Tema 4: Flexbox

- Comprender los conceptos básicos de Flexbox y su utilidad en el diseño web.
- Familiarizarse con las propiedades clave de Flexbox y cómo se aplican.
- Ser capaz de crear diseños flexibles y responsivos utilizando Flexbox.

1. ¿Qué es Flexbox y por qué es importante en el diseño web?

Flexbox (Flexible Box Layout) es un módulo de diseño en CSS que proporciona un método flexible para organizar y distribuir elementos. Es habitual utilizarlo para la creación de barras de navegación, galerías de imágenes, entre otras. Cabe destacar que facilita la alineación vertical y horizontal de elementos.

Ventajas de usar Flexbox en comparación con otros métodos de diseño.

- **1. Simplifica la Maquetación:** elimina la necesidad de flotar elementos o usar técnicas complicadas de posicionamiento para lograr diseños complejos.
- **2. Control Preciso del Espaciado:** permite un control preciso del espacio entre elementos, tanto a nivel horizontal como vertical, lo que facilita la creación de diseños limpios y estéticamente agradables.
- **3. Alineación Sencilla:** facilita la alineación de elementos en el eje principal y transversal, lo que soluciona uno de los problemas más comunes en el diseño web.
- **4. Adaptable y Responsivo:** está diseñado para crear diseños que se adapten fácilmente a diferentes tamaños de pantalla y dispositivos, lo que mejora la experiencia del usuario.
- **5. Ordenación de Elementos:** permite cambiar el orden visual de los elementos sin cambiar el orden en el HTML, lo que es útil para presentar la información de manera más efectiva.

Usa Flexbox cuando:

- **Diseño unidimensional:** Flexbox es especialmente efectivo para diseños en una dimensión, es decir, para organizar elementos en filas o columnas. Es excelente para barras de navegación, galerías de imágenes y diseños de tipo lista.

- **Alineación y distribución flexible:** Si necesitas alinear o distribuir elementos de manera flexible dentro de un contenedor.
- **Flujo de contenido dinámico:** Si el contenido de una página puede cambiar dinámicamente (por ejemplo, una lista de elementos de longitud variable), Flexbox permite adaptarnos fácilmente a esos cambios.
- **Control de elementos individuales:** Flexbox proporciona un control detallado sobre los elementos individuales dentro de un contenedor. Puedes cambiar el orden de los elementos, ajustar su tamaño y alineación de manera específica.

Usa Grid Layout cuando:

- **Diseño bidimensional:** Grid Layout es ideal para diseños bidimensionales, donde se necesita organizar elementos en filas y columnas al mismo tiempo. Es perfecto para diseños más complejos y estructurados.
- **Diseños de tipo cuadrícula o mosaico:** Si estás creando una disposición que se asemeja a una cuadrícula, donde los elementos se alinean en filas y columnas definidas, Grid Layout es la mejor opción.
- **Distribución de contenido compleja:** Si necesitas dividir el contenido de tu página en áreas distintas, Grid Layout te permite crear áreas explícitas y controlar cómo se distribuye el contenido en cada una de ellas.
- **Control de diseño a nivel de contenedor:** Grid Layout es excelente para controlar la estructura completa de tu página, definiendo áreas y subgrids dentro del diseño.

Consideraciones generales:

- **Combinación de ambas:** A menudo, la mejor estrategia es combinar Flexbox y Grid Layout en la misma página. Flexbox puede ser usado a nivel de elementos individuales dentro de un contenedor Grid.
- **Práctica y experiencia:** La práctica y la experiencia en el uso de ambas técnicas te ayudarán a determinar cuál es la mejor elección para cada situación específica.

2. Contenedor Flex y Elementos Flex

Un **contenedor** Flex es un elemento HTML que tiene sus hijos directos configurados para usar Flexbox. Se convierte en un contexto de diseño flexible. Con **display** se define un elemento como contenedor flex. En este ejemplo, el elemento con la clase .contenedor-flex se convierte en un contenedor flex.

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

Los **elementos** hijos directos de un contenedor Flex se convierten en elementos flexibles que participan en el diseño flexible.

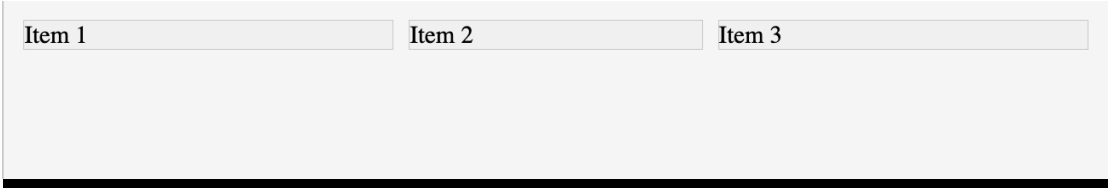
```
.elemento-flex {
    flex: 1; /* Propiedad para especificar el comportamiento flexible del  elemento */
 }
```

flex: Esta es una propiedad abreviada que combina tres propiedades relacionadas con la flexibilidad del elemento: **flex-grow, flex-shrink, y flex-basis.**

```
.item {
   flex: <flex-grow> <flex-shrink> <flex-basis>;
}
```

- **flex-grow:** Indica cuánto espacio adicional debe tomar el elemento en relación con los otros elementos flexibles dentro del mismo contenedor. Un valor de 1 significa que tomará todo el espacio adicional disponible.
- **flex-shrink:** Define la capacidad del elemento para reducir su tamaño si es necesario para evitar el desbordamiento del contenedor.
- **flex-basis:** Establece el tamaño inicial del elemento antes de que se distribuya el espacio adicional o se apliquen los factores de reducción.

Ejemplo: En este ejemplo, hay tres elementos en un contenedor flex. Todos tienen flex-grow: 1; para que se expandan y ocupen el espacio disponible. Además, pueden encogerse si el espacio es limitado debido a flex-shrink: 1;. El flex-basis establece el tamaño inicial antes de que se apliquen las reglas de crecimiento o encogimiento.




```
.container {
    display: flex;
}
.item {
    flex-grow: 1; /* Ocupa todo el espacio disponible */
    flex-shrink: 1; /* Puede encogerse si es necesario */
    flex-basis: 200px; /* Tamaño inicial */
    background-color: #f0f0f0;
    border: 1px solid #ccc;
    margin: 5px;
}
.item:nth-child(2) {
    flex-basis: 150px; /* Este elemento tendrá un tamaño inicial mayor */
}
<div class="container">
    <div class="item">Item 1</div>
    <div class="item">Item 2</div>
    <div class="item">Item 3</div>
</div>
```

- .**container** es el contenedor flex que alberga los elementos.
- .**item** son los elementos flex dentro del contenedor.
- **flex-grow: 1;** permite que los elementos crezcan y ocupen todo el espacio disponible.
- **flex-shrink: 1;** indica que los elementos pueden encogerse si es necesario.
- **flex-basis: 200px;** establece el tamaño inicial de los elementos a 100 píxeles.
- **flex-basis: 150px;** se aplica al segundo elemento, lo que significa que su tamaño inicial es de 150 píxeles.

3. Ejes Principal y Transversal

3.1. Eje Principal:

El eje principal es la dirección en la que se colocan los elementos flexibles dentro del contenedor. Puede ser horizontal (fila) o vertical (columna).

```
.contenedor-flex {
    flex-direction: row; /* Eje principal horizontal (por defecto) */ /* o */
    flex-direction: column; /* Eje principal vertical */
}
```

3.2. Eje Transversal:

Es el eje perpendicular al eje principal. Si el eje principal es horizontal, el transversal es vertical y viceversa.

```
.contenedor-flex {
    align-items: center; /* Alinea elementos verticalmente en el eje transversal */ /* o */
    justify-content: center; /* Alinea elementos horizontalmente en el eje transversal */
}
```

3.3. Dirección de Escritura y Dirección del Eje Principal

*** Dirección de Escritura:**

Es la dirección en la que se escribe el texto. Puede ser de izquierda a derecha (predeterminado) o de derecha a izquierda.

```
.contenedor-flex {
    direction: ltr; /* De izquierda a derecha (predeterminado) */ /* o */
    direction: rtl; /* De derecha a izquierda */
}
```

*** Dirección del Eje Principal:**

En una fila, el eje principal sigue la dirección de escritura. En una columna, el eje principal es vertical.

```
.contenedor-flex {
    flex-direction: row; /* Eje principal en dirección de escritura (horizontal) */ /* o */
    flex-direction: row-reverse; /* Eje principal en dirección opuesta a la escritura */ /* o */
    flex-direction: column; /* Eje principal vertical */
}
```

3.4. Propiedades

3.4.1. flex-direction

- **Flex-direction**. flex-direction establece la dirección en la que se colocan los elementos dentro del contenedor flex.

```
.contenedor-flex {
        flex-direction: row; /* Opciones: row, row-reverse, column, column-reverse */
  }
```

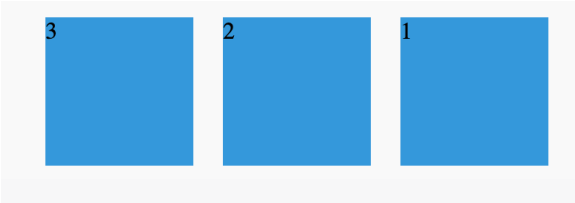
En los ejemplos se usa el siguiente código html:

```
<!DOCTYPE html>
<html lang="es">
<head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width,
initial-scale=1.0"> <title>Ejemplo Flexbox - Row</title>
<link rel="stylesheet" href="styles.css"> </head>
<body>
    <div class="contenedor-flex row">
    <div class=“elemento">1</div>
    <div class="elemento">2</div>
    <div class="elemento">3</div> </div>
 </body>
</html>
```

Ejemplo 1: flex-direction: row; (Fila Horizontal)

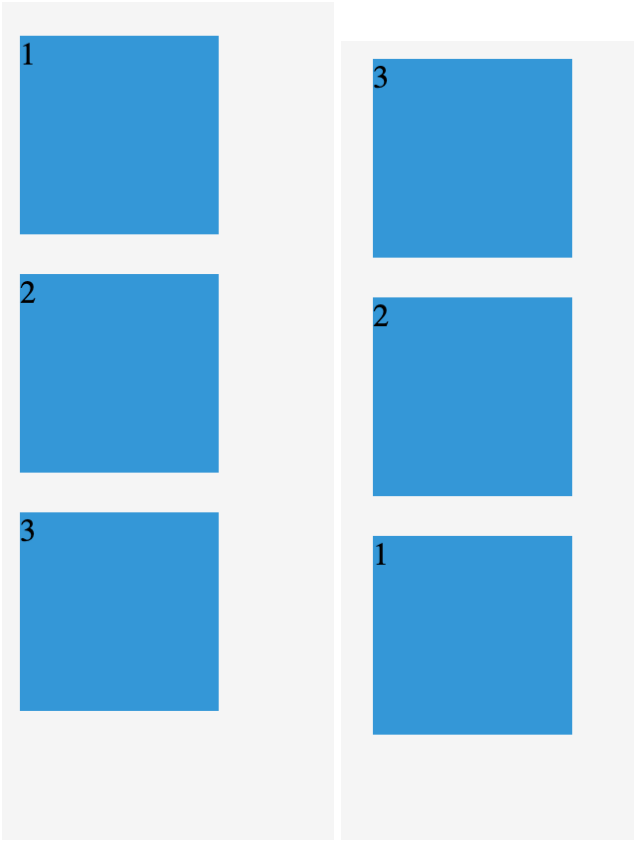

```
.contenedor-flex {
        display: flex;
}
.row {
        flex-direction: row;
}
 .elemento {
        width: 100px;
        height: 100px;
        background-color: #3498db;
        margin: 10px;
}
```

Ejemplo 2: flex-direction: row-reverse; (Fila Horizontal Invertida)




```
.contenedor-flex {
        display: flex;
}
 .row-reverse {
        flex-direction: row-reverse;
 }
 .elemento {
        width: 100px;
         height: 100px;
        background-color: #3498db;
        margin: 10px;
}
```

Ejemplo 3: flex-direction: column; (Columna Vertical) y Ejemplo 4: flex-direction: column-reverse; (Columna Vertical Invertida)




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


 }
 .elemento {
        width: 100px;
        height: 100px;
        background-color: #3498db;
         margin: 10px;
}
```

EJERCICIO:

Crea una página con cuatro secciones, cada una representada por una caja rectangular. Cada caja debe contener un título (usando etiquetas <h2>) y un párrafo de contenido (usando etiquetas <p>).

Cada sección tendrá un color diferente.

Las secciones deben estar organizadas utilizando Flexbox con las siguientes configuraciones de flex-direction.

- La primera sección debe tener una disposición en filas (horizontal).
- La segunda sección debe tener una disposición en columnas (vertical).
- La tercera sección debe tener una disposición en filas inversa.
- La cuarta sección debe tener una disposición en columnas inversa.

3.4.2. flex-wrap

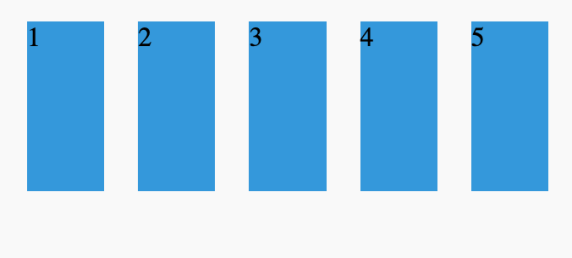
- **Flex-wrap.** Esta propiedad permite que los elementos se distribuyan en varias líneas si el espacio es insuficiente.

```
.contenedor-flex {
    flex-wrap: wrap; /* Opciones: nowrap, wrap, wrap-reverse */
}
```

En los ejemplos siguientes se utilizará el código base:

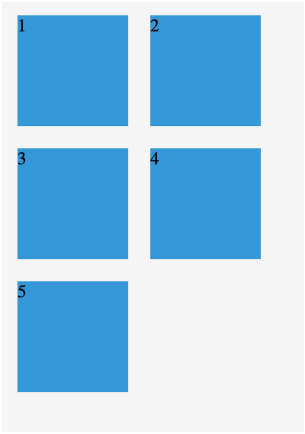
```
<!DOCTYPE html> <html lang="es"> <head>
<meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Ejemplo Flexbox - No Wrap</title>
<link rel="stylesheet" href="styles.css">
</head>
 <body>
    <div class="contenedor-flex no-wrap">
        <div class="elemento">1</div>
        <div class="elemento">2</div>
        <div class="elemento">3</div>
        <div class="elemento">4</div>
        <div class="elemento">5</div>
    </div>
</body>
 </html>
```

Ejemplo 1: flex-wrap: nowrap; (Sin Envoltura)



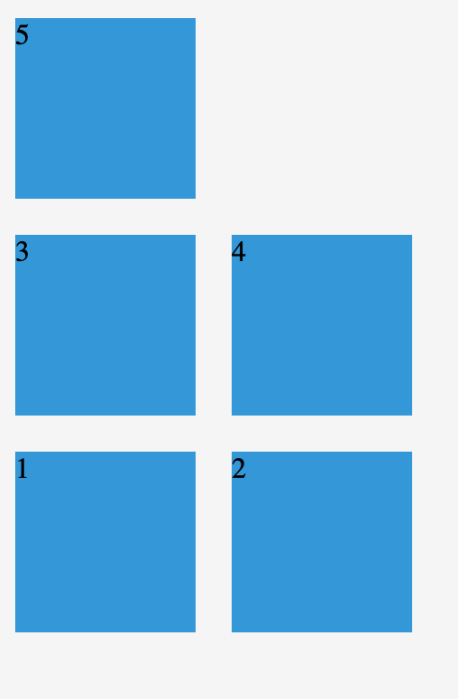
```
.contenedor-flex {
        display: flex;
 }
.no-wrap {
        flex-wrap: nowrap;
}
 .elemento {
        width: 100px;
        height: 100px;
        background-color: #3498db;
        margin: 10px;
}
```

Ejemplo 2: flex-wrap: wrap; (Envoltura)




```
.contenedor-flex {
        display: flex;
}
.wrap {
        flex-wrap: wrap;
}
.elemento {
        width: 100px;
        height: 100px;
        background-color: #3498db;
        margin: 10px;
}
```

Ejemplo 3: flex-wrap: wrap-reverse; (Envoltura Invertida)




```
.contenedor-flex {
        display: flex;
}
 .wrap-reverse {
         flex-wrap: wrap-reverse;
}
 .elemento {
        width: 100px;
        height: 100px;
        background-color: #3498db;
        margin: 10px;
}
```

3.4.3. flex-flow

- **Flex-flow.** es una propiedad abreviada que combina flex-direction y flex-wrap. Este ejemplo establece una fila con envoltura.

```
.contenedor-flex {
    flex-flow: row wrap; /* Dirección y envoltura */
}
```

Ejemplo 1: flex-flow: row nowrap; (Fila Horizontal sin Envoltura)-Usando la misma base de html que lo anterior

```
.contenedor-flex {
    display: flex;
}
.row-no-wrap {
    flex-flow: row nowrap;
}
.elemento {
    width: 100px;
    height: 100px;
    background-color: #3498db;
    margin: 10px;
}
```

Ejemplo 2: flex-flow: row wrap; (Fila Horizontal con Envoltura)

```
.contenedor-flex {
    display: flex;
}
.row-wrap {
    flex-flow: row wrap;
}
.elemento {
    width: 100px;
    height: 100px;
    background-color: #3498db;
    margin: 10px;
}
```

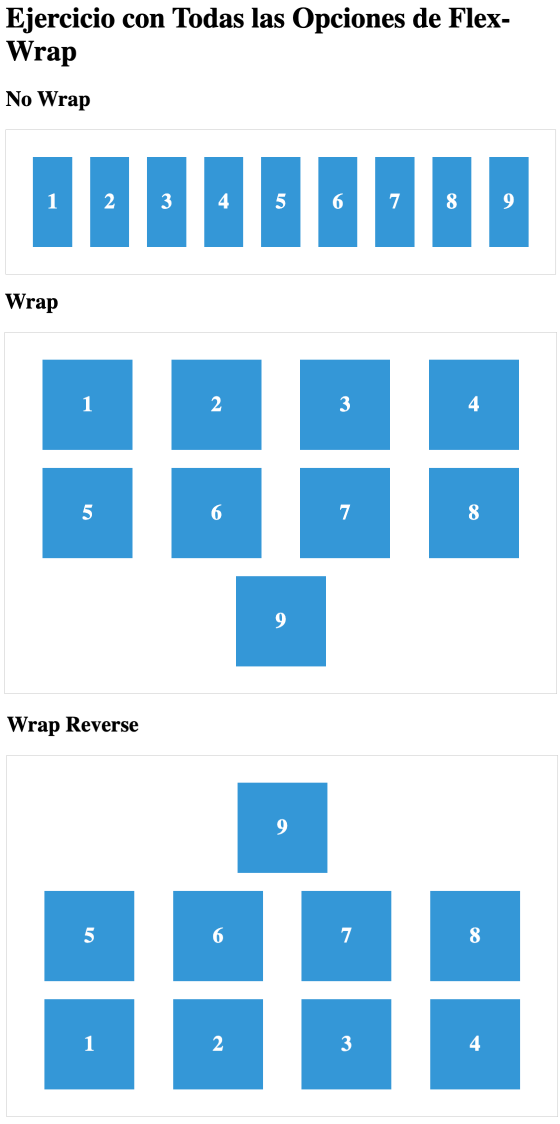
Ejemplo 3: flex-flow: row-reverse nowrap; (Fila Horizontal Invertida sin Envoltura)

```
.contenedor-flex {
        display: flex;
 }
.row-reverse-no-wrap {
        flex-flow: row-reverse nowrap;
}
 .elemento {
        width: 100px;
        height: 100px;
        background-color: #3498db;
        margin: 10px;
}
```

EJERCICIO:

Crea una página que muestre tres contenedores flexibles, cada uno con una configuración diferente de flex-wrap. Cada contenedor debe contener nueve elementos en contenedores cuadrados numerados del 1 al 9.

- El primer contenedor debe tener la configuración flex-wrap: nowrap;, lo que significa que los elementos no se envolverán y seguirán en la misma línea incluso si no hay suficiente espacio horizontal.
- El segundo contenedor debe tener la configuración flex-wrap: wrap;, lo que permitirá que los elementos se envuelvan a la siguiente línea si no hay suficiente espacio horizontal.
- El tercer contenedor debe tener la configuración flex-wrap: wrap-reverse;, haciendo que los elementos se envuelvan en orden inverso.



3.4.5. Alineación

- **Justify-content**. alinea los elementos a lo largo del eje horizontal del contenedor.

```
.contenedor-flex {
    justify-content: center; /* Opciones: flex-start, flex-end, center, space-between, space-around */
}
```

  - **flex-start:** Los elementos se alinean hacia el inicio del contenedor. El primer elemento estará en la esquina inicial y los demás se seguirán en la dirección del eje principal.
  - **flex-end:** Los elementos se alinean hacia el final del contenedor. El último elemento estará en la esquina final y los demás se colocarán antes en la dirección del eje principal.
  - **center:** Los elementos se centran dentro del contenedor a lo largo del eje principal.
  - **space-between:** Los elementos se distribuyen uniformemente a lo largo del eje principal. El primer elemento se coloca al inicio, el último al final, y el espacio restante se distribuye igualmente entre los elementos.
  - **space-around:** Similar a space-between, pero con espacio adicional antes del primer elemento y después del último elemento. Es decir, deja entre todo todos los elementos el mismo espacio, incluso entre el primero y el último desde los extremos opuestos.

- **Align-items.** alinea los elementos a lo largo del eje transversal del contenedor (eje perpendicular al principal). Los elementos se centran verticalmente dentro del contenedor.

```
.contenedor-flex {
    align-items: center; /* Opciones: flex-start, flex-end, center, baseline, stretch */
}
```

  - **flex-start:** Los elementos se alinean al principio del contenedor.
  - **flex-end:** Los elementos se alinean al final del contenedor.
  - **center:** Los elementos se centran verticalmente dentro del contenedor.

- **baseline:** Los elementos se alinean en su línea de base (es decir, la línea imaginaria en la que se apoyan las letras).
- **stretch:** Los elementos se estiran para llenar todo el espacio disponible en el eje transversal.

EJERCICIO. Con justify-center y aling-items

- **Utilizar justify-content: center; a nivel de body y html. Esto centra todo el contenido horizontalmente.**
- **align-items: center; también se aplica a body y html, centrando todo el contenido verticalmente.**
- **En el contenedor .container, se crean tres elementos .box que son cajas de 100x100**
- **Dentro de cada .box, se utiliza justify-content: center; y align-items: center; que centra el contenido (en este caso, el texto "Caja 1", "Caja 2" y "Caja 3") tanto horizontal como verticalmente.**

```
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link rel="stylesheet" href="styles.css">
  </head>
  <body>
    <div class="container">
      <div class="box">Caja 1</div>
      <div class="box">Caja 2</div>
      <div class="box">Caja 3</div>
    </div>
  </body></html>
body, html {
  height: 100%;
  margin: 0;
  display: flex;
  justify-content: center;
  align-items: center;
}
```

```
.container {
   display: flex;
}
.box {
   width: 100px;
   height: 100px;
   background-color: #f0f0f0;
   border: 1px solid #ccc;
   margin: 5px;
   display: flex;
   justify-content: flex-start;
   align-items: ;
}
```



```
.container {
   display: flex;
}
.box {
   width: 100px;
   height: 100px;
   background-color: #f0f0f0;
   border: 1px solid #ccc;
   margin: 5px;
   display: flex;
   justify-content: flex-start;
   align-items:center ;
}
```

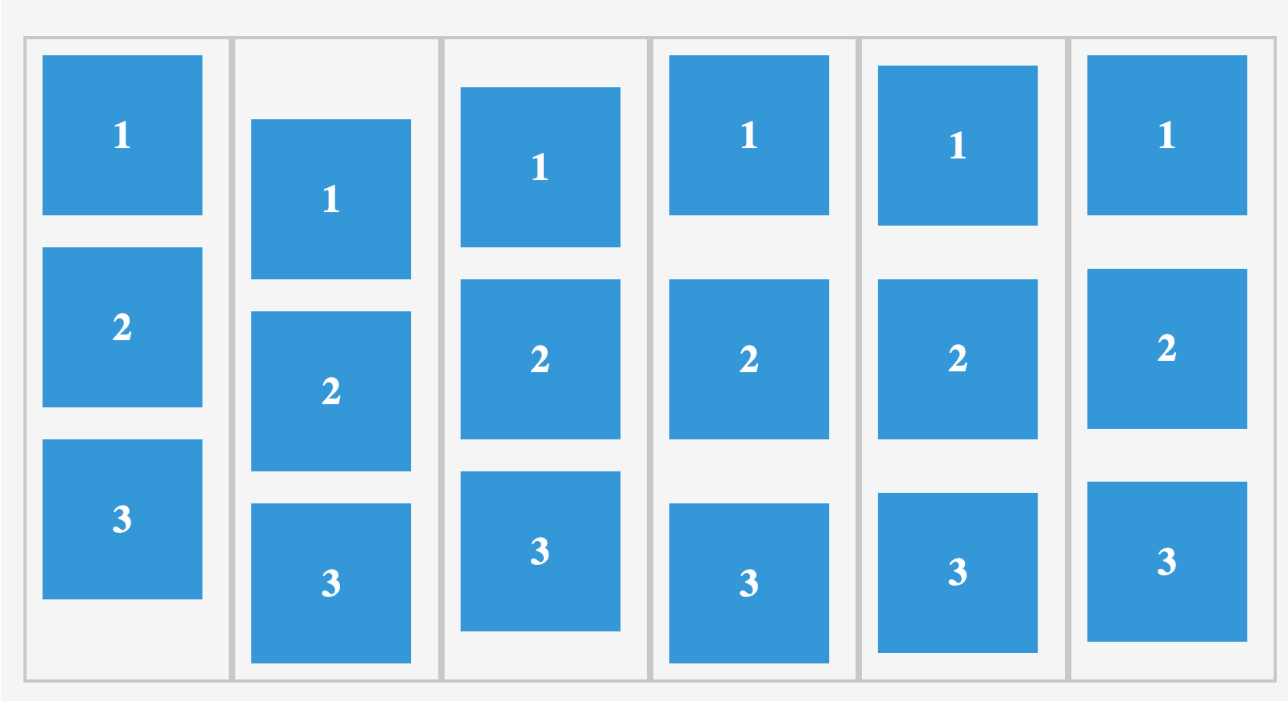
- **Align-content. C**ontrola el espacio extra en el eje transversal si hay múltiples líneas de elementos. Esta propiedad se aplica cuando hay envoltura y afecta al espacio entre las filas.

.contenedor-flex {

align-content: space-around; /* Opciones: flex-start, flex-end, center, space-between, space-around, stretch */

}

- **flex-start:** Las filas se alinearán al principio del contenedor.
- **flex-end:** Las filas se alinearán al final del contenedor.
- **center**: Las filas se centrarán verticalmente dentro del contenedor.
- **space-between:** Las filas se distribuirán uniformemente a lo largo del eje transversal.
- **space-around:** Las filas se distribuirán uniformemente a lo largo del eje transversal, con espacio adicional alrededor de cada fila.
- **stretch**: Las filas se estirarán para llenar todo el espacio disponible en el eje transversal.



4. Order

La propiedad order en Flexbox es utilizada para cambiar el orden de visualización de los elementos dentro de un contenedor flexible. Por defecto, todos los elementos tienen un valor de 0, lo que significa que se mostrarán en el orden en el que aparecen en el HTML. Al ajustar el valor de order, puedes alterar este orden visual sin modificar la estructura del DOM.

```
elemento {
    order: <número>;
}
```

- order no afecta la estructura del DOM, solo el orden visual en la presentación.
- Puedes usar order con valores decimales.
- order solo afecta los elementos directos dentro del contenedor flexible.
- Elementos con el mismo valor de order se mostrarán en el orden en el que aparecen en el HTML

Ejemplo 1: Cambiar el Orden de Visualización

```
.contenedor-flex {
    display: flex;
}
.elemento-1 {
    order: 2;
}
.elemento-2 {
    order: 1;
}
.elemento-3 {
    order: 3;
}
```

En este ejemplo, hay 3 elementos dentro de un contenedor flexible. El elemento-1 tendrá un valor de order de 2, lo que significa que se mostrará después del elemento-2 (que tiene un valor de order de 1) pero antes del elemento-3 (que tiene un valor de order de 3).

Ejemplo 2: Orden Negativo

```
.elemento-4 {
    order: -1;
}
```

Si asignas un valor de order negativo a un elemento, se moverá hacia el principio del orden visual, incluso antes de los elementos con un valor de order de 0.

Ejemplo 3: Valor Predeterminado

```
.elemento-5 {
    /* Por defecto, el valor de order es 0 */
}
```

Si no se especifica un valor para order, el elemento se mostrará en el orden en el que aparece en el HTML.