-start:` los elementos se comienzan colocando desde el principio. Este es el valor por defecto.
- `flex-end:` los elementos se colocan al final
- `center:` los elementos están centrados a lo largo de la línea.
- `space-between:` los elementos se distribuyen de forma equidistante en la línea. No hay espacio antes del primer elemento ni espacio después del último.
- `space-around:` igual que el valor anterior pero se tiene en cuenta el espacio antes y después de cada elemento. Esto quiere decir que habrá espacio antes del primer elemento y después del último. Los espacios intermedios serán el doble de grandes ya que suponen la unión del espacio del elemento anterior más el del posterior.



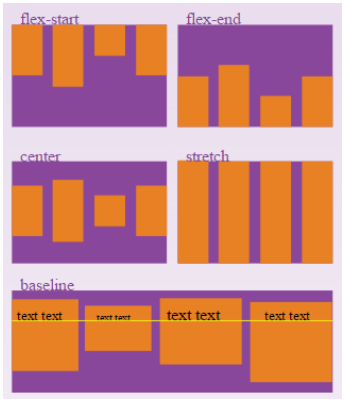

## PROPIEDAD **ALIGN-ITEMS**

Define el comportamiento de los elementos flex en cuanto a su disposición a lo largo del eje secundario respecto a la línea actual. Ten en cuenta que el alto de los elementos está definido por su contenido a no ser que se especifique lo contrario mediante la propiedad `height`.

```
.flex-center-v {
  text-align: center;
    -webkit-box-align: center;
  -webkit-align-items: center;
    -moz-align-items: center;
  align-items: center;
}
```

Los valores disponibles son:

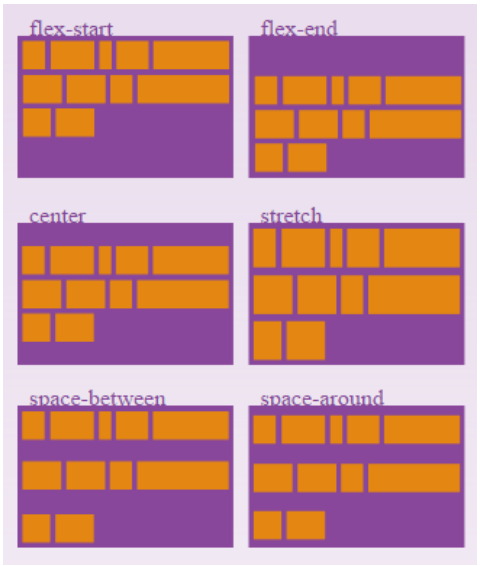
- `flex-start:` el inicio de los elementos ubicados en el eje secundario se sitúa en la línea de inicio secundario.
- `flex-end:` el borde final de los elementos ubicados en el eje secundario se sitúa sobre la línea de fin secundario.
- `center:` los elementos se centran sobre el eje secundario.
- `baseline:` los elementos se alinean en función de la línea base de su contenido.
- `stretch:` los elementos se expanden para ocupar todo el espacio disponible en el eje secundario (valor por defecto).




## PROPIEDAD **ALIGN-CONTENT**

Esta propiedad establece cómo se gestionan los **espacios en blanco entre elementos a lo largo del eje secundario** (de forma equivalente a como lo hace `justify-content` en el eje principal).

El alto del contenedor, por defecto, es el necesario para contener los elementos flex. En este caso esta propiedad no se aplica ya que no va a quedar espacio libre en el eje secundario.




Los posibles valores son:

- `flex-start`: los elementos se agrupan al principio del contenedor.
- `flex-end`: los elementos se agrupan al final del contenedor.
- `center`: los elementos se agrupan centrados en el contenedor.
- `space-between`: las líneas se distribuyen de forma equidistante, situándose la primera sobre la línea de inicio del eje y la segunda sobre la línea final del mismo.
- `space-around`: cada elemento tiene un espacio antes y después, distribuyéndose el tamaño de los espacios de forma equitativa.
- `stretch`: las líneas se redimensionan para ocupar todo el espacio disponible.

# 3.- PROPIEDADES PARA LOS ELEMENTOS FLEX

## PROPIEDAD **ORDER**

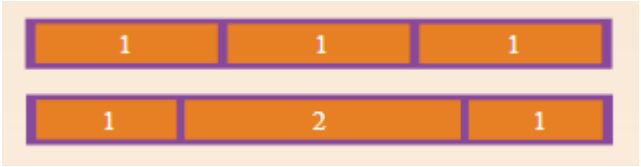
Por defecto los elementos se colocan en el mismo orden en que se encuentran en el código fuente. Sin embargo este orden se puede modificar mediante esta propiedad. El valor es numérico, colocándose primero los elementos que menor valor tengan.

```
.main-nav {
  -webkit-box-ordinal-group: 1;
  -moz-box-ordinal-group: 1;
  -ms-flex-order: 1;
  -webkit-order: 1;
  order: 1;
}
```

Esta propiedad es muy útil cuando queremos aplicar diferentes estilos a una misma página en función del ancho de la ventana del navegador.

## PROPIEDAD **FLEX-GROW**

Define la habilidad de los elementos flex para crecer si es necesario. Acepta un valor numérico que sirve como proporción, cuanto mayor sea el valor de un elemento con respecto al valor del resto de los elementos mayor será el espacio que ocupará.




## PROPIEDAD **FLEX-SHRINK**

Indica la capacidad de un elemento flex para encogerse. No se permiten valores negativos. El valor por defecto es 1 y cuanto mayor sea el valor indicado más capacidad tendrá el elemento para encogerse.

## PROPIEDAD **FLEX-BASIS**

Define el tamaño por defecto de un elemento antes de distribuir el espacio que queda. Puede ser una longitud (20%, 5rem,…) o una palabra clave. La palabra clave `auto` indica que se utilizará el tamaño establecido por las propiedades `width` y `height`.

Otras palabras clave son `content (basado en el contenido del elemento)`, `max-content`, `min-content` y `fit-content`, pero aún no está soportadas por los navegadores.

## PROPIEDAD **FLEX**

Es la propiedad abreviada para `flex-grow`, `flex-shrink` y `flex-basis`. Es altamente recomendado utilizar esta propiedad en lugar de las tres anteriores ya que optimiza su aplicación por parte de los navegadores.

```
.main-sidebar {
  -webkit-box-flex: 1;      /* OLD - iOS 6-, Safari 3.1-6 */
  -moz-box-flex: 1;         /* OLD - Firefox 19- */
  width: 20%;               /* For old syntax, otherwise collapses. */
  -webkit-flex: 1;          /* Chrome */
  -ms-flex: 1;              /* IE 10 */
  flex: 1;                  /* NEW, Spec - Opera 12.1, Firefox 20+ */
}
```

Aprovechemos esta propiedad para explicar estas tres propiedades ya que su comportamiento en ocasiones puede resultar desconcertante.

Lo primero que tenemos que tener en cuenta es que cada elemento flex tiene un **tamaño inicial**. Este tamaño inicial se puede establecer de tres maneras:

- Mediante la propiedad `width`.
- Mediante la propiedad `flex-basis`.
- Si no se ha definido ninguna de las dos propiedades anteriores tomará el tamaño que requiera su contenido.

Si dejamos los valores por defecto el elemento se comportará de la siguiente manera:

- **Si tiene espacio de sobra** en el contenedor flex, bien porque el ancho de todos los elementos de la línea sea inferior al ancho del contenedor, bien porque está establecida la propiedad `flex-wrap: wrap;` **el elemento conservará su tamaño inicial**.
- **Si no hay espacio suficiente** en el contenedor flex **el elemento se encogerá** lo necesario para caber en el contenedor.

Este valor por defecto corresponde con las propiedades `flex-grow: 0` y `flex-shrink: 1`, que quiere decir que el elemento no se expande pero sí se encoge lo que sea necesario.

Indicar un valor superior a la propiedad `flex-grow` significa que los elementos sí que se expandirán todo lo necesario hasta alcanzar el ancho de la ventana del navegador. Si todos los elementos tienen el mismo valor lo harán de forma equitativa, pero si tienen diferentes valores el tamaño que cogen será proporcional a su valor. Así por ejemplo en la siguiente imagen se pueden ver tres elementos que tienen valores 1, 2 y 1 respectivamente. De esta forma el segundo elemento ocupa el doble de ancho que los otros dos.


Pero observa lo que ocurre en la siguiente imagen donde hemos cambiado el contenido de los elementos, aunque no las propiedades (siguen teniendo el valor de `flex-grow` establecido en 1, 2 y 1 respectivamente):


Ahora parece que el navegador está ignorando la propiedad `flex-grow` ya que el segundo elemento ya no es el doble de grande que los otros dos. Sin embargo sí que está aplicando la propiedad pero de una forma que no es la que se nos puede dar a entender cuando leemos la descripción de dicha propiedad.

Para comprenderlo tenemos que acordarnos del **valor inicial** del que hablamos unos párrafos antes. Este valor inicial habíamos dicho que se puede establecer mediante la propiedad `width`, mediante la propiedad `flex-basis` o, en caso de que no se hayan definido ninguna de las dos propiedades, en función del contenido.

En nuestro ejemplo es el contenido el que determina el valor inicial del tamaño del elemento.

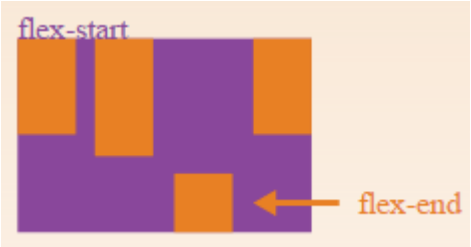
Los cálculos que realiza flex para calcular cuánto debe crecer cada elemento son los siguientes:

- **Calcula el valor inicial** de cada elemento. En el ejemplo anterior depende del contenido y supongamos que son 30, 30 y 450 píxeles.
- **Calcula el espacio libre que queda hasta completar el contenedor**. Siguiendo con las suposiciones imaginemos que la ventana del navegador tiene 900 píxeles de ancho por lo que el espacio libre será 900-30-30-450 = 390 píxeles. Esto es lo que tienen que crecer en total todos los elementos.
- Luego **suma el valor de la propiedad `flex-grow` de todos los elementos**. Por tanto en nuestro ejemplo será 1+2+1=4
- Y **divide el espacio libre entre el resultado de la suma anterior**, siendo por tanto 390/4=97,5
- Este tamaño equivaldrá a cada unidad de `flex-grow`, por tanto al primer elemento le añadirá aproximadamente 97 píxeles, al segundo 195 píxeles (97.5*2) y 97 al tercero. Este tamaño se suma al valor inicial.

La propiedad `flex-shrink` sigue un proceso similar para calcular cuánto debe encoger un elemento en caso de que no haya espacio suficiente. En este caso calcula el espacio total que debe quitar a todos los elementos y quita a cada uno la proporción que corresponda en función de su valor establecido en dicha propiedad.

## PROPIEDAD **ALIGN-SELF**

Permite que el alineamiento por defecto (o el especificado por la propiedad `align-items`) sea modificado para un elemento individual. Sus valores posibles son los mismos que en la propiedad `align-items`.



## 4.- DETECTAR EL TAMAÑO DE LA VENTANA DEL NAVEGADOR

El modelo FlexBox es muy útil para diseñar páginas que se adapten al contenido de la ventana del navegador pero con las propiedades que hemos visto hasta ahora no se llega a alcanzar todo su potencial. Por mucho que se puedan estirar o encoger los elementos HTML está claro que no se puede mantener la misma disposición en la página en un navegador con una ventana de 1920 píxeles de ancho que un teléfono móvil con un ancho de únicamente 640 píxeles.

Para ello es habitual combinar el uso de FlexBox con la regla **@media**. Como recordarás las reglas at son unas reglas especiales en CSS que agrupan una serie de propiedades.

A la regla @media se le indica una serie de condiciones que ha de cumplir el navegador y en caso de que se verifiquen aplica las propiedades indicadas entre las llaves. En la siguiente línea de código se puede ver un ejemplo de su uso:

```
header {
  flex-direction: column;
}
@media (max-width: 700px) {
  header {
    flex-direction: column;
  }
}
```

En este ejemplo los elementos dentro del *header* estarán dispuestos en una línea en el eje horizontal. Sin embargo si se cumple la condición de que la ventana del navegador no excede un acho de 700 píxeles se aplicará la propiedad `flex-direction: column`; sobre el *header*, sobre-escribiendo la regla anterior y haciendo que se dispongan verticalmente.

Aunque por ahora solamente utilizaremos las condiciones **max-width** y **min-width** hay un gran número de condiciones que se pueden pasar a la regla @media como puede ser la orientación del dispositivo, su resolución o los colores soportados.

## 5.- REFERENCIAS


- **CSS-Tricks. A complete guide to Flebox.** https://css-tricks.com/snippets/css/a-guide-to-flexbox/
- **Usando cajas flexibles en CSS.** https://developer.mozilla.org/es/docs/Web/CSS/CSS_Flexible_Box_Layout/Usando_las_cajas_flexibles_CSS
- **@media en MDN (Mozilla Developers Network).** https://developer.mozilla.org/es/docs/Web/CSS/@media
- **Ejemplo de uso de Flexbox.** https://codepen.io/martinwolf/pen/kcnsF


**Práctica**

http://codepen.io/indyplanets/pen/qBuLi