**GUIA DE FLEXBOX – FUNDAMENTOS**

Comencemos diciendo que:

* **Block** es un modelo de layout, son elementos de bloque que están uno debajo del otro.
* **Inline** es un modelo de layout, son elementos en línea que aparecen uno alado del otro.
* **Display Table** es un modelo de layout, no es el “Table” del HTML.
* **Position** es un modelo de layout.
* **Display Flex** es un nuevo modelo de layout.
* **Display Grid** es un nuevo modelo de layout. (Guía Pendiente…)

**Flexbox**, junto con **Grid**, son los nuevos **modelos de layout** **(esto es la organización que tendrán los elementos para aparecer en la pantalla)**, pero hablaremos sólo de Flexbox por ahora.

**¿CÓMO FUNCIONA FLEXBOX?**

Para poder trabajar con este nuevo modelo es necesario remarcar que **Flexbox necesita la relación** **PADRE-HIJO** o con otro termino, **CONTAINER/ITEMS**, sin esta relación Flexbox no funciona.

Para hacer que un contenedor se vuelva Flexbox, primero que nada, se declara la propiedad "**display**" y con su valor "**flex**".

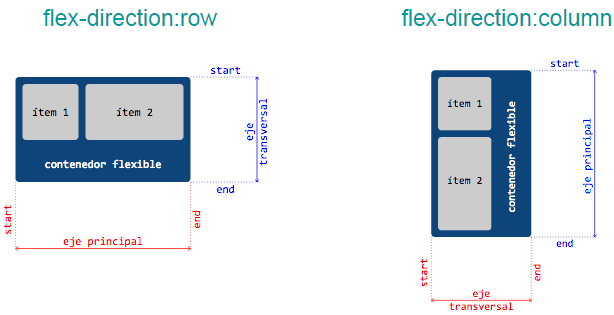
**PROPIEDADES PARA EL CONTENEDOR FLEXBOX**

Existen propiedades que únicamente se le pueden aplicar a contenedores, y las conoceremos a continuación. Algo muy importante que debe comprenderse es que con este modelo se manejan 2 ejes:

* **Main Axis o Eje Principal**
* **Cross Axis** **o Eje Secundario**

Un contenedor Flexbox posee estos 2 ejes o direcciones sobre los cuales se trabaja, la propiedad para manipular estos ejes es "**flex-direction**" y posee los valores "**row**" y "**column**":

* **Main Axis o Eje Principal:** Por default es horizontal, y es de izquierda/START a derecha/END. La propiedad es "**flex-direction:row;",** pero puede cambiarse a vertical usando la propiedad "**flex-direction:column**;".
* **Cross Axis o Eje Secundario:** La dirección de este eje es lo contario al Main Axis. Si Main Axis es horizontal entonces Cross Axis es vertical y va de arriba/START hacia abajo/END, y si Main Axis es vertical entonces Cross Axis es horizontal y va de izquierda a derecha.
* **display: flex;** Como se mencionó en la 2ª sección de este documento, esta propiedad es indispensable activar para comenzar a trabajar con Flexbox. Todo contenedor que posea esta propiedad declarada se convierte en un contenedor padre. El valor “**flex**” es para que los hijos/ítems que habrá dentro del contenedor padre se vuelvan elementos flexibles. Al activar esta propiedad se activan otras propiedades que pueden utilizarse.
* Nuevamente hay que remarcar que al usar “**display:flex**” automáticamente se generan 2 ejes: el **Main (principal) axis** y el **Cross (secundario) axis**. Por defecto, cuando se utiliza el valor “**flex**”, el Main axis es horizontal y el Cross axis es vertical.



Ahora, pasemos con las propiedades únicas para los contenedores **padre/container**:

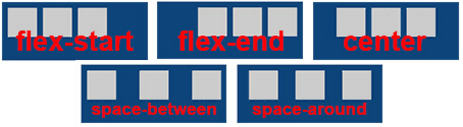
* **Flex-direction (Definición del Axis)**
* **Justify-content (De acuerdo al eje Main)**
* **Align-items (De acuerdo al eje Cross)**
* **Flex-wrap (De acuerdo al eje Main)**
* **Flex-flow (Shorhand)**
* **Align-content (De acuerdo al eje Cross)**
* **flex-direction (valor por default: row)**: Define la dirección de los ejes principal y secundario que tendrá el contenedor Flexbox y tiene los siguientes valores:

1. **row**: Coloca el eje principal de forma horizontal, de izquierda a derecha.
2. **row-reverse**: Coloca el eje principal de forma horizontal derecha a izquierda.
3. **column**: Coloca el eje principal de forma vertical, de arriba hacia abajo.
4. **column-reverse**: Coloca el eje principal de forma vertical, de abajo hacia arriba.

Para distribuir, organizar o posicionar los elementos que estarán dentro de un contenedor Flexbox:

* **justify-content (valor por default: flex-start)**: Posiciona los elementos que hay dentro de un contenedor Flexbox **de acuerdo al eje principal** **que esté usándose** y tiene los siguientes valores:

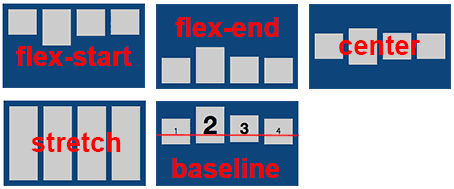
1. **flex-start**: Los elementos del grupo se posicionan al inicio del eje principal que esté usando el contenedor.
2. **flex-end**: Los elementos del grupo se posicionan al final del eje principal que esté usando el contenedor.
3. **center**: Los elementos del grupo se posicionan al centro del eje principal que esté usando el contenedor. Este valor puede reemplazar fácilmente lo que se hacía usando “**margin auto**” para dejar algún elemento centrado.
4. **space-between**: Distribuye uniformemente los elementos a lo largo del eje principal que esté usando el contenedor, de manera que el primer elemento se alinee al principio/start y el último elemento se alinee al final/end.
5. **space-around**: Distribuye uniformemente los elementos a lo largo del eje principal de modo que todos los elementos tengan el mismo espacio alrededor de ellos.



* **align-items (valor por default: stretch)**: Posiciona o alinea los elementos que hay dentro de un contenedor Flexbox **de acuerdo al eje secundario** **que esté usándose** y tiene los siguientes valores:

1. **flex-start**: Alinea los elementos a lo largo del inicio del eje secundario del contenedor.
2. **flex-end**: Alinea los elementos a través del final del eje secundario del contenedor.
3. **center**: Alinea los elementos a través del centro del eje secundario del contenedor.
4. **baseline**: Hace que los elementos se posicionen de acuerdo a una base línea, se puede tomar como ejemplo la base de algún texto.
5. **stretch**: Estiramiento de tal manera que abarquen todo el eje secundario, o sea, ajusta los elementos acorde al alto o ancho que tenga el contenedor padre. Es necesario no utilizar valores como “**height**” o “**width**” sobre algún elemento en el que utilizaremos “**stretch**”.

Cuando el Flexbox es multilínea, o sea, cuando está usando “**flex-wrap:wrap|wrap-reverse;**”, la propiedad “**align-items**” no funciona de nada, debe de utilizarse “**align-content**”.



* **flex-wrap (valor por default: nowrap)**: Los elementos dentro un contenedor se acomodan o distribuyen de acuerdo al ancho del contenedor padre, o sea, si no hay espacio saltan a una siguiente línea. Extiende o distribuye los elementos Flexbox que se encuentren dentro del contenedor Flexbox **de acuerdo al eje principal** **que esté usándose**, utilizar el valor “**wrap|wrap-reverse**” activa la propiedad “**align-content**”, podría confundirse con que es igual a "**justify-content**", pero no son lo mismo, y tiene los siguientes valores:

1. **nowrap**: Los elementos flex aparecen en una sola línea. En ciertas circunstancias los elementos flex aparecen redimensionados para puedan acomodarse dentro del contenedor flex, o sea, se adecuan, ajustan o encajan en una sola línea.
2. **wrap**: Los elementos flex aparecen colocados en varias líneas, o sea, se adecuan, ajustan o encajan en múltiples líneas.
3. **wrap-reverse**: Los elementos flex aparecen colocados en varias líneas, o sea, se adecuan, ajustan o encajan en múltiples líneas en sentido inverso.

Cuando el Flexbox es multilínea, o sea, cuando está usando “**flex-wrap:wrap|wrap-reverse;**”, la propiedad “**align-items**” no funciona de nada, debe de utilizarse “**align-content**”.



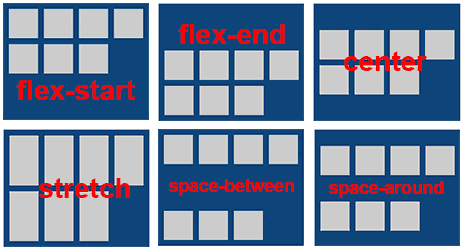
* **flex-flow (valor por default: row nowrap)**: Es un shorthand que combina "**flex-direction**" y "**flex-wrap**", y así podemos usarlos conjuntamente, por ejemplo: “**flex-flow: row wrap;**”.

Sintaxis: “**flex-flow: #flex-direction #flex-wrap (opcional);**”

* **align-content (valor por default: stretch)**: Establece cómo las líneas múltiples de elementos están separadas o cuánto espacio hay entre sí, o sea, determina el espacio entre líneas de elementos **de acuerdo al eje secundario que esté usándose**. Esta propiedad sólo se utiliza o activa cuando conjuntamente usamos "**flex-wrap:wrap|wrap-reverse**", y podría confundirse con que es igual a "**justify-content**" porque posee los mismos valores, pero no son lo mismo, tiene los siguientes valores:

1. **flex-start**: Las líneas de elementos se agrupan hacia la parte del inicio del eje principal que esté usando el contenedor.
2. **flex-end**: Las líneas de elementos se agrupan hacia la parte del final del eje principal que esté usando el contenedor.
3. **center**: Las líneas de elementos se agrupan hacia la parte central del eje principal que esté usando el contenedor. Este valor puede reemplazar fácilmente lo que se hacía usando “**margin auto**” para dejar algún elemento centrado.
4. **space-between**: Las líneas de elementos se agrupan uniformemente a lo largo del eje principal que esté usando el contenedor, de manera que la primera línea de elementos se agrupa al principio y el último elemento se agrupa al final.
5. **space-around**: Las líneas de elementos se agrupan uniformemente a lo largo del eje principal que esté usando el contenedor, de modo que todas las líneas de elementos tengan el mismo espacio alrededor de ellas.
6. **stretch**: Las líneas de elementos se agrupan de forma estirada para adaptarse al contenedor. Parece ser que este valor tiene por default esta propiedad.

Cuando el Flexbox es multilínea, o sea, cuando está usando “**flex-wrap:wrap|wrap-reverse;**”, la propiedad “**align-items**” no funciona de nada, debe de utilizarse “**align-content**”.



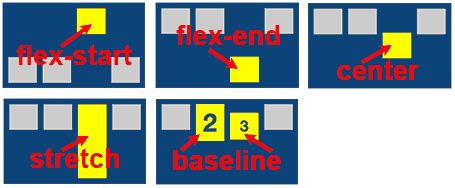
**PROPIEDADES PARA LOS ELEMENTOS FLEXBOX**

Así como existen propiedades que únicamente se le pueden aplicar a los contenedores, también hay propiedades que únicamente se le pueden aplicar a los elementos están dentro del contenedor Flexbox, y las conoceremos a continuación. Existen también una que otra propiedad que se le puede aplicar únicamente a elementos individuales o específicos:

* **order (valor por default: 0)**: Establece el orden en el que estarán los elementos, se puede cambiar el orden de la secuencia de los elementos que vienen maquetándose, o visto de otra forma, define el orden en el que un elemento aparece en el contenedor Flexbox y acepta valores enteros positivos y negativos. Todos los elementos Flexbox comienzan con un valor predeterminado de orden 0, por lo que un elemento con un valor de orden mayor se reposicionará en relación con los elementos aún configurados en sus órdenes predeterminados.

Sintaxis: “**order: #número;**”

* **align-self (valor por default: stretch)**: Tiene la misma descripción que la propiedad "**align-items**" y los mismos valores, también funciona **de acuerdo al eje secundario que se esté usando**,pero con la diferencia que esta propiedad solamente se aplica a un elemento individual.

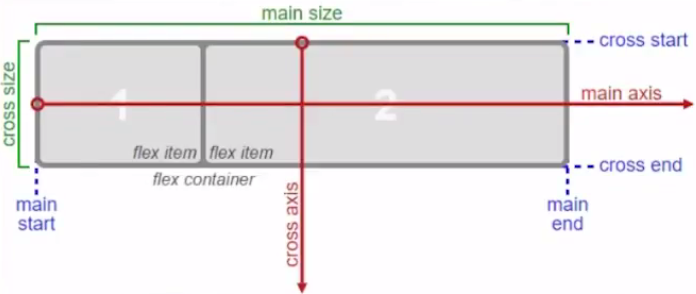


En cuanto a propiedades que son para los elementos/ítems hay una cosa más que se debe tomar en cuenta, y es que al igual que con los contenedores padre se juega y elige con los ejes principales y secundarios, algo muy similar sucede con los **Tamaños Principales y Secundarios** que tiene un elemento o ítem.

Existen propiedades específicas que se utilizan para los tamaños y las conoceremos a continuación:

**Main Size** y **Cross Size** son las dimensiones que tendrán los flex ítems, y estos tamaños están vinculados al eje que esté utilizándose, por ejemplo: Si se está usando como eje principal “**column**”, entonces sobre el flex ítem se podrá manipular el ancho (**width**) y viceversa.

* **Main Size** **o** **Tamaño Principal** -> **Main Axis** (Default: **width**)
* **Cross Size** **o** **Tamaño Secundario** -> **Cross Axis** (Default: **height**)



Utilizar **Cross Size** tiene 3 escenarios:

1. Se le ha definido un tamaño al contenedor o a los elementos por medio de “**width**” o “**height**”, por ejemplo:

* *.f-item{ height: 200px; }*
* *.f-item{ width: 200px; }*

1. No se ha definido un tamaño y no se ha definido “**align-items**” o “**align-content**”. El Cross Size ocupa todo el espacio disponible del contenedor, o sea, está en “**stretch**”.

* *.f-container{*

*background: lightblue;*

*margin: auto;*

*display: flex;*

*}*

1. No se define un tamaño, pero si se define “**align-items**” o “**align-content**” diferente de “**stretch**”.

* *.f-container{*

*width: 500px;*

*height: 500px;*

*background: lightblue;*

*margin: auto;*

*display: flex;*

*align-items: flex-start; //****el tamaño que se adquiere con esta propiedad está relacionado al contenido que hay dentro del flex ítem***

*}*

Al utilizar **Main Size** se debe tomar en cuenta lo siguiente:

1. El espacio disponible que no están ocupando los ítems, espacio ocupado que están ocupando los ítems (relativo al eje en el que estén).
2. Al espacio disponible siempre hay que quitarle los márgenes. Los márgenes también se toman en cuenta como espacio ocupado. Los márgenes nunca se colapsan, o sea, nunca se reducen en Flexbox, solo las cajas son las que se reducen, los márgenes no.

* *.f-item{*

*background: teal;*

*margin: 10px;*

*}*

1. Los ítems se rigen por los mismos principios del modelo de caja para calcular sus tamaños. Los “**paddings**” y los bordes se suman al tamaño total de la caja, excepto cuando se utiliza “**box-sizing: border-box**”, con esta propiedad los “**padding’s**” y los bordes se calculan por dentro de la caja.
2. El espacio disponible podría ser negativo/positivo

Existen propiedades encargadas de todo lo que arriba se ha explicado. Estas son 3 propiedades importantes que se utilizan para calcular el **Tamaño Principal** o **Main Size**:

* **flex-basis**: Es la propiedad que me da el tamaño base del tamaño. nuevamente, dependiendo del eje que se utilice será el Size que se manipulará. Por default, el Main Size es el “**width**”, o sea, es lo mismo que usar “**width**” (hay que tomar en cuenta el eje principal que esté usándose).

Sintaxis: “**flex-basis: #número;**”

* + - *.f-container{ flex-direction: column; //Eje principal a utilizar }*
    - *.f-item{*

*flex-basis: 150px; // Este “****basis****” se convierte en “****height****”, si utilizara como eje principal “****row****” entonces mi “****basis****” se convertiría en “****width****”*

*}*

**NOTA: Ahora, independientemente del “basis” que estemos controlando, pueden utilizarse las propiedades “width” o “height”. Si nuestro “basis” se convierte en “height”, podemos utilizar “width”. si nuestro “basis” se convierte en “width”, podemos utilizar “height”.**

* *.f-container{ flex-direction: column; // Eje principal vertical }*
* *.f-item{*

*flex-basis: 150px; // Si el eje principal fuera “****row****” entonces nuestro “****basis****” seria “****width****” y podríamos utilizar “****height****”*

*width: 200px;*

*}*

* *.f-item{*

*flex-basis: 150px; // Si el valor fuera “****auto****” el flex ítem tomaría el tamaño dependiendo de su contenido*

*width: 200px;*

*}*

**Recomendación de Álvaro Felipe**: **NO USAR “flex-basis” Y UTILIZAR “width”.**

* **flex-grow**: Establece el crecimiento que tendrá cada elemento. Según el valor que le pongamos será el tamaño que tomará nuestro elemento sobre los demás elementos dentro del contenedor padre. Los valores hacen que los elementos tengan tamaños proporcionales, o sea, que se ajusten.

Sintaxis: “**flex-grow: #número;**”

* **flex-shrink (valor por default: 1)**: Parecido a la propiedad “**flex-grow**”, pero esta propiedad reduce el tamaño de los elementos, o sea, se encojen para que no haya desbordamientos.

Sintaxis: “**flex-shrink: #número;**”

* **flex**: Es un shorthand que combina estas 3 propiedades que calculan el tamaño “**flex-grow**”, “**flex-shrik**” y “**flex-basis**”, y así podemos usarlos conjuntamente.

Sintaxis: “**flex: #número 1 #número 2 #número 3**”

1.- Explicación, por medio de un ejemplo, para el uso de “**flex-grow**” **sin usar “flex-grow”** sobre el flex ítem:

* Espacio total del contenedor padre: 500px
* Espacio ocupado por cada elemento: 100px (4 ítems = 400px ocupando)
* Espacio disponible: 100px

2.- **Usando** **“flex-grow: 1;”** sobre el flex ítem:

* El tamaño de los elementos ha crecido hasta llenar el espacio que estaba disponible.

3.- **Usando** “**flex-grow: 2;**” sobre alguno de estos flex ítems:

* El tamaño de este elemento ha crecido un poco más que el resto (no tiene el doble de tamaño, o sea, 200... NO ES EL DOBLE).

**NOTA: De acuerdo a la cantidad de elementos que estén usando la propiedad “flex-grown” se harán las operaciones de división, multiplicación y suma.**

El número que se coloca como valor en “**flex-grow**” es el **factor de crecimiento**, por ejemplo:

* El elemento 1 tiene factor de crecimiento 1
* El elemento 2 tiene factor de crecimiento 2
* El elemento 3 tiene factor de crecimiento 1
* El elemento 4 tiene factor de crecimiento 1
* Suma total de “**flex-grow**” de todos los ítems = 5
* Entonces, el **espacio disponible** (100px) o **unidad de crecimiento** se está dividiendo entre 5 (100px/5=20px)
* Entonces 20 es la **unidad de crecimiento** que se multiplica con el **factor de crecimiento** (20\*1=20) (20\*2=40)
* Entonces 20 + 100px (espacio individual del elemento) = 120px
* *.fb-item:nth-child(1){*

*flex-basis: 200px; // Este elemento siempre tendrá la medida de 200px, aunque se estire la ventana del navegador*

*flex-grow: 0; // Con esto se ordena que nunca crezca, el factor de crecimiento es 0*

*}*

* *.fb-item:nth-child(2){*

*flex-grow: 1; // No se necesita poner un “****flex-basis****” para este elemento, pues el tamaño que toma relativo al espacio disponible*

*}*

Utilizando las mismas propiedades y los mismos valores del ejemplo que arriba se citó, ahora veamos el mismo procedimiento al utilizar “**flex-shrink**”:

* 500px (contenedor padre) - 800px (200px por item \* 4 items) = -300px (Espacio disponible (negativo))
* Suma de “**flex-shrink**” 5 (mismo procedimiento que en flex-grow)
* Unidad de reducción -300px/5 = -60px \* por el factor que tenga el item (1\*-60px=-60) (200px-60=-140)

**REFERENCIAS**

* Muchos Vídeos de EDteam relacionados con los temas.
* Algunas páginas que almaceno en mis marcadores de Google Chrome, en la sección Relacionado a páginas web>Utilidades web>Consejos Prácticos>Entender Flexbox.
* Vídeos de otros profesores que están en Youtube, y cuyos vídeos almaceno en mi disco duro.