Flexbox CSS3. Tutorial en español. Descripción y ejemplos HTML.

30 marzo, 2017

Hasta hace relativamente poco, podríamos incluso decir "todavía" si tu desarrollo requiere de una alta compatibilidad en versiones antiguas de navegadores, maquetar módulos adaptables para web tales como listados en filas de N bloques suponía crear una larga lista de estilos css que cambiaban según la resolución de los diferentes dispositivos. En ocasiones "flotábamos" los elementos y nos las veíamos con los márgenes laterales, en otras aplicábamos "inline-block" y utilizábamos alguna técnica extraña para no contar con la separación que se creaba entre capas por arte de magia, añadíamos una capa vacía al final del listado para poder justificar el conjunto, y un largo etcétera de desventajas que Flexbox viene a solucionar.

Qué es Flexbox y cómo se utiliza.

Flexbox viene de "*Flexible Box Layout*", que se puede traducir como "*Diseño de caja flexible*", y nos aporta una magnífica solución para todos nuestros desarrollos "responsive". Lo que nos permite es crear un conjunto de elementos flexibles que se adaptan automáticamente a su contenedor y con el que podemos controlar parámetros tales como la alineación, dirección (horizontal/vertical), ajuste de la fila según tamaños y multitud de posibilidades que vamos a presentar en este artículo.

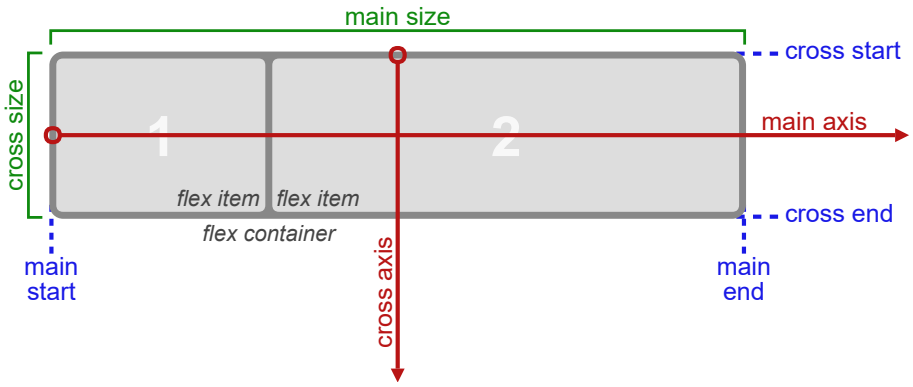
En cuanto a compatibilidad, Flexbox es actualmente **compatible con los navegadores web más importantes** y diferentes versiones de los mismos, si bien en algunos debemos utilizar prefijos css para su buen funcionamiento (ver en caniuse.com). Si tenéis problemas con algunas versiones antiguas, podéis probar a utilizar "Autoprefixer". En los siguientes ejemplos utilizaremos el código css tal cual, sin añadidos, así que espero que vuestro navegador esté actualizado.

La última definición conocida sobre Flexbox del W3C es del 26 de Mayo de 2016 en el momento de escribir este artículo. En el enlace podemos ver su definición y diferentes ejemplos de uso. Como se indica en la introducción, hasta ahora conocíamos cuatro modos de disponer los elementos, tres de presentación (con variaciones alternativas) y otro de posición:

- Elementos en línea (display:inline)
- Elementos en bloque (display:block)
- Elementos en tabla (display:table)
- Tipos de posición (position:absolute/relative/fixed)

Flexbox es una mezcla de todos ellos en cuanto a cómo afecta a la disposición de una estructura de elementos contenidos en una capa padre o contenedor, y se define como "flex" en la propiedad "display" (display:flex). Podemos crear una estructura de elementos en línea similar a una tabla, o hacer que funcionen como un bloque y en orden inverso en la siguiente resolución, todo ello a la vez que se adapta automáticamente al tamaño que necesitamos, tanto en anchura como en altura. Sí, ¡¡también en altura!! 😉

El W3C define a esta estructura como un "*flujo flexible*" de elementos en dirección (arriba/abajo/izquierda/derecha) y tamaño (anchura/altura) según los **ejes principal (horizontal) y transversal (vertical)**.



Como para profundizar en su definición ya os he añadido en enlace del W3C, pasaremos directamente a ver ejemplos de funcionamiento.

Ejemplos de maquetación utilizando Flexbox.

IMPORTANTE: Si no visualizas correctamente los siguientes ejemplos, lo más probable es que tu navegador no sea compatible con esta propiedad CSS. Por favor, prueba a actualizarlo o visualizarlo en otro navegador antes de increpar al autor. 😛

Para poder probar todas las posibilidades que ofrece Flexbox CSS, vamos a crear una capa **class="contenedor"** que hará de padre y siete capas **class="elemento"**numeradas. ¿Y por qué siete? Para los ejemplos nos viene bien que sea un número impar para ver tanto la ordenación como la adaptación y su resultado, y siete también para ver un listado de elementos no muy corto, aunque podéis probar a colocar cualquier número de elementos.

HTML:

```
<div class="contenedor">
    <div class="elemento">1</div>
    <div class="elemento">2</div>
    <div class="elemento">3</div>
    <div class="elemento">4</div>
    <div class="elemento">5</div>
    <div class="elemento">6</div>
    <div class="elemento">7</div>
</div>
```

Más allá de los estilos de diseño definidos para la visualización de los ejemplos, vamos a colocar de inicio un tamaño de anchura del 25% para los elementos en relación al contenedor padre. Para comenzar a utilizar Flexbox añadimos al contenedor la propiedad **"display:flex"**

CSS EJEMPLO 1:

```
.contenedor{
    display:flex;
}
.elemento{
    width:25%;
}
```

EJEMPLO 1:

Como vemos, al no haber definido aún el comportamiento de dirección y tamaño que tendrán los elementos de nuestro contenedor, aunque hayamos definido una anchura de elementos del 25% éstos se adaptan a su padre ocupando el 100% de anchura entre la suma de todos. Por defecto, tiene ese comportamiento "flexible" como indica su nombre. Pero Flexbox es mucho más.

flex-direction:

Vamos a ver la propiedad "**flex-direction**", que puede tomar 4 valores y se aplica al padre (contenedor):

– **flex-direction:row;** -> Los elementos se visualizan **de izquierda a derecha**(valor por defecto, similar al ejemplo 1)

– **flex-direction:row-reverse;** -> Los elementos se visualizan **de derecha a izquierda**.

– **flex-direction:column;** -> Los elementos se visualizan **de arriba hacia abajo**.

– **flex-direction:column-reverse;** -> Los elementos se visualizan **de abajo hacia arriba**.

CSS EJEMPLO 2:

```
.contenedor{
    display:flex;
    flex-direction:row-reverse;
}
```

EJEMPLO 2:

Los elementos invierten su orden de visualización de derecha a izquierda, sin tener en cuenta el orden de la maquetación.

CSS EJEMPLO 3:

```
.contenedor{
    display:flex;
    flex-direction:column;
}
```

EJEMPLO 3:

Los elementos se visualizan formando una columna de arriba hacia abajo. En este caso, como los elementos miden el 25% del contenedor, su anchura sí que toma el valor indicado. Si eliminamos la anchura, las capas se ajustarán al tamaño del contenedor, ocupando el 100% de anchura.

CSS EJEMPLO 4:

```
.contenedor{
    display:flex;
    flex-direction:column-reverse;
}
```

EJEMPLO 4:

Los elementos se visualizan formando una columna de abajo hacia arriba, sin tener en cuenta el orden de la maquetación. La anchura funciona de forma similar al ejemplo 3.

flex-wrap:

A continuación vamos a ver la propiedad "**flex-wrap**", cuyo valor afecta a cómo se distribuyen los elementos en fila y, por consiguiente, a su tamaño. Los posibles valores son:

– **flex-wrap:nowrap;** -> Los elementos se muestran en línea, en una sola fila, y su tamaño se ajusta al contenedor siempre y cuando la suma de todos ellos sea mayor o igual que el 100% de la anchura del contenedor. Si es inferior, se siguen mostrando en línea pero conservan su tamaño. Este es el valor por defecto, y como veíamos en el *ejemplo 1*, aunque la anchura de los elementos es el 25% del contenedor, todos se muestran en línea modificando su tamaño para que la suma total no sea superior al 100% de su contenedor.

– flex-wrap:wrap; -> Los elementos se muestran en línea, pero si su anchura supera la del contenedor, se distribuyen en varias filas.

– flex-wrap:wrap-reverse; -> Los elementos se muestran en línea, pero si su anchura supera la del contenedor, se distribuyen en varias filas, y además lo hacen en orden inverso al de maquetación.

CSS EJEMPLO 5:

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

EJEMPLO 5:

Como comentábamos en los valores de la propiedad, en este caso los elementos se muestran en línea, pero como habíamos otorgado un valor de anchura del 25% del contenedor a los elementos, cuando la fila termina los elementos continúan en otra, y así sucesivamente.

CSS EJEMPLO 6:

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

EJEMPLO 6:

Este ejemplo de ordenación de Flexbox es muy curioso, ya que, además de realizar la adaptación de tamaños por filas como en el caso anterior, realiza su ordenación a la inversa, de abajo hacia arriba, y en este caso de izquierda a derecha (que es el sentido por defecto).

Podemos modificar el sentido de la orientación horizontal si además utilizamos "**flex-direction:row-reverse**" como podemos ver en el siguiente ejemplo:

CSS EJEMPLO 7:

```
.contenedor{
    display:flex;
    flex-direction:row-reverse;
    flex-wrap:wrap-reverse;
}
```

EJEMPLO 7:

flex-flow:

Estas mismas propiedades podemos especificarlas en una sola mediante "**flex-flow**" de la siguiente forma:

flex-flow: <flex-direction> <flex-wrap>

Por lo que el código anterior podría quedar de la siguiente forma:

CSS:

```
.contenedor{
    display:flex;
    flex-flow:row-reverse wrap-reverse;
}
```

justify-content:

En cuanto a la **alineación horizontal** de los elementos en Flexbox, encontramos la propiedad "**justify-content**", que alinea los elementos a lo largo del eje principal (main axis) de su contenedor, pero a diferencia de la alineación de un texto, en Flexbox hay que tener en cuenta también la dirección de los elementos. Esto lo veremos más claramente con los siguientes ejemplos tras la definición de sus valores más utilizados:

– **justify-content:flex-start;** -> Alinea los elementos en horizontal desde el inicio de la dirección del eje principal de su contenedor (partiendo desde el inicio de la línea). Este es el valor por defecto. Es importante destacar que, como veremos más adelante, **la dirección establecida en "flex-direction" afecta a la alineación**.

– **justify-content:flex-end;** -> Alinea los elementos en horizontal desde el final de la dirección del eje principal de su contenedor (partiendo desde el final de la línea)

– **justify-content:center;** -> Alinea los elementos al centro del eje principal de su contenedor. Similar a un texto alineado al centro.

– **justify-content:space-between;** -> Alinea los elementos justificándolos a lo largo del eje principal de su contenedor. Similar a un texto justificado. Los elementos laterales se pegan a los extremos y el resto se distribuyen a lo largo del eje principal dejando el mismo espacio entre ellos.

– **justify-content:space-around;** -> Alinea los elementos distribuyendo sus centros de forma horizontal a lo largo del eje principal de su contenedor, dejando el mismo espacio lateral de separación al comienzo, al final y entre ellos.

Esta propiedad como mejor se comprende es mediante un ejemplo, ya que si probamos "**justify-content:flex-start**" con el resto de propiedades por defecto, obtendremos un resultado similar a la alineación de un texto a la izquierda. Pero… ¿y si la dirección es inversa? En este caso obtendríamos que la alineación sería a la derecha, ya que **lo que Flexbox toma en cuenta para la alineación de los elementos es el inicio de la dirección del eje** (ya sea principal –horizontal- o transversal –vertical-, como veremos más tarde).

CSS EJEMPLO 8:

```
.contenedor{
    display:flex;
    flex-direction:row;
    flex-wrap:wrap;
    justify-content:flex-start;
}
```

EJEMPLO 8:

En este caso vemos que la alineación de los elementos es a la izquierda de su contenedor. Ahora vamos a invertir la dirección de los mismos mediante "**flex-direction:row-reverse**" y veremos lo que ocurre:

CSS EJEMPLO 9:

```
.contenedor{
    display:flex;
    flex-direction:row-reverse;
    flex-wrap:wrap;
    justify-content:flex-start;
}
```

EJEMPLO 9:

Vemos que en este caso los elementos se alinean a la derecha. Se puede llegar a pensar que "**justify-content:flex-start**" realmente no está haciendo nada, ya que en los *ejemplos 1 y 2* ya obteníamos estos resultados sin necesidad de esta propiedad, pero… ¿y si queremos que la dirección de los elementos sea inversa pero estén alineados a la izquierda? Si cambiamos la propiedad a "**justify-content:flex-end**" conseguimos dicho resultado, comprobando que realmente funciona:

CSS EJEMPLO 10:

```
.contenedor{
    display:flex;
    flex-direction:row-reverse;
    flex-wrap:wrap;
    justify-content:flex-end;
}
```

EJEMPLO 10:

A continuación comprobamos mediante ejemplos los valores de alineación horizontal que ya hemos visto, realizando modificaciones en la anchura de los elementos para que se puedan comprender mejor sus efectos. Primero una alineación al centro:

CSS EJEMPLO 11:

```
.contenedor{
    display:flex;
    flex-direction:row;
    flex-wrap:wrap;
    justify-content:center;
}
.elemento{
    width:21%;
}
```

EJEMPLO 11:

Ahora veremos una alineación "justificada". La separación entre la línea superior e inferior es diferente debido al número de elementos entre ellas:

CSS EJEMPLO 12:

```
.contenedor{
    display:flex;
    flex-direction:row;
    flex-wrap:wrap;
    justify-content:space-between;
}
.elemento{
    width:21%;
}
```

EJEMPLO 12:

Y ahora una alineación distribuyendo sus centros horizontales. En este caso podemos ver que se crea una especie de "márgenes laterales similares" entre los elementos (y entre líneas, dependiendo del número de elementos que existan en cada una de ellas).

CSS EJEMPLO 13:

```
.contenedor{
    display:flex;
    flex-direction:row;
    flex-wrap:wrap;
    justify-content:space-around;
}
.elemento{
    width:21%;
}
```

EJEMPLO 13:

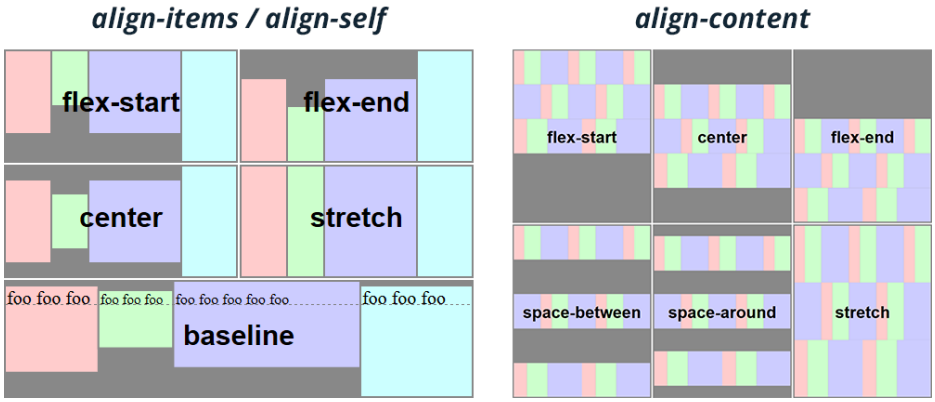
align-items | align-self | align-content:

La **alineación vertical** se realiza a través del llamado **"eje transversal" (cross axis)**, y para ello contamos con tres propiedades diferentes, "align-items", "align-self" y "align-content". Aquí viene el lío, ya que si no se entiende el concepto muchas veces se obtienen resultados inesperados. Vamos a ver qué nos cuenta el W3C sobre estas propiedades:

– **“align-items”** establece la alineación predeterminada para todos los elementos del contenedor, incluidos los elementos independientes.

– **“align-self”** permite alinear elementos independientes del contenedor.

– **“align-content”** alinea las líneas/filas de elementos de un contenedor.



Por lo tanto, “**align-items**” nos sirve para **alinear los elementos** y “**align-content**” para **alinear las filas** de éstos, mientras que “**align-self**” nos permite **alinear elementos de forma independiente**. Sabiendo esto, podemos deducir que “align-content” sólo funciona cuando tenemos más de una fila de elementos. La imagen superior representa muy bien cómo funcionan estas propiedades. A continuación veremos los valores que pueden tomar cada una de ellas (los valores de “align-items” son válidos para “align-self”):

– **align-items:stretch;** -> Valor por defecto. La altura de los elementos se ajusta al tamaño del contenedor (o fila), dividiendo el espacio sobrante entre todos los elementos por igual.

CSS EJEMPLO 14:

```
.contenedor{
    height:240px;
    display:flex;
    flex-flow:row wrap;
    justify-content:flex-start;
    align-items:stretch;
}
.elemento{
    width:21%;
}
```

EJEMPLO 14:

– **align-items:flex-start;** -> Alinea en vertical los elementos desde el inicio de la dirección del eje transversal de su contenedor (al igual que ocurría en horizontal). También **afecta el valor de “flex-direction” al sentido de la alineación vertical**.

CSS EJEMPLO 15:

```
.contenedor{
    height:240px;
    display:flex;
```

```
    flex-flow:row wrap;
    align-items:flex-start;
}
.elemento{
    width:21%;
}
```

EJEMPLO 15:

– **align-items:flex-end;** -> Alinea en vertical los elementos desde el final de la dirección del eje transversal de su contenedor (al igual que ocurría en horizontal).

CSS EJEMPLO 16:

```
.contenedor{
    height:240px;
    display:flex;
    flex-flow:row wrap;
    align-items:flex-end;
}
.elemento{
    width:21%;
}
```

EJEMPLO 16:

– **align-items:center;** -> Alinea al centro vertical los elementos a lo largo del eje transversal de su contenedor.

CSS EJEMPLO 17:

```
.contenedor{
    height:240px;
    display:flex;
    flex-flow:row wrap;
    align-items:center;
}
.elemento{
    width:21%;
}
```

EJEMPLO 17:

– **align-items:baseline;** -> Alinea en vertical las "líneas base" de los elementos a lo largo del eje transversal de su contenedor. Para poder comprender este ejemplo, vamos a añadir algunos estilos diferentes a los elementos:

CSS EJEMPLO 18:

```
.contenedor{
    height:240px;
    display:flex;
    flex-flow:row wrap;
    align-items:baseline;
}
.elemento{
    width:21%;
```

```
    line-height:2rem;
}
.elemento:first-child{
    line-height:4rem;
}
.elemento:nth-child(3){
    font-size:1rem;
    line-height:1.2rem;
}
.elemento:nth-child(6){
    font-size:3rem;
    line-height:5rem;
}
.elemento:last-child{
    line-height:3rem;
}
```

EJEMPLO 18:

Como podemos observar, lo que se alinea en vertical es la línea base del contenido de cada elemento, en este caso, la línea base de cada número, aunque entre ellos tengan alturas de línea o tamaños diferentes.

Pasamos a ver la descripción para los valores de la propiedad "align-content":

– **align-content:stretch;** -> Valor por defecto. La altura de las filas se ajustan al tamaño del contenedor, dividiendo el espacio sobrante entre todas las líneas por igual.

CSS EJEMPLO 19:

```
.contenedor{
    height:240px;
    display:flex;
    flex-flow:row wrap;
    align-content:stretch;
}
.elemento{
    width:21%;
}
```

EJEMPLO 19:

– **align-content:flex-start;** -> Alinea en vertical las filas desde el inicio de la dirección del eje transversal de su contenedor.

CSS EJEMPLO 20:

```
.contenedor{
    height:240px;
    display:flex;
    flex-flow:row wrap;
    align-content:flex-start;
}
.elemento{
    width:21%;
}
```

EJEMPLO 20:

– **align-content:flex-end;** -> Alinea en vertical las filas desde el final de la dirección del eje transversal de su contenedor.

CSS EJEMPLO 21:

```
.contenedor{
    height:240px;
    display:flex;
    flex-flow:row wrap;
    align-content:flex-end;
}
.elemento{
    width:21%;
}
```

EJEMPLO 21:

– **align-content:center;** -> Alinea al centro vertical las filas a lo largo del eje transversal de su contenedor.

CSS EJEMPLO 22:

```
.contenedor{
    height:240px;
    display:flex;
    flex-flow:row wrap;
    align-content:center;
}
.elemento{
    width:21%;
}
```

EJEMPLO 22:

– **align-content:space-between;** -> Alinea las filas justificándolas a lo largo del eje transversal de su contenedor. Similar a un texto justificado en vertical. Las filas superior e inferior se pegan a sus extremos y el resto se distribuyen a lo largo del eje transversal dejando el mismo espacio entre ellas.

CSS EJEMPLO 23:

```
.contenedor{
    height:240px;
    display:flex;
    flex-flow:row wrap;
    align-content:space-between;
}
.elemento{
    width:21%;
}
```

EJEMPLO 23:

– **align-content:space-around;** -> Alinea las líneas distribuyendo sus centros de forma vertical a lo largo del eje transversal de su contenedor, dejando el mismo espacio vertical de separación en la parte superior, inferior y entre ellos.

CSS EJEMPLO 24:

```css
.contenedor{
    height:240px;
    display:flex;
    flex-flow:row wrap;
    align-content:space-around;
}
.elemento{
    width:21%;
}
```

EJEMPLO 24:

Las propiedades más curiosas e interesantes de Flexbox.

Hasta ahora hemos visto propiedades que influyen en la dirección, alineación y tamaño de elementos y filas (y aunque no hemos entrado en detalle, columnas) de un contenedor Flexbox. Pero a continuación vamos a ver propiedades impresionantes, ya que con ellas podemos modificar elementos de forma independiente aumentando las posibilidades de presentación de webs e interfaces.

order:

Entre estas propiedades se encuentra "**order**", y es que como hemos podido comprobar con Flexbox no nos importa el orden de maquetación de nuestros elementos, y podemos alterar ese orden con tan sólo una instrucción. Que levante la mano quien no hubiera pagado por tener esta propiedad y olvidarse de wrap/unwrap de jquery 😉

– **order:** -> Posiciona un elemento en el orden asignado por el número entero especificado en la propiedad, teniendo en cuenta el valor asignado al resto de elementos, cuyo valor por defecto es 0 (cero). Esto significa que, a no ser que se especifique un orden a todos los elementos, cualquier número aplicado a un solo elemento lo llevará a la última posición. Esto es debido a que por defecto todos tienen "cero", por lo que si asignamos por ejemplo "order:3" a cualquier elemento, éste se colocará al final del listado. Más que un orden, podríamos llamarlo un "valor de posición de ordenación". El siguiente ejemplo puede servirnos para comprender mejor esta propiedad:

CSS EJEMPLO 25:

```css
.contenedor{
    display:flex;
}
.elemento:nth-child(even){
    order:1;
}
```

EJEMPLO 25:

Con este ejemplo podemos entender mejor el funcionamiento de esta propiedad, y es que estamos asignando el orden/posición "1" a todos los elementos pares, que se sitúan tras los elementos impares, que tendrían por defecto el valor "0".

flex:

A continuación veremos la propiedad "**flex**" que se aplica a los elementos del contenedor y que a su vez se estructura de tres propiedades diferentes y que se pueden presentar por separado, **"flex- grow", "flex-shrink" y "flex-basis"**, y se define de la siguiente forma:

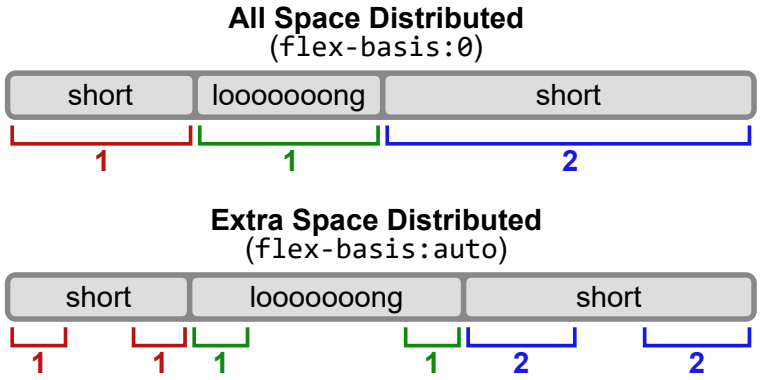
– **flex: none | <flex-grow> <flex-shrink> <flex-basis>**

Vamos a ver las propiedades por separado:

– flex-grow:<número>; -> Determina el factor de incremento de tamaño de uno o varios elementos en relación al resto teniendo en cuenta el espacio "vacío" o "libre" entre ellos. Como espacio "vacío/libre" nos referimos al espacio que no ocupan los elementos en relación a su fila. Cuando este valor se omite, se establece en 1.

– flex-shrink:<número>; -> Determina el factor de disminución de tamaño de uno o varios elementos en relación al resto teniendo en cuenta el espacio "vacío" o "libre" entre ellos. Cuando este valor se omite, se establece en 1.

– flex-basis: auto | 0 | <width> -> Especifica el tamaño principal inicial de los elementos teniendo en cuenta el espacio "vacío" o "libre" entre ellos. Cuando este valor se omite, se establece en 0. Esta propiedad hace que, dependiendo del valor asignado, el tamaño "base" se interprete como "**todo el espacio**" del elemento o como el "**espacio sobrante**" del mismo. A continuación vemos un gráfico donde visualmente se explica cómo se interpretan estos valores en relación al tamaño del elemento.



Esta propiedad "flex" puede funcionar de diferentes formas dependiendo del valor que establezcamos y no podemos presentar todos los posibles ejemplos que pueden darse. Lo mejor en este caso es que probéis directamente a modificar sus valores con una maquetación básica y así poder observar su comportamiento. Nosotros os ofrecemos un punto de partida como el del siguiente ejemplo con el que podéis comenzar a jugar con el inspector de contenido de vuestro navegador:

CSS EJEMPLO 26:

```
.contenedor{
    display:flex;
}
.elemento{
    flex:1 1 0;
}
.elemento:nth-child(4){
```

```
8       flex-grow:3;
9   }
```

EJEMPLO 26:

Flexbox. La solución óptima para el diseño adaptable.

Como hemos podido ver en este artículo, Flexbox es el complemento perfecto para la maquetación de diferentes módulos de contenido en lista y mediante pequeñas propiedades poder cambiar su aspecto, distribución, tamaño y alineación para que se adecúe sin problemas al dispositivo donde lo estemos visualizando, pero esto es sólo un pequeño ejemplo de todo lo que podemos desarrollar. Con imaginación y práctica se puede aplicar a multitud de casuísticas y reducir muy notablemente el tiempo de desarrollo. Espero que sirva a des/conocedores de esta técnica CSS como guía de consulta. ¡Saludos!