# UNIDAD TEMÁTICA 3
# FLEXBOX

## ÍNDICE DE CONTENIDOS



**Bibliografía :**

- https://css-tricks.com/snippets/css/a-guide-to-flexbox/
- https://lenguajecss.com/css/maquetacion-y-colocacion/flexbox/

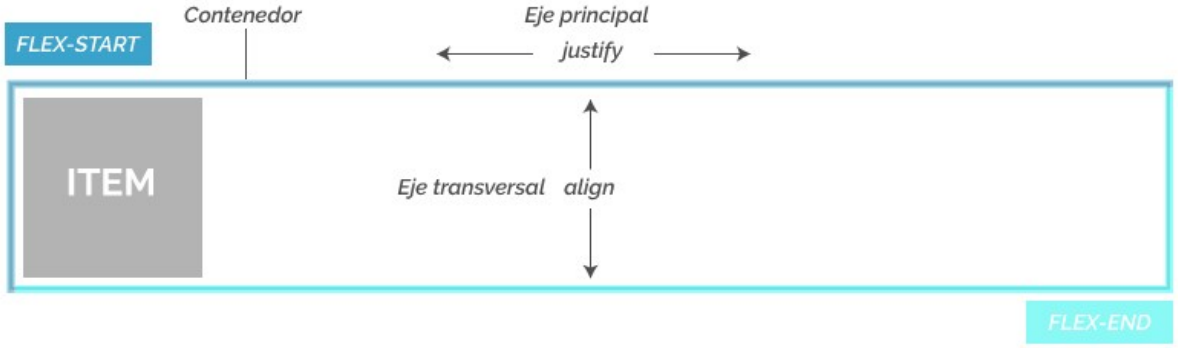
## 1.- Introducción

Flexbox es un sistema de disposición de elementos flexibles en donde los elementos se adaptan y colocan automáticamente y es más fácil personalizar los diseños y establecer los parámetros para la visualización de los mismos en responsive.

Veremos como aplicar la disposición flex y todas sus variantes para poder lograr un diseño donde puedas establecer el orden de los elementos de tu sitio web.

Lo primero que debes saber son los elementos a tomar en cuenta para aplicar este método:

- **Contenedor:** El elemento padre que es el contenedor que tendrá en su interior los ítems flexibles y adaptables.
- **Ítem:** Son los hijos flexibles que se encontraran dentro del contenedor.
- **Eje principal:** Los contenedores flexibles tendrán una orientación principal específica. Por defecto, es en horizontal (fila).
- **Eje transversal:** De la misma forma, los contenedores flexibles tendrán una orientación secundaria, perpendicular a la principal. Si la principal es en horizontal, la secundaria será en vertical, y viceversa.



Ejemplo :

```html
<div class="container">
    <div class="flex-child" style="background-color: #b2ebf2;">1</div>
    <div class="flex-child" style="background-color: #80deea;">2</div>
    <div class="flex-child" style="background-color: #4dd0e1;">3</div>
    <div class="flex-child" style="background-color: #26c6da;">4</div>
    <div class="flex-child" style="background-color: #00bcd4;">5</div>
    <div class="flex-child" style="background-color: #00acc1;">6</div>
    <div class="flex-child" style="background-color: #0097a7;">7</div>
    <div class="flex-child" style="background-color: #00838f;">8</div>
</div>
```

```css
.container{
    display: flex;
    max-width: 800px;
    margin: 0 auto;
    border: 2px solid #000000;
}

.flex-child {
    font-size: 2.5rem;
    text-align: center;
    line-height: 132px;
    color: #FFFFFF;
    width: 100%;
}
```

Resultado :


Si añadimos 2 elementos más :

## 2.- Propiedades Flexbox

### 2.1.- Flex-direction

Vamos a ver como cambiar la disposición de los elementos y el comportamiento de los ítems a lo largo del eje principal del contenedor utilizando la propiedad ***flex-direction***

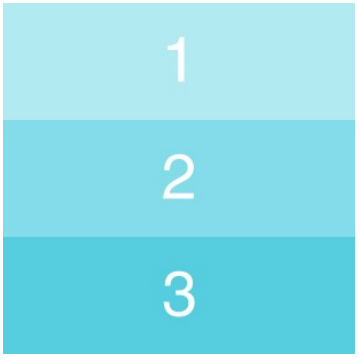
```css
.container{
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    justify-content: flex-start;
    max-width: 800px;
    margin: 0 auto;
    border: 2px dashed #505050;
}

.flex-child {
    padding: 0 2rem;
    font-size: 2.5rem;
    text-align: center;
    line-height: 132px;
    color: #FFFFFF;
}
```

Puede tomar los siguientes valores :

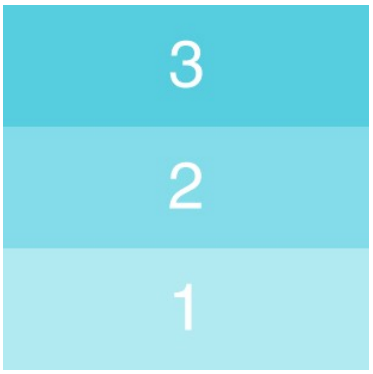
- **flex-direction: row** ( por defecto )


- **flex-direction: row-reverse**


- **flex-direction: column**




- **flex-direction: column-reverse**

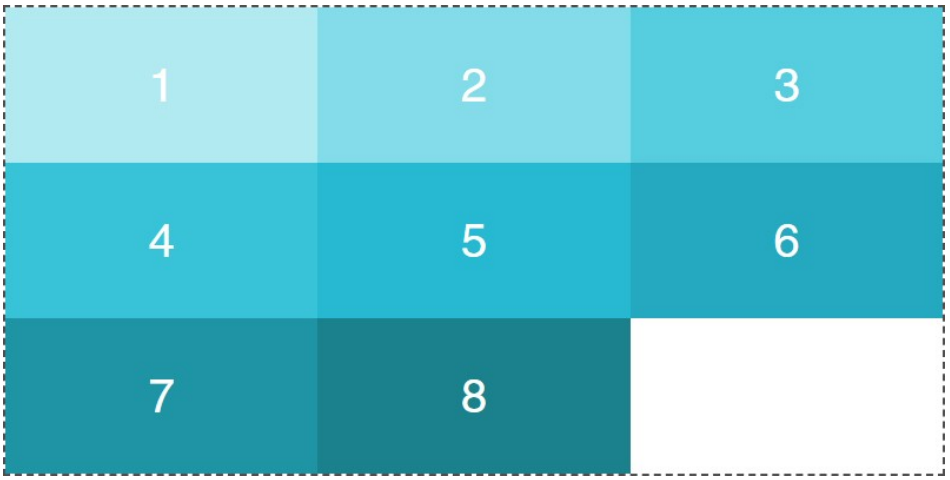



## 2.2.- Flex-wrap

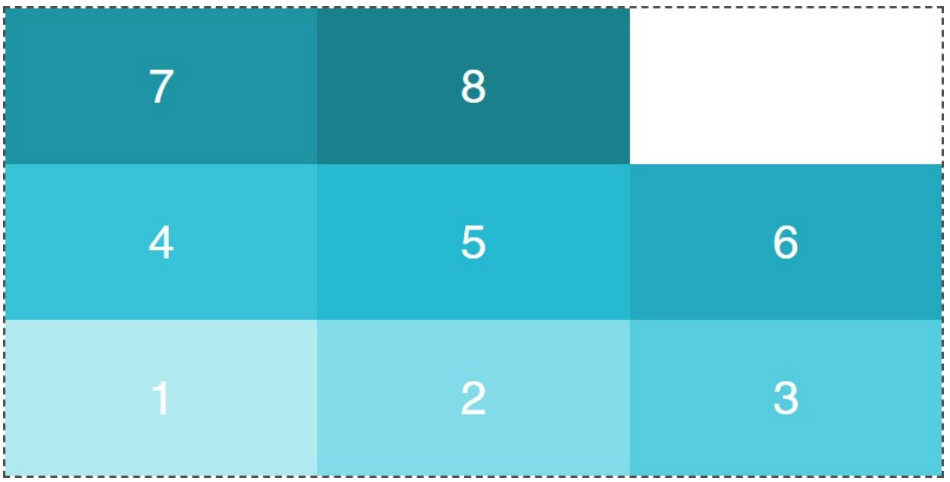
Si los elementos dentro del contenedor son más anchos que el contenedor estos se ajustan automáticamente (solo aplica si el ancho combinado de todos los ítems son mayores que la del contenedor) esto generaría un contenedor de ítems multilínea. Para esto utilizamos ***flex-wrap***

- **flex-wrap: wrap**

  Cuando el ancho de todos los items en total supera el ancho del contenedor, la propiedad flex-wrap: wrap ajustara las columnas dentro del contenedor.




- **flex-wrap: wrap-reverse,** esta propiedad invertirá el orden de cada línea de columnas que se formen.




## 2.3.- Justify-content

Con esta propiedad podremos colocar los items de un contenedor mediante una disposición concreta a lo largo del eje principal:


```
.container{
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    max-width: 800px;
    margin: 0 auto;
    border: 2px dashed #505050;
}

.flex-child {
    padding: 0 2rem;
    font-size: 2.5rem;
    text-align: center;
    line-height: 132px;
    color: #FFFFFF;
    width: 33.33%;
}
```

- **justify-content : flex-start** ( por defecto )


- **justify-content : flex-end**


- **justify-content : center**

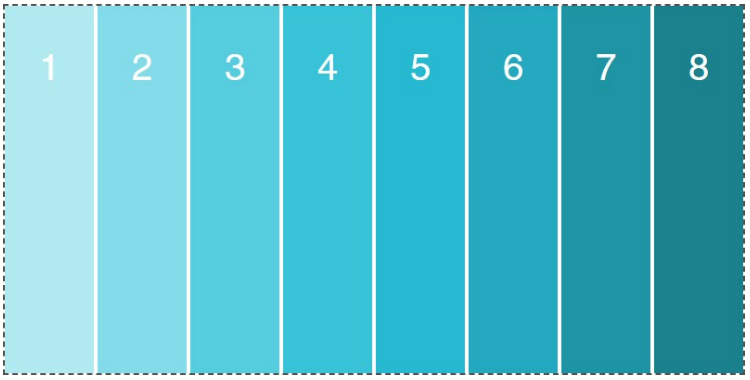

- **justify-content : space-around**


- **justify-content : space-between**

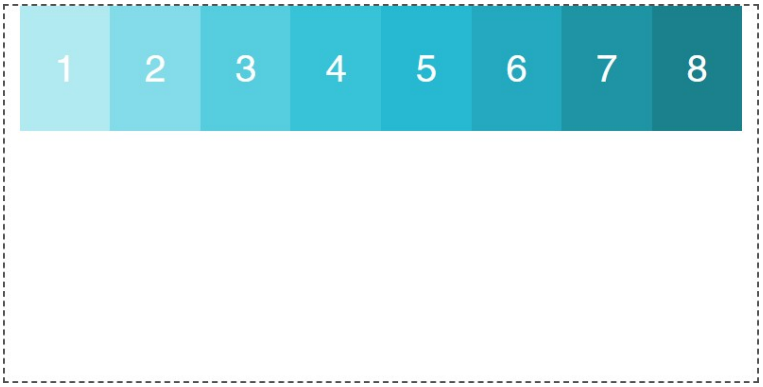

## 2.4.- Align-items

```css
.container{
   display: flex;
   flex-direction: row;
   flex-wrap: wrap;
   justify-content: space-between;
   align-items: stretch;
   max-width: 800px;
   height: 400px;
   margin: 0 auto;
   border: 2px dashed #505050;
}

.flex-child {
  padding: 0 2rem;
  font-size: 40px;
  text-align: center;
  line-height: 132px;
  color: #FFFFFF;
  width: 25%;
}
```

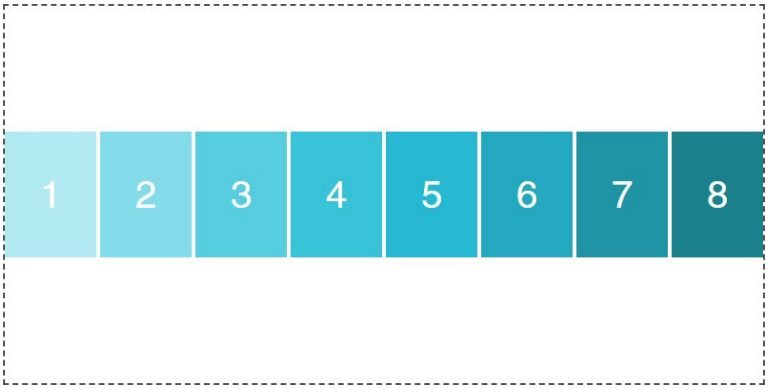
- **align-items: stretch,** con esta propiedad, los elementos ocupan todo el alto y ancho del contenedor.



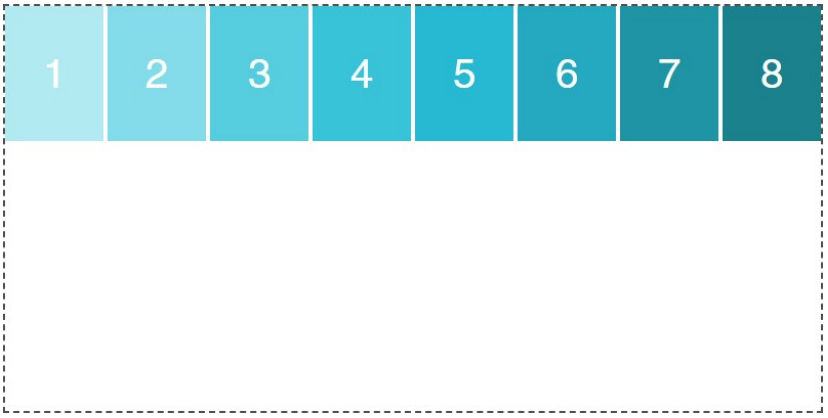

- **align-items: baseline,** alinea los elementos en el contenedor según la base del contenido de los ítems del contenedor.



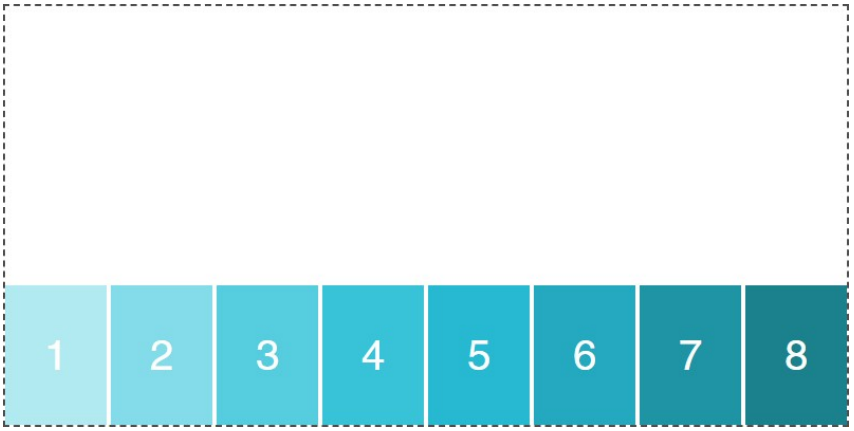

- **align-items: center**




- **align-items: flex-start**




- **align-items: flex-end**

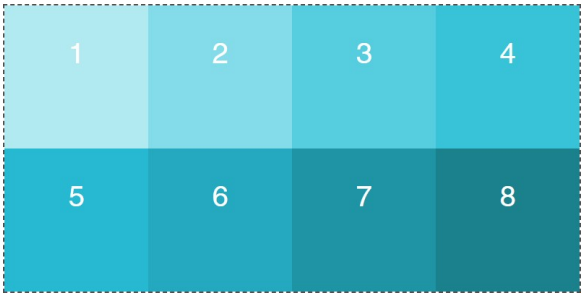



## 2.5.- Align-contents

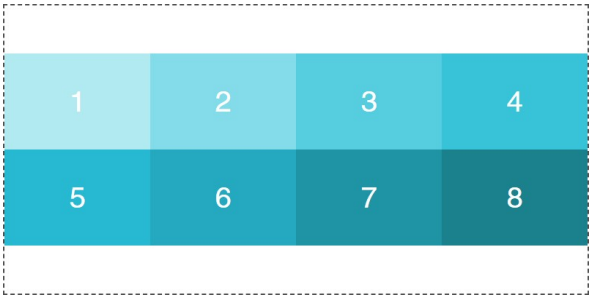
Tiene utilidad para contenedores flex multilínea ( los ítems no caben en el ancho disponible y el eje principal se divide en múltiples líneas. ( Solo es válido para este caso )

```
.container{
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    justify-content: space-between;
    align-items: center;
    align-content: stretch;
    max-width: 800px;
    height: 400px;
    margin: 0 auto;
    border: 2px dashed #505050;
}

.flex-child {
    padding: 0 2rem;
    font-size: 40px;
    text-align: center;
    line-height: 132px;
    color: #FFFFFF;
    width: 25%;
}
```

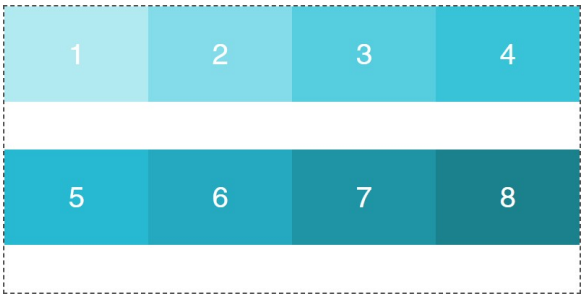
- **align-content: stretch**



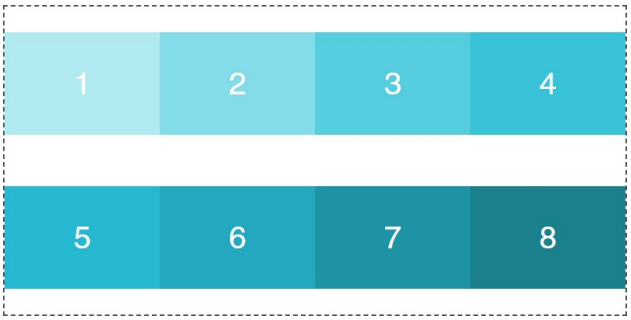

- **align-content: center**



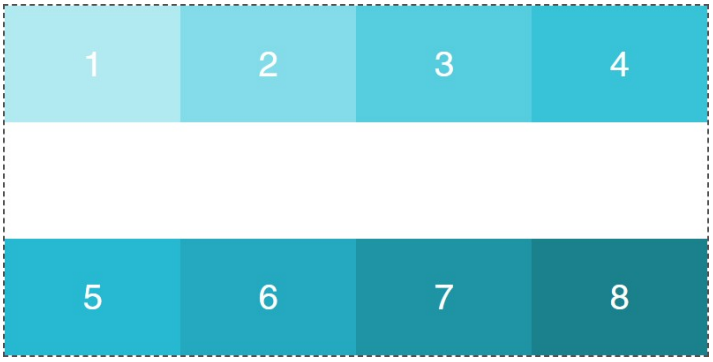

- **align-content: flex-start**




- **align-content: space-around**




- **align-content: space-between**




## 2.6.- Orden elementos

Una de las propiedades para los hijos del contenedor que utiliza flexbox es **order** que permite establecer el orden de cada uno de los ítems solamente desde el código css. De esta forma podemos recolocar fácilmente los ítems incluso utilizando media queries o responsive design.


```
<div class="container">
     <div class="flex-child" style="background-color: #b2ebf2;">1</div>
     <div class="flex-child" style="background-color: #80deea;">2</div>
     <div class="flex-child" style="background-color: #4dd0e1;">3</div>
     <div class="flex-child" style="background-color: #26c6da;">4</div>
</div>
```

```
.container > div:nth-child(1) {
    order: 3;
}

.container > div:nth-child(2) {
    order: 0;
}

.container > div:nth-child(3) {
    order: 1;
}

.container > div:nth-child(4) {
    order: 2;
}
```


## 2.7.- Flex

Flex es la propiedad abreviada para ***flex-grow,*** ***flex-shrink*** y ***flex-basis*** en ese orden específico.

- **flex-grow :** En flex-grow:x el número x indica cuantas unidades crecerá el item para calcular su tamaño final. Por ejemplo, flex-grow: 3 significa que el ítem crecerá 3 unidades.

  Para el ejemplo anterior, sabemos que el ítem crecerá 3 unidades. ¿Pero cuánto vale cada unidad? La forma para calcularlo es bastante sencilla. Simplemente hay que dividir el espacio disponible entre la suma de los flex-grow de todos los ítems (en la misma línea).

  Ver ejemplos

- **flex-shrink :** Si el espacio disponible es negativo (el tamaño del contenedor es menor a la suma de los tamaños de los ítems), de forma predeterminada los ítems se encogen en proporciones iguales para caber en una sola línea (en un próximo artículo trabajaremos flexbox con varias líneas de items). Con la propiedad flex-shrink se controla cómo se encogerán los elementos.

  Ver ejemplos

- **flex-basis:** La propiedad ***flex-basis***, como su nombre lo indica, define el main-size base para un flex-item. Tamaño base principal (main size base) significa que no necesariamente ese será su tamaño al dibujarse por el navegador, pero que será un punto de partida para calcular el tamaño final.

  En otras palabras, si el main axis es horizontal (predeterminado), flex-basis será equivalente a width; y si el main axis es vertical, flex-basis será equivalente a height.

  Ten en cuenta que el tamaño definido por flex-basis es, como su

nombre lo dice, el tamaño base. Es decir, que podrá variar (crecer o encogerse), según los valores de flex-grow y flex-shrink

Ver ejemplos

El valor predeterminado de flex (si no se especifica) es:

**flex: 0 1 auto;**

Es decir que por defecto los ítems no van a crecer pero si reducirse de manera proporcional para ajustarse al espacio. Además, que flex-basis se establece en auto (respecto a su contenido).

Importante :

- flex-basis siempre gana sobre el valor de width o height.
- Si no se define el valor de flex-basis o se establece en auto, se tomará en cuenta el valor de width o height según sea el caso.
- Si no se define un valor para flex-basis y tampoco se especifica el tamaño por width o height, se definirá el main-size según su contenido.