CSS Flexbox

Flexbox permite de manera muy simple, acomodar elementos dentro del body, este facilita la creación del layout de la página web. Con flexbox, prácticamente no necesitas un framework de CSS para temas de diseño responsivo.

Flex box es una forma simple y efectiva para crear el layout de un documento a través de propiedades que permiten definir la distribución del espacio, alineación de contenido y como los elementos son ordenados visualmente.
El contenido puede ser organizado fácilmente de manera vertical y horizontal y cada elemento puede ser independiente del orden inicialmente definido.

Propiedad Display Flex (Contenedor Flex)

Flexbox funciona por medio de la relación padre e hijo (parent & child). Una vez se haya definido un contenedor padre con la propiedad **display:flex** o **display:inline-flex** los elementos hijo son controlados por esta propiedad.

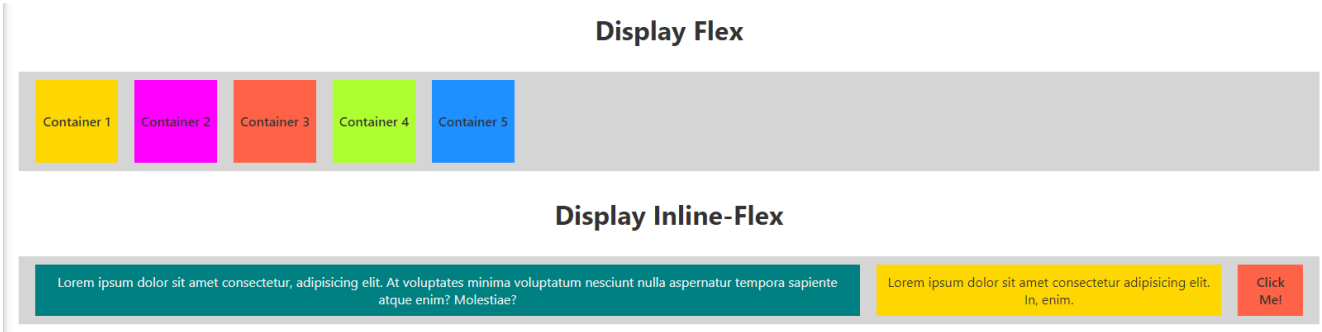
Código HTML

```
<body>
  <h1>Display Flex</h1>
  <div class="flex-container">
    <div class="child1">Container 1</div>
    <div class="child2">Container 2</div>
    <div class="child3">Container 3</div>
    <div class="child4">Container 4</div>
    <div class="child5">Container 5</div>
  </div>
  <h1>Display Inline-Flex</h1>
  <div class="flex-inline-container">
    <p class="inline-child in-child1">
      Lorem ipsum dolor sit amet consectetur, adipisicing elit. At voluptates
      minima voluptatum nesciunt nulla aspernatur tempora sapiente atque enim?
      Molestiae?
    </p>
    <p class="inline-child in-child2">
      Lorem ipsum dolor sit amet consectetur adipisicing elit. In, enim.
    </p>
    <a class="btn inline-child" href="#">Click Me!</a>
  </div>
</body>
```

Código CSS

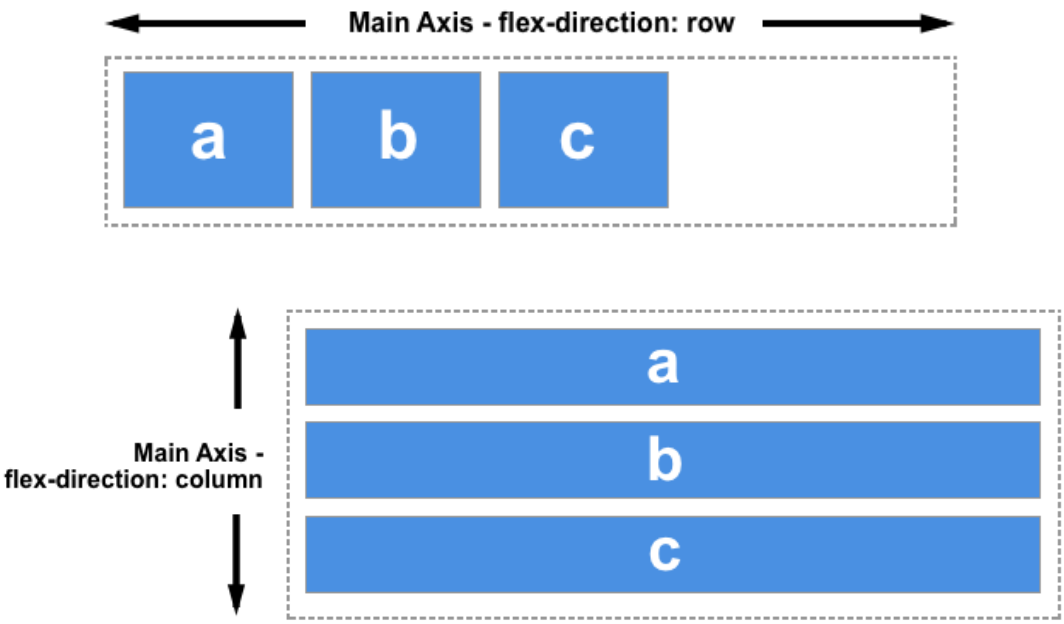
```css
* {
  margin: 0;
  padding: 0;
}

body {
  color: #333;
  font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
}

h1 {
  text-align: center;
  padding: 10px;
}

.btn {
  display: inline-block;
  background: tomato;
  padding: 10px 15px;
  text-align: center;
  text-decoration: none;
  color: inherit;
  font-weight: 500;
}

.in-child1 {
  background-color: teal;
  color: #f3f3f3;
}
.in-child2 {
  background-color: gold;
}

.child1,
.child2,
.child3,
.child4,
.child5 {
  width: 100px;
  height: 100px;
  line-height: 100px;
  text-align: center;
  font-weight: 500;
  margin: 0 10px;
}

.child1 {
```

```
  background-color: gold;
}
.child2 {
  background-color: magenta;
}
.child3 {
  background-color: tomato;
}
.child4 {
  background-color: greenyellow;
}
.child5 {
  background-color: dodgerblue;
}

.inline-child{
    text-align: center;
    margin: 0 10px;
    padding: 10px;
}

.flex-container {
  background-color: rgb(213, 213, 213);
  display: flex;
  margin: 20px;
  padding: 10px;
}

.flex-inline-container {
  background-color: rgb(213, 213, 213);
  display: inline-flex;
  margin: 20px;
  padding: 10px;
}
```



Ahora, los elementos se han convertido en ítems flex. Note que cuando son elementos inline, se acomodan a lo largo del viewport a diferencia de los elementos block que se le han asignado valores width y height, respectivamente.

Flexbox es un modelo unidimensional, esto quiere decir que solo maneja una dimensiona la vez, ya sea filas o columnas, contrario a lo que hace Grid que controla filas y columnas a la vez.
Cuando se trabaja con flexbox, es común ver el termino 'main-axis' (eje principal). Este eje principal es definido por dos direcciones, las filas y las columnas que son controladas por la propiedad **flex-direction.**

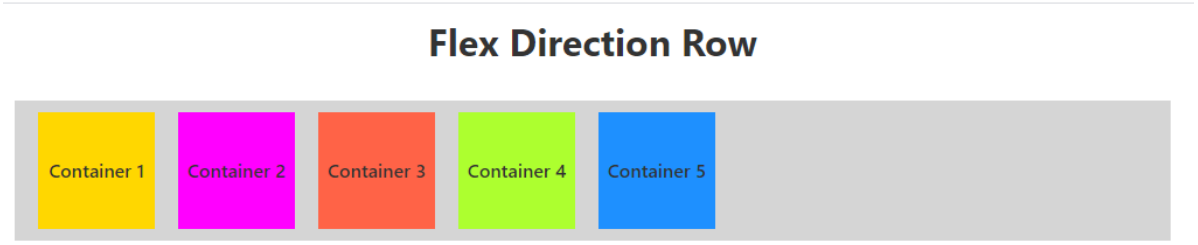


Propiedad Flex Direction

Cuando estamos trabajando con flexbox, debemos entender el concepto de ejes y pensar en eje principal y eje cruzado. La propiedad **flex-direction** funciona con respecto al main axis (eje principal) y el cross axis (eje cruzado) es perpendicular al main axis. **flex-direction** tiene los siguientes valores para su uso.

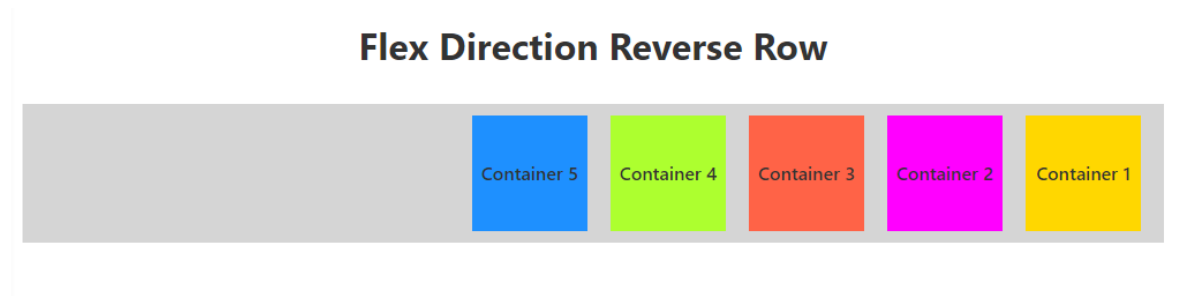
- row –permite que los elementos fluyan en direccion left to right dentro de la misma fila.
- row-reverse – los elementos fluyen, al contrario, es decir, right to left dentro de la misma fila
- column – permite que los elementos fluyan top to bottom en la columna.
- column-reverse – los elementos fluyen bottom to top dentro de la columna.

Flex-direction Row




```
.flex-container {
  background-color: rgb(213, 213, 213);
  display: flex;
  flex-direction: row;
  margin: 20px;
  padding: 10px;
}
```

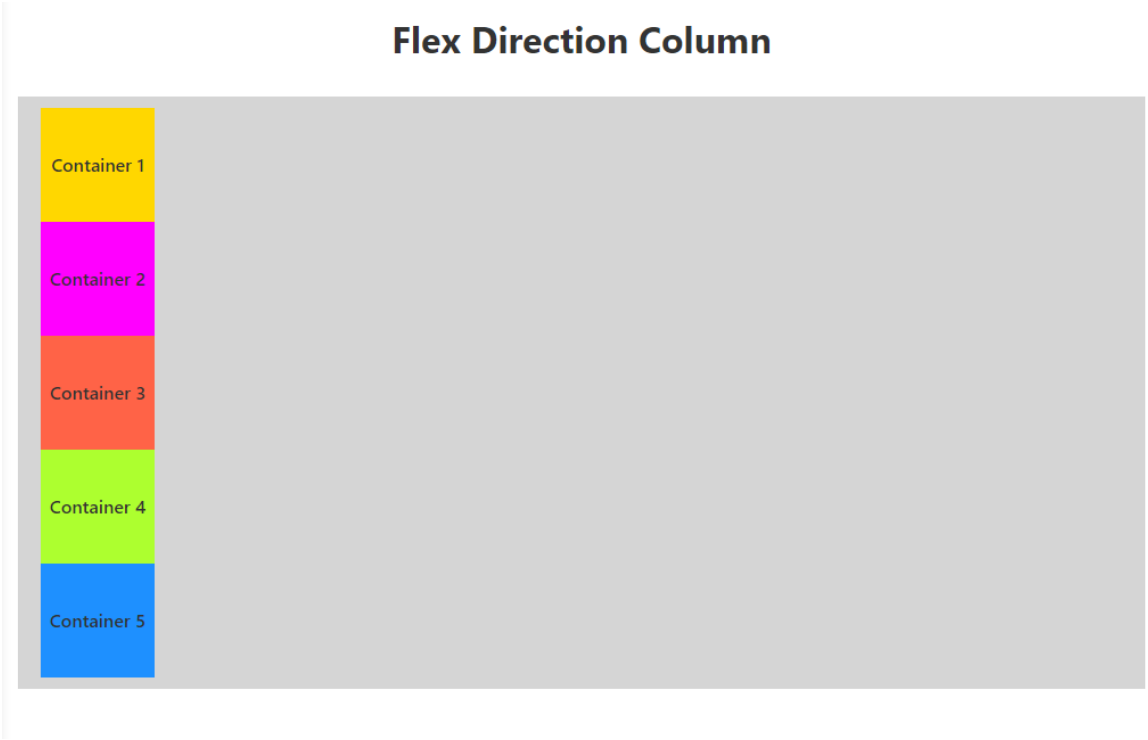
Flex-direction Reverse Row




```
.flex-container {
  background-color: rgb(213, 213, 213);
  display: flex;
  flex-direction: row-reverse;
  margin: 20px;
  padding: 10px;
}
```

Flex-direction Column

```
.flex-container {
  background-color: rgb(213, 213, 213);
  display: flex;
  flex-direction: column;
  margin: 20px;
  padding: 10px;
}
```



Flex-direction Column Reverse


```
.flex-container {
  background-color: rgb(213, 213, 213);
  display: flex;
  flex-direction: column-reverse;
  margin: 20px;
  padding: 10px;
}
```

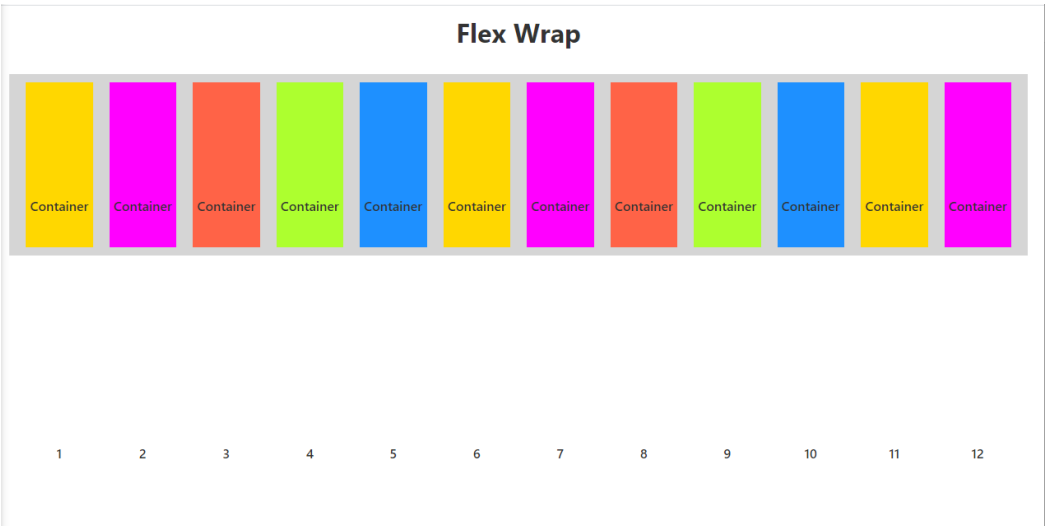
Propiedad Flex Wrap

En muchas ocasiones al agrupar elementos, nos encontramos con que salieron inesperadamente del contenedor, o incluso, los elementos que se encuentran dentro de cada flex item, no se visualizan correctamente.

La propiedad **flex-wrap** ayuda a resolver este tipo de problemas; esta obliga a que los elementos permanezcan en la misma línea o varias líneas del main axis.

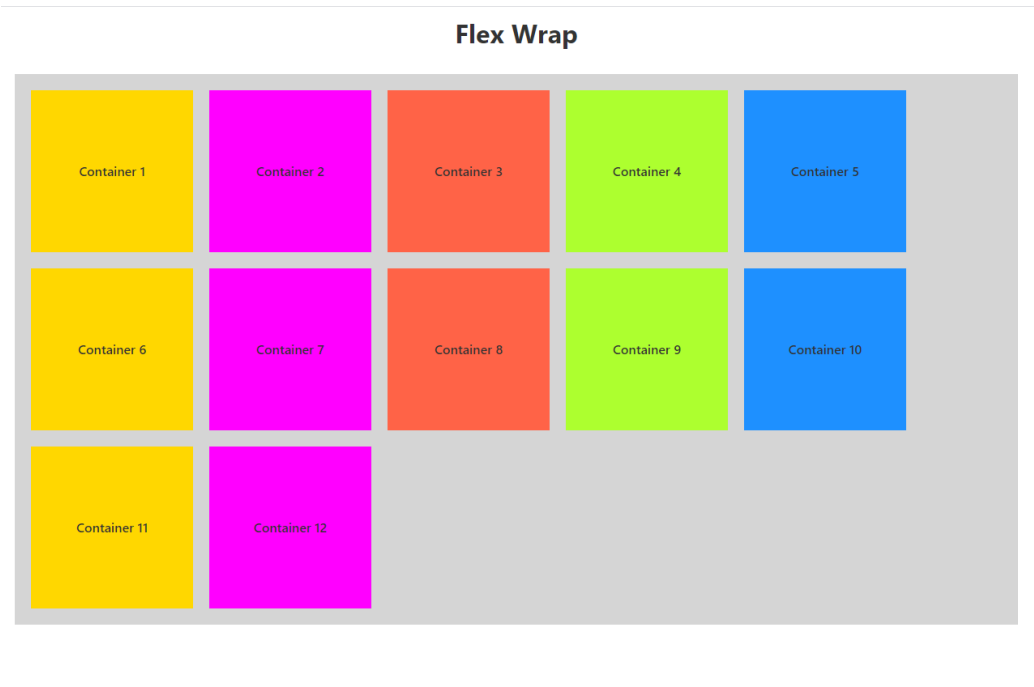
Como vemos en el siguiente ejemplo, el contenedor tiene 12 elementos, el texto no se visualiza correctamente debido a que por defecto el flex container tiene como valor **nowrap.**

Para solucionar este inconveniente, se debe usar la propiedad **flex-wrap** con el valor **wrap**.



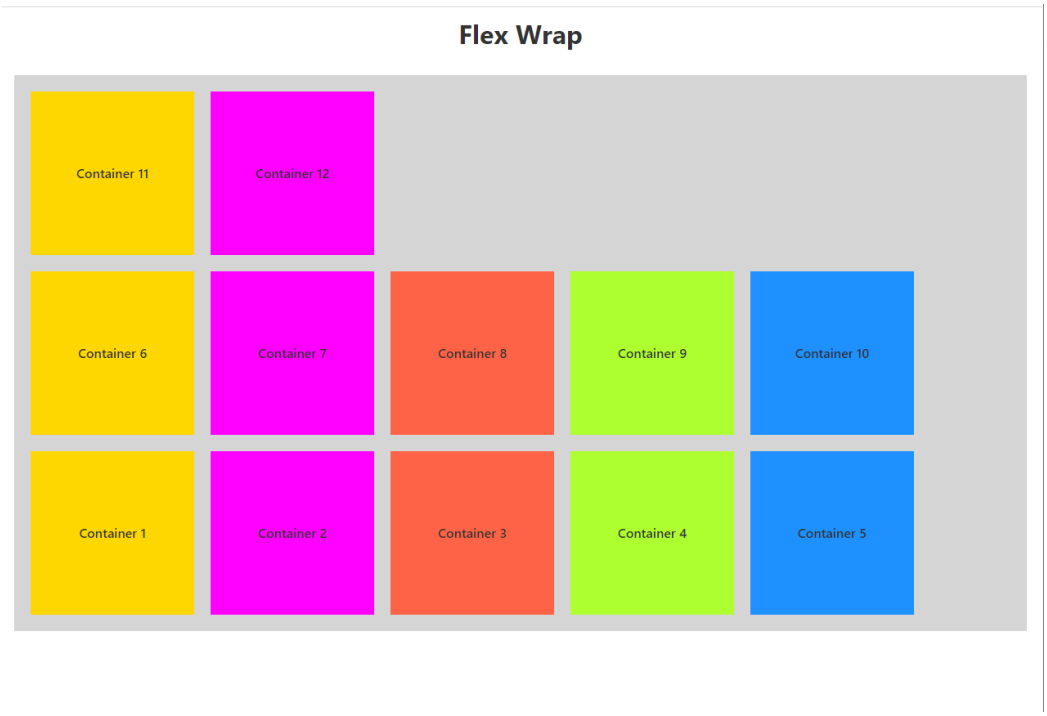
Con flex-wrap tendríamos el resultado de esta manera.

```
.flex-container {
  background-color: rgb(213, 213, 213);
  display: flex;
  flex-wrap: wrap;
  flex-direction: row;
  margin: 20px;
  padding: 10px;
}
```



También se puede hacer el uso de la opción wrap-reverse en caso tal de que se necesite realizar la agrupación inversa de todos los elementos.

```
.flex-container {
  background-color: rgb(213, 213, 213);
  display: flex;
  flex-wrap: wrap-reverse;
  flex-direction: row;
  margin: 20px;
  padding: 10px;
}
```

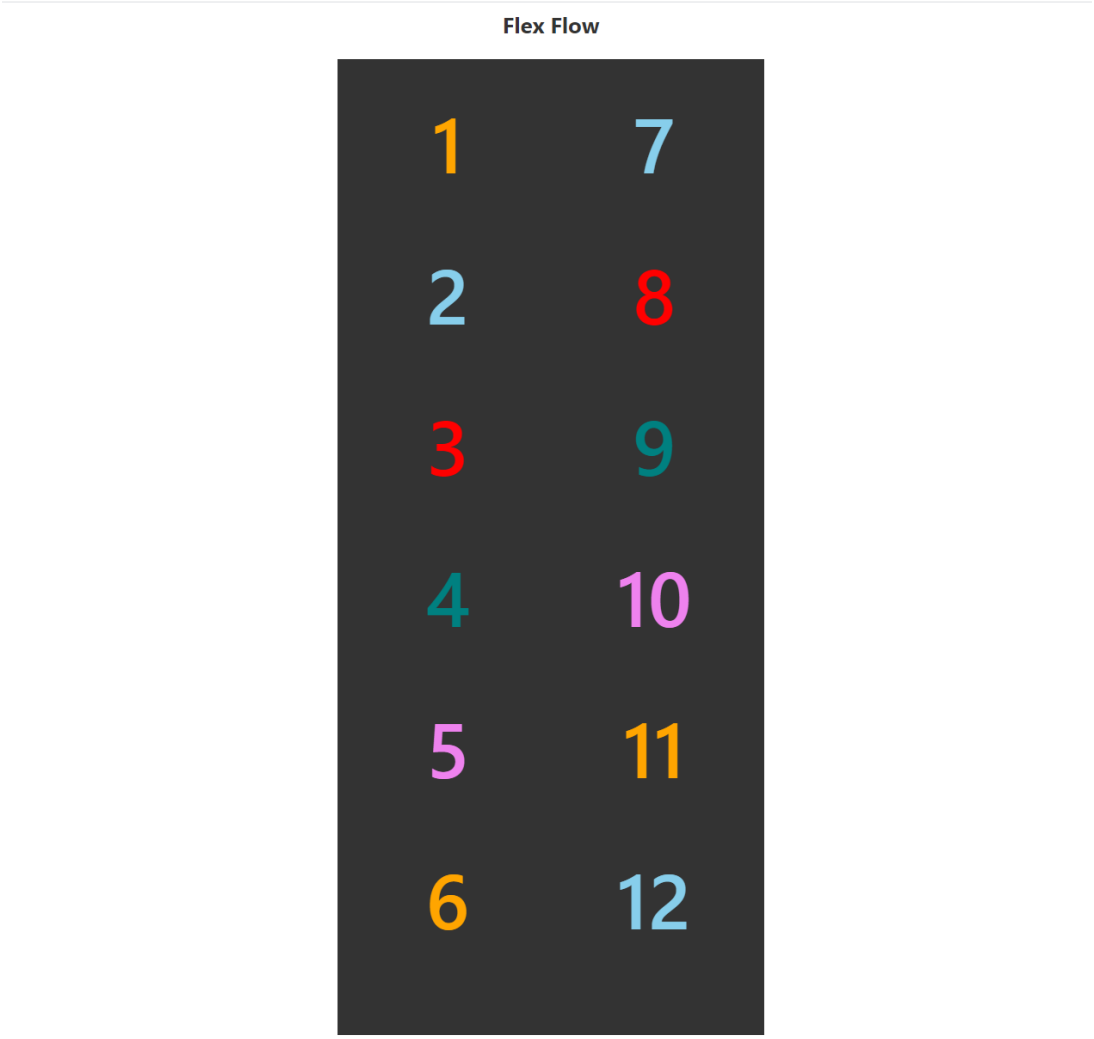


Propiedad Flex Flow

La propiedad **flex-flow** es una abreviación de las propiedades **flex-direction** y **flex-wrap**. Solo basta declarar la propiedad **flex-flow** y los argumentos **row, row-reverse, column, column-reverse, wrap, wrap-revserse**.

Si quisiéramos realizar una agrupación wrap en columnas, solo basta definir la propiedad **flex-flow** con los valores **column** y **wrap.**

```
.flex-container {
  width: 600px;
  height: 1400px;
  background-color: #333;
  display: flex;
  flex-flow: column wrap;
  margin: 20px auto;
  padding: 10px;
}
```



Propiedad Justify Content

Al usar esta propiedad, todos los ítems dentro del contendor son distribuidos dentro de su main axis, esto solo aplica al contenedor no a los ítems.

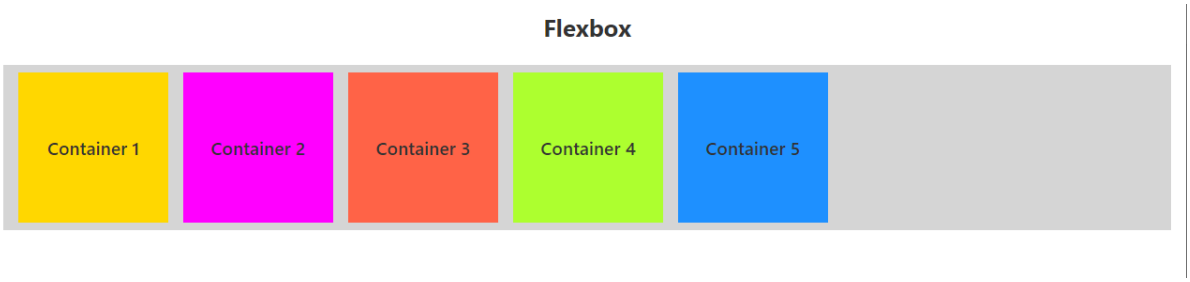
La propiedad **justifiy-content** permite alinear los elementos y define como se muestran en el espacio del contenedor o entre los ítems. Existen seis posibilidades de agrupación de elementos.

- flex-start
- flex-end
- center
- space-between
- space-evenly
- space-around

Flex Start

Permite agrupar los elementos desde el inicio del flex container con relación al main start.

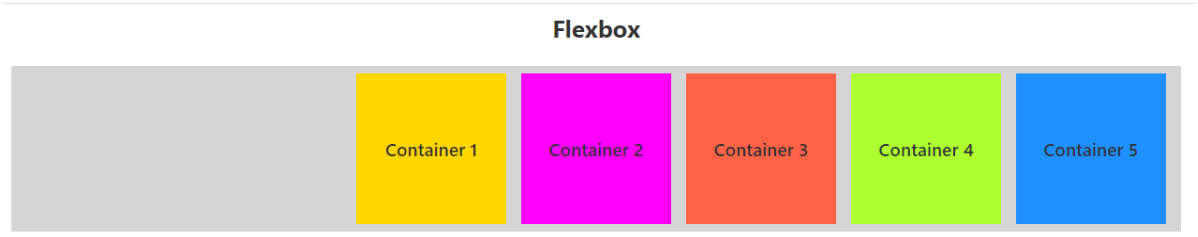
```
.flex-inline-container {
  background-color: rgb(213, 213, 213);
  display: flex;
  justify-content: flex-start;
  margin: 20px;
  padding: 10px;
}
```



Flex End

Permite agrupar los elementos al final del flex container con relación al main end.

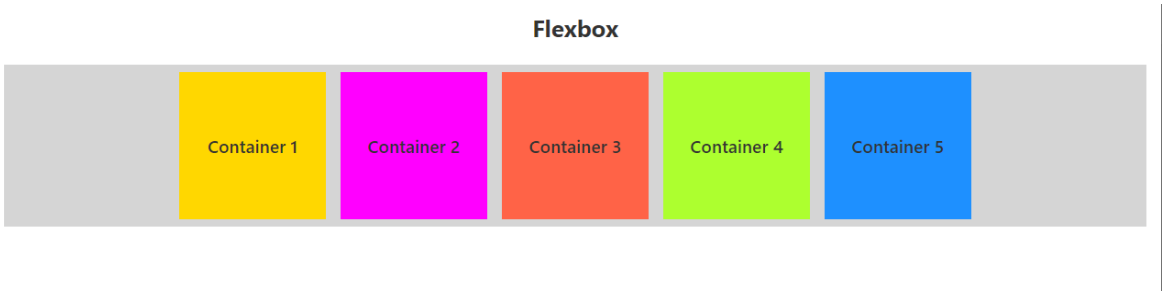
```
.flex-container {
  background-color: rgb(213, 213, 213);
  display: flex;
  margin: 20px;
  padding: 10px;
  justify-content: flex-end;
}
```



Center

Centra los elementos del flex container con relación al main axis.

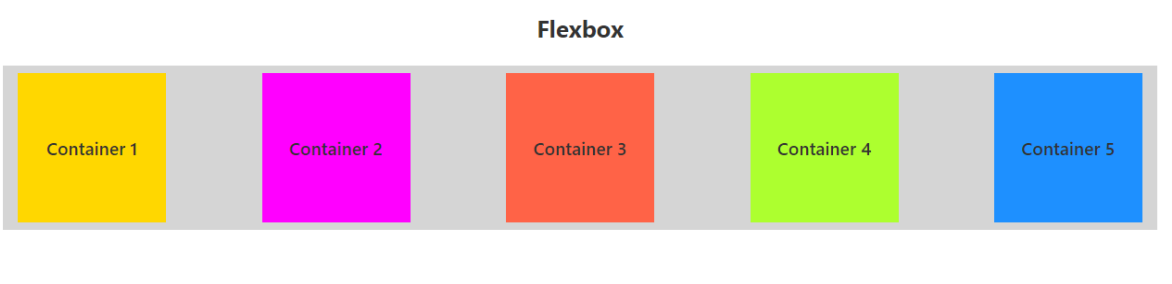
```css
.flex-container {
  background-color: rgb(213, 213, 213);
  display: flex;
  margin: 20px;
  padding: 10px;
  justify-content: center;
}
```



Space Between

Pone el primer item en el main start y el item final en el main end. Los ítems intermedios son pocisionados según el calculo restante del espacio entre el item inicial y el final.

```css
.flex-container {
  background-color: rgb(213, 213, 213);
  display: flex;
  margin: 20px;
  padding: 10px;
  justify-content: space-between;
}
```

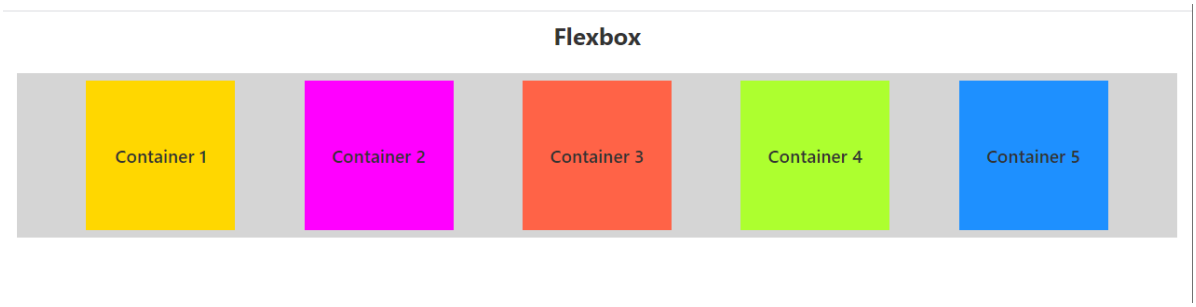


Space Evenly

Toma el espacio restante y lo divide para que se genere un espacio equitativo, esto significa que el espacio entre los ítems será de la misma longitud.

```css
.flex-container {
  background-color: rgb(213, 213, 213);
  display: flex;
  margin: 20px;
```

```
  padding: 10px;
  justify-content: space-evenly;
}
```



Space Around

Los espacios se distribuyen uniformemente entre los elementos adyacentes, sin embargo toma la mitad del espacio al inicio del primer item y al final del ultimo item.

```
.flex-container {
  background-color: rgb(213, 213, 213);
  display: flex;
  margin: 20px;
  padding: 10px;
  justify-content: space-around;
}
```