Contenedor flexible

Propiedades del Contenedor

Flex-flow

Las propiedades disponibles para un *contenedor flexible* son `flex-direction: row, row-reverse, column, column-reverse` y `flex-wrap: wrap, wrap-reverse, nowrap`.
Las dos propiedades se combinan en una única llamada **flex-flow**, que contendrá dos valores separados por un espacio. El primero hace referencia a la propiedad *direction*, y el segundo a la propiedad *wrap*.
Se puede decir que si la propiedad es *wrap* la anchura de los elementos contenidos manda sobre el tamaño o anchura del contenedor; y si es *nowrap* la anchura del contenedor fuerza a los elementos contenidos a modificar su anchura. En este ejemplo vemos el resultado de la propiedad `flex-flow: row nowrap;`

Ejemplo

En todos los ejemplos que vamos a ver a continuación vamos a trabajar sobre un elemento *ul* con 6 elementos *li*. El **ul** será nuestro contenedor flex, y los **li** nuestros elementos flexibles. Vamos a ver cuál es el comportamiento de estos elementos según las diferentes propiedades *flex* que definamos sobre ellos.

El HTML para los ejemplos es:

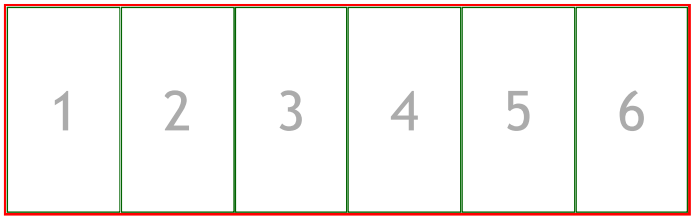
```
<ul class="contenedor-flex" id="caso1">
        <li class="elem_flex">1</li>
        <li class="elem_flex">2</li>
        <li class="elem_flex">3</li>
        <li class="elem_flex">4</li>
        <li class="elem_flex">5</li>
        <li class="elem_flex">6</li>
</ul>
```

En cada ejemplo modificaremos el atributo *id* del contenedor para definir en él sus propiedades flex específicas.

Propiedad flex por defecto

Cuando definimos un elemento flex sin especificar ninguna propiedad adicional, su valor predeterminado para *flex-flow* es, com hemos dicho `row nowrap`, y este es el resultado

```
.contenedor-flex {
        border: 2px solid red;
        display:flex;
        width: 500px;
        margin: auto;
        background-color:#FFF;
}
.elem_flex {
        width:150px;
        height: 150px;
        border: 1px solid black;
        background-color: green;
        list-style: none;
        box-sizing:border-box;
        text-align: center;
        line-height: 150px;
        font-size: 2.5rem;
        color: white;
}
```

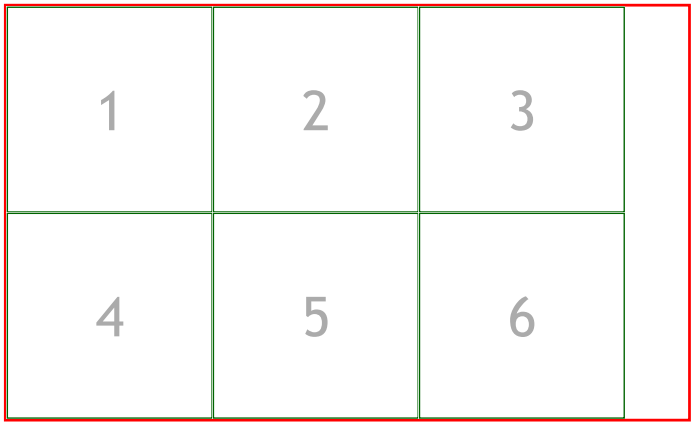


Como vemos, el contenedor flex mantiene la anchura asignada por CSS y hace que sus elementos flexibles anidados se ajusten a esa anchura, quedando todos en la misma línea e ignorando su anchura específica, en este caso de 150px. Este comportamiento viene dado por la propiedad *nowrap* del contenedor flex.

Cambiando a *row wrap*

En este caso vemos como queda el mismo elemento con la propiedad `flex-flow: row wrap;`

```
#caso2 {
        flex-flow: row wrap;
}
```

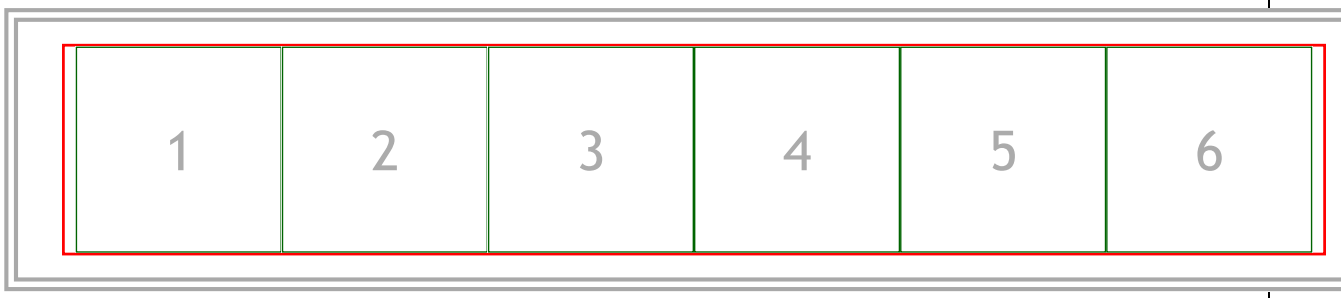


En este caso la anchura del contenedor flex se mantiene también, pero ahora además se conserva la anchura de los elementos flex aniadados, lo que hace que los elementos que no caben en una primera fila se distribuyan en filas adicionales, aumentando la altura del contenedor flex.

Justify-content

En este tercer caso vemos el atributo **justify-content**, que permite distribuir de manera uniforme los elementos contenidos dentro del contenedor flexible. Sus valores principales son: `flex-start, flex-end, center, space-around, space-between y space-evenly`

```
#caso3 {
        flex-flow: row nowrap;
        width: 920px;
        justify-content:center;
}
```



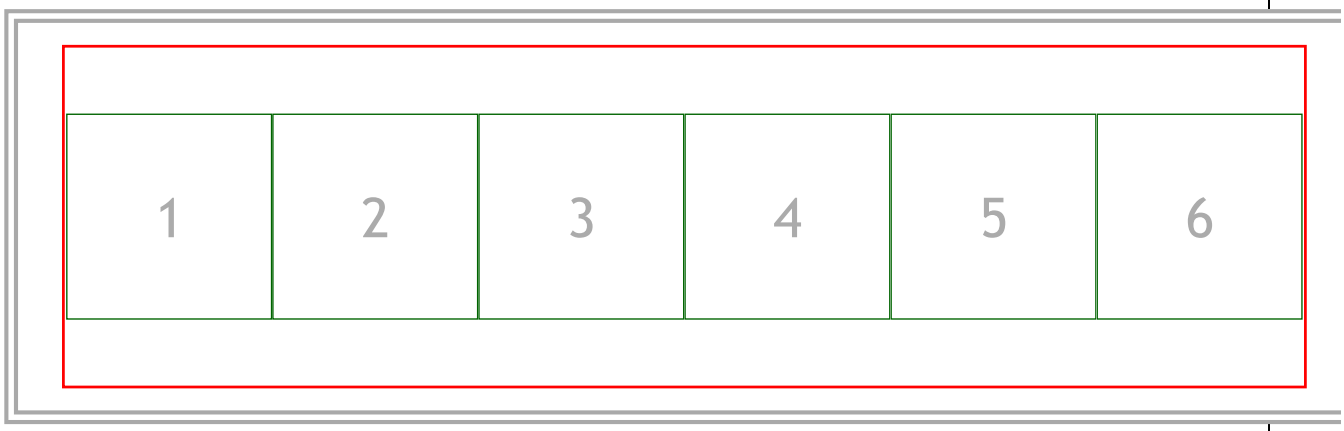
Aquí vemos una de las propiedades más interesantes y útiles de **flex**, la posibilidad de distribuir de forma uniforme los elementos dentro del contenedor, sin necesidad de realizar cálculos. Los diferentes valores de esta propiedad hacen que el navegador realice por nosotros los cálculos necesarios para alinear y distribuir adecuadamente los elementos. En este ejemplo los elementos quedan alineados al centro del contenedor, dejando el espacio sobrante a ambos lados de forma uniforme.

Align-items

En este cuarto caso vemos el atributo **align-items**, que permite distribuir de manera uniforme los elementos contenidos dentro del contenedor flexible, pero a través de su eje transversal, en lugar de usar el principal como en el caso anterior. Sus valores son: `flex-start, flex-end, center, baseline, stretch`.
El valor *baseline* alinea todos los elementos por su línea base del texto que contengan cada uno de ellos, sin tener como referencia la esquina superior izquierda del contenedor.
Y el valor *stretch* "estira" todas las alturas (o anchuras,

dependiendo de la dirección del contenedor) para que cubran todo el espacio disponible dentro del contenedor. Pero siempre que los elementos del contenedor no tengan definida una altura o anchura, que en ese caso sobrescribiría el valor de stretch.
Esta propiedad se aplica a cada una de las filas que pueda tener el contenedor, alineando los elementos respecto a la fila que ocupan dentro del contenedor, pero no tratando todos los elemenentos como un único bloque. Para este segundo caso recurrimos a **align-content**

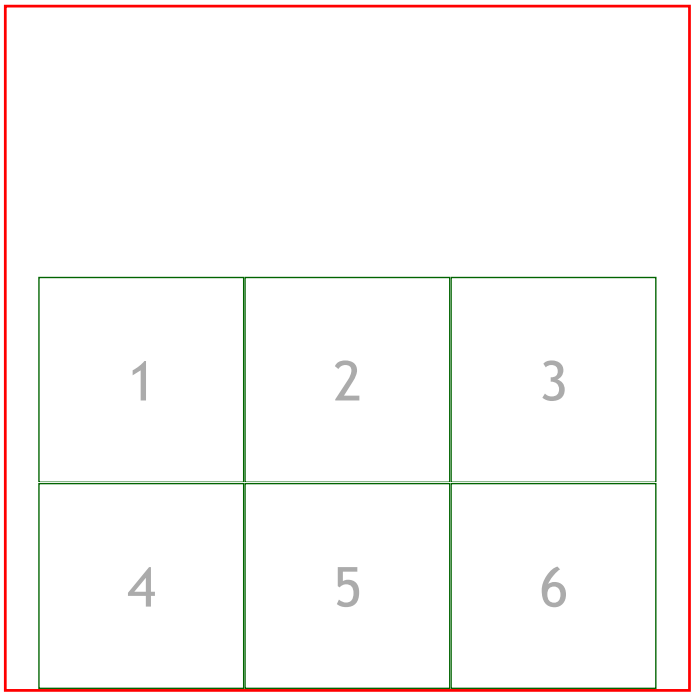
```
#caso4 {
  align-items: center;
  flex-flow: row wrap;
  height: 250px;
  justify-content: center;
  width: auto;
}
```



Align-content

Esta propiedad permite alinear todo el contenido del contenedor respecto al eje secundario o transversal, tratándolo como un único bloque.
Para que la propiedad funcione, el contenido tiene que estar distribuido en más de una fila, de lo contrario no tendrá efecto, y sería suficiente con trabajar con la propiedad **align-items**. De hecho, si el valor asignado a *align-content* es compatible con alguno de los valores de *align-items*, se comportará como si fuera esa la propiedad aplicada. Sus principales valores son: `flex-start, flex-end, center, space-around, space-between, space-evenly`.

```
#caso5 {
  align-content: flex-end;
  flex-flow: row wrap;
  height: 500px;
  justify-content: center;
  width: 500px;
}
```



Propiedades del contenido

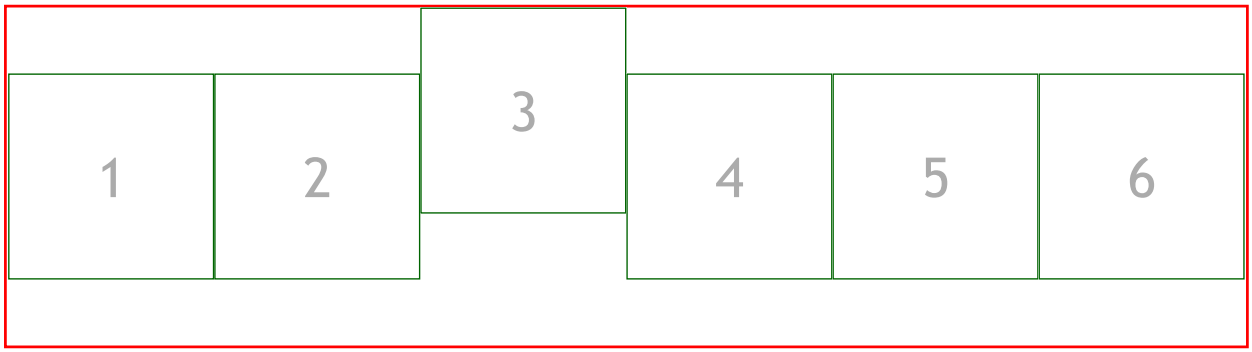
Aunque el comportamiento de los elementos anidados del contenedor flexible puede ser manejado desde las mismas propiedades del contenedor, existe un conjunto de propiedades específicas para manejar el contenido

Propiedad align-self

Esta propiedad permite cambiar el ajuste recibido por *align-items* para un elemento en concreto, haciendo que modifique su posición respecto al resto de elementos flex del contenedor.

Sus valores son los mismos que usamos para la propiedad *align-items* pero en este caso sólo afectan al elemento seleccionado

```
#caso4b {
  align-items: center;
  flex-flow: row wrap;
  height: 250px;
  justify-content: center;
  width: auto;
}
#caso4b li:nth-child(3) {
        align-self: flex-start;
}
```



Propiedad Order

Con este atributo podemos definir el orden de presentación de los elementos en el navegador. Acepta valores numéricos, y de manera predeterminada todos los elementos del mismo contenedor **flex** tienen asignado el valor 0 (cero) para esta propiedad. Por lo que debemos tener en cuenta que para modificar el orden de un único elemento del contenedor sería necesario definir esta propiedad para todos los elementos dentro del contenedor. Otra opción, si solamente queremos hacer que un elemento pasa por delante de todos los demás es asignarle un valor negativo a ese elemento. Incluso sería posible alterar el orden de todos los elementos del contenedor usando únicamente valores negativos.

```
#caso6 li:first-child {
        order: 5;
}
#caso6 li:nth-child(2) {
        order: 6;
}
#caso6 li:nth-child(3) {
        order: 4;
}
#caso6 li:nth-child(4) {
        order: 1;
}
#caso6 li:nth-child(5) {
        order: 3;
}
#caso6 li:last-child {
        order: 2;
}
#caso6 li:hover{
        flex:1 0 auto;
        background-color: blue;
}
```

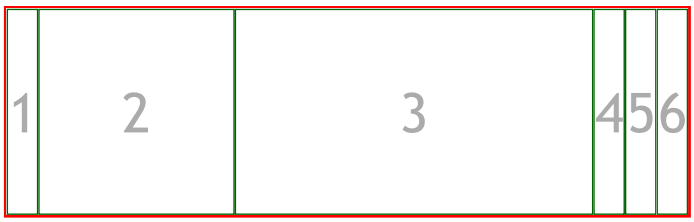
Propiedad Flex

Esta propiedad reune tres atributos diferentes, relacionados los tres con el tamaño del elemento contenido, especialmente con la forma en la que cambia ese tamaño, creciendo o encogiéndose para ajustarse al contenedor.
Las tres propiedades son `flex-grow, flex-shrink y flex-basis`.

flex-grow

Esta propiedad define cómo debe crecer el elemento respecto a los demás cuando hay espacio extra disponible en el contenedor. Se define con valores númericos que hacen referencia a la porción del espacio adicional que debe ocupar. Así un elemento con este valor definido como 1, cuando ninguno más lo tiene definido, significa que ese elemento se "apropia" de todo el espacio disponible.
En cambio, si un elemento tiene esta propiedad con valor "1" y otro la tiene con valor "2", quiere decir que el primero se queda con un tercio de ese espacio y el segundo con los dos tercios restantes. Su valor predeterminado es 0.

```
#caso7 li:nth-child(2) {
        flex: 1 1 auto;
}
#caso7 li:nth-child(3) {
        flex: 2 1 auto;
}
```

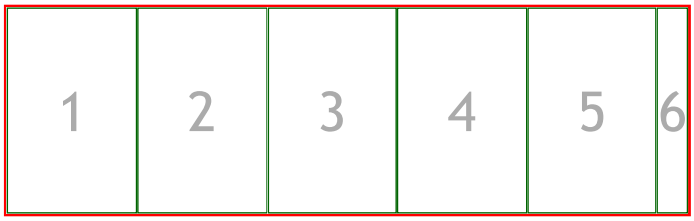


flex-shrink

Esta propiedad define cómo debe encogerse el elemento respecto a los demás cuando el contenedor es más pequeño que el tamaño total de los elementos (y está definido como *nowrap*). Se define con valores númericos que hacen referencia a la porción del espacio que deben perder entre todos para ajustarse al tamaño del contenedor. Por defecto, todos los elementos se reducen de forma equitativa (valor "1"), pero podemos definir qué proporción pierde cada uno de ellos de manera similar al caso anterior.

```
#caso8 li {
        width: 150px;
}
#caso8 li:last-child {
```

```
        flex: 1 3 auto;
}
```



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.

```
#caso9 li:nth-child(3) {
        flex-basis: 50%;
}
```

EJEMPLO

Dada esta vista inicial de una lista:

```
<ul class="flexible">
        <li class="elemento-fl" id="uno">1</li>
        <li class="elemento-fl" id="dos">2</li>
        <li class="elemento-fl" id="tres">3</li>
        <li class="elemento-fl" id="cuatro">4</li>
        <li class="elemento-fl" id="cinco">5</li>
        <li class="elemento-fl" id="seis">6</li>
```

```
        </ul>
```

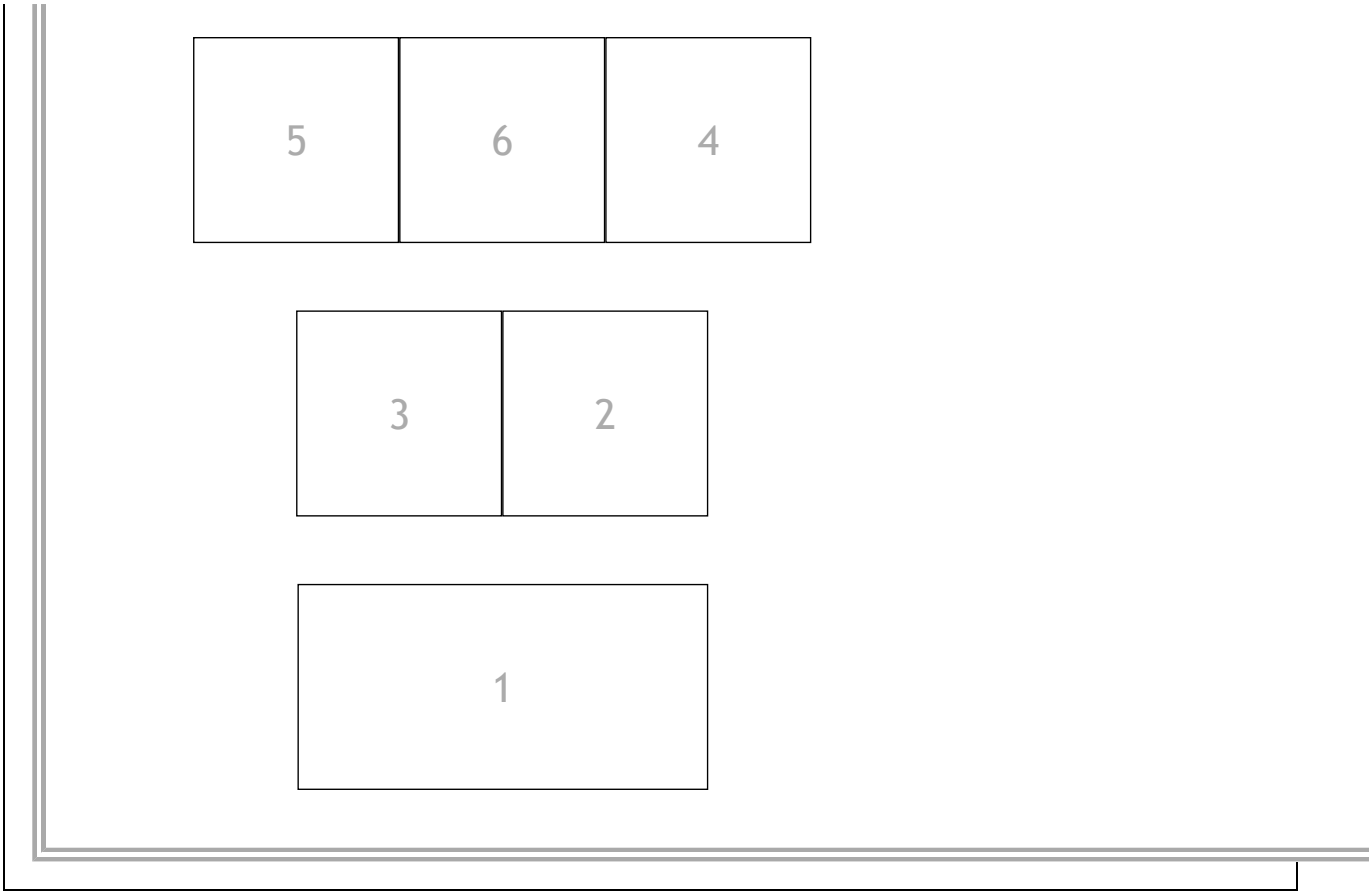
```
1
2
3
4
5
6
```

Obtenemos un resultado muy diferente con este CSS

```
.flexible {
        border:1px solid;
        width: 600px;
        height: 600px;
        display: flex;
        flex-flow: row wrap;
        justify-content: center;
        align-items: center;
}
.elemento-fl {
        width: 150px;
        height: 150px;
        background-color: lime;
        list-style: none;
        font-size: 30px;
        color: white;
        text-align: center;
        line-height: 150px;
        border: 1px solid black;
}
#uno {
        order:6;
        flex-basis: 50%;
}
#dos {
        order:5;
}
#tres {
        order:4;
}
#cuatro {
        order:3;
}
#cinco {
        order:1;
}
#seis {
        order: 2;
}
```

Si además añadimos la pseudo-clase *hover* a los elementos y al contenedor, tendremos un comportamiento muy diferente al pasar el ratón sobre ellos

```
.flexible:hover {
        flex-direction: column;
        justify-content: space-between;
}
.elemento-fl:hover {
        flex-grow: 1;
        background-color: red;
}
```


5
6
4
3
2
1