CSS Flexbox

Objetivos

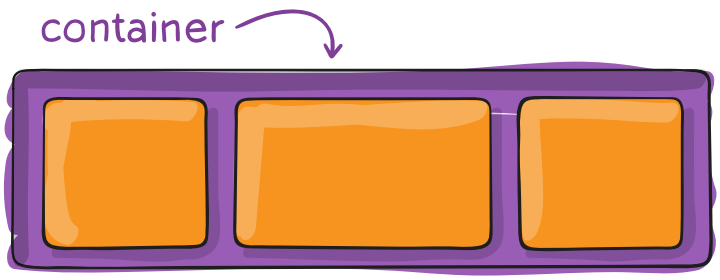
- Comprender como construir layout a través de Flexbox
- layout

Fundamentos

- css-tricks
- Mozilla
- https://cssreference.io/

Contenedor

Envolvemos todos nuestros items dentro de un div contenedor o parent, el cual tendrá la clase `display: flex`




```
<section class="flex-container">
    <article class="flex-child">1</article>
    <article class="flex-child">2</article>
    <article class="flex-child">3</article>
    <article class="flex-child">4</article>
</section>
```

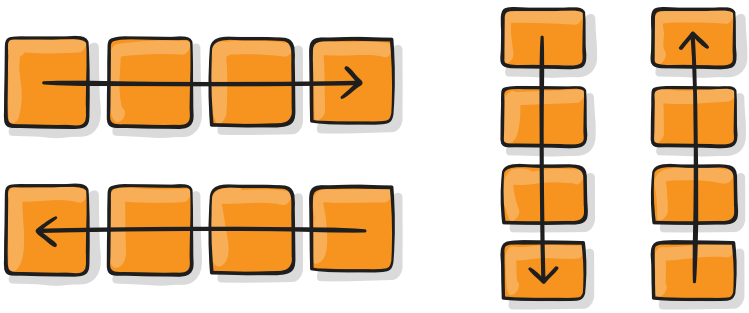
```
.flex-container {
    background-color: rgb(158, 158, 255);
    display: flex;
}

.flex-child {
    background-color: #fb7813;
    text-align: center;
    border: solid 5px black;
    margin: 10px;
    padding: 10px;
}
```

display:flex; display:inline-flex;

flex-direction

- **Esto establece el eje principal**, definiendo así la dirección en que se colocan los artículos flexibles en el contenedor.



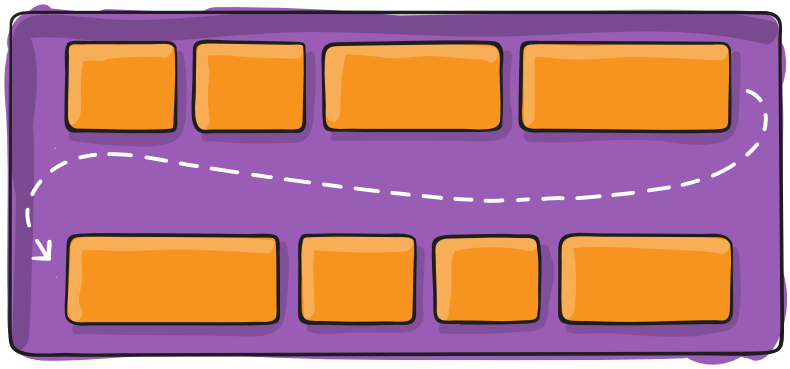
- row
- row-reverse
- column
- column-reverse

```
.flex-container {
    background-color: rgb(158, 158, 255);
    display: flex;
    flex-direction: row;
}
```

Juega con el siguiente código:

```
.flex-container {
    background-color: rgb(158, 158, 255);
    display: flex;
    flex-direction: column;
}
@media (min-width: 600px) {
    .flex-container {
        flex-direction: row;
    }
}
```

flex-wrap



- nowrap
- wrap
- wrap-reverse

Disminuir pantalla para visualizar cambios:

```
.flex-container {
    background-color: rgb(158, 158, 255);
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
}

.flex-child {
    background-color: #fb7813;
    text-align: center;
    border: solid 5px black;
    margin: 10px;
    padding: 10px;
    width: 70px;
}
```

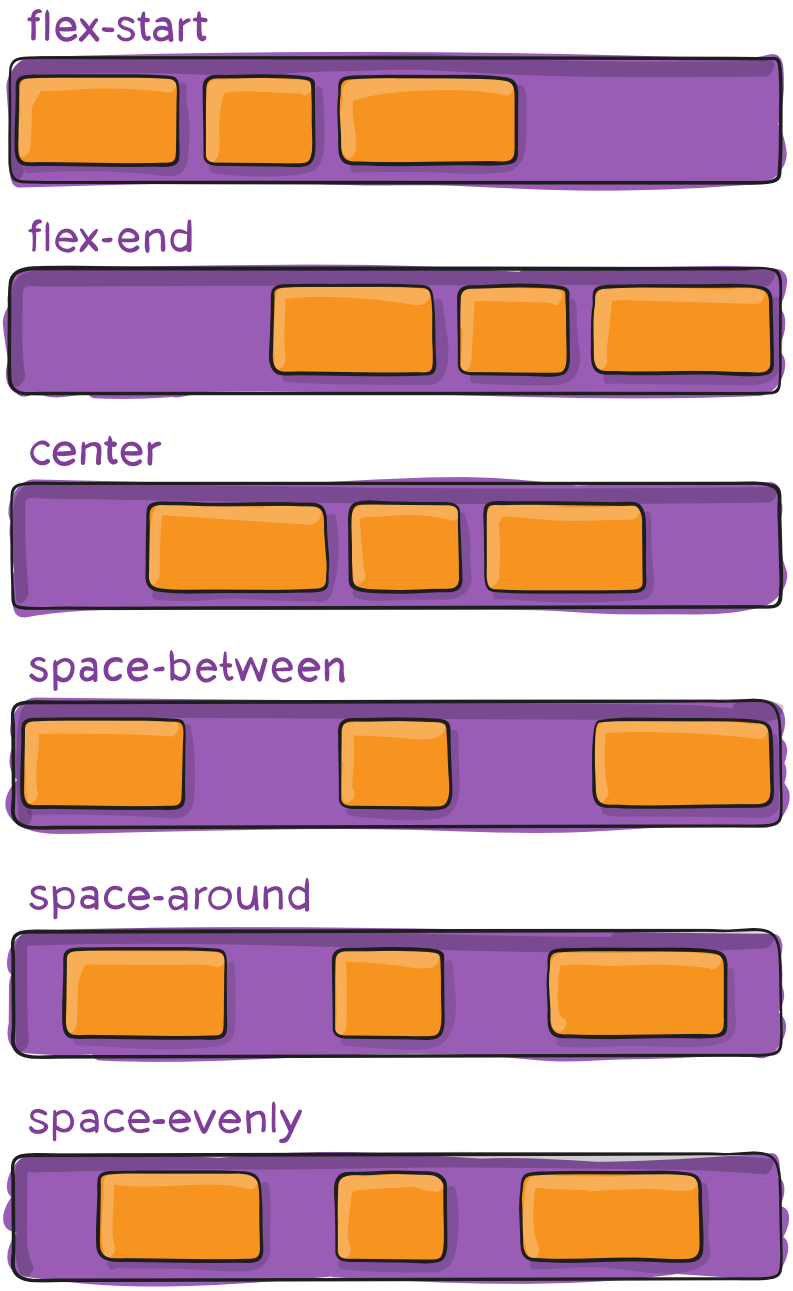
flex-flow (shorthand)

- Esta es una forma abreviada de las propiedades `flex-direction` y `flex-wrap`.

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

justify-content (eje principal)

- Esto define la alineación a lo largo del eje principal.
- Ayuda a distribuir el espacio libre adicional sobrante.
- También ejerce cierto control sobre la alineación de los elementos cuando desbordan la línea.



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

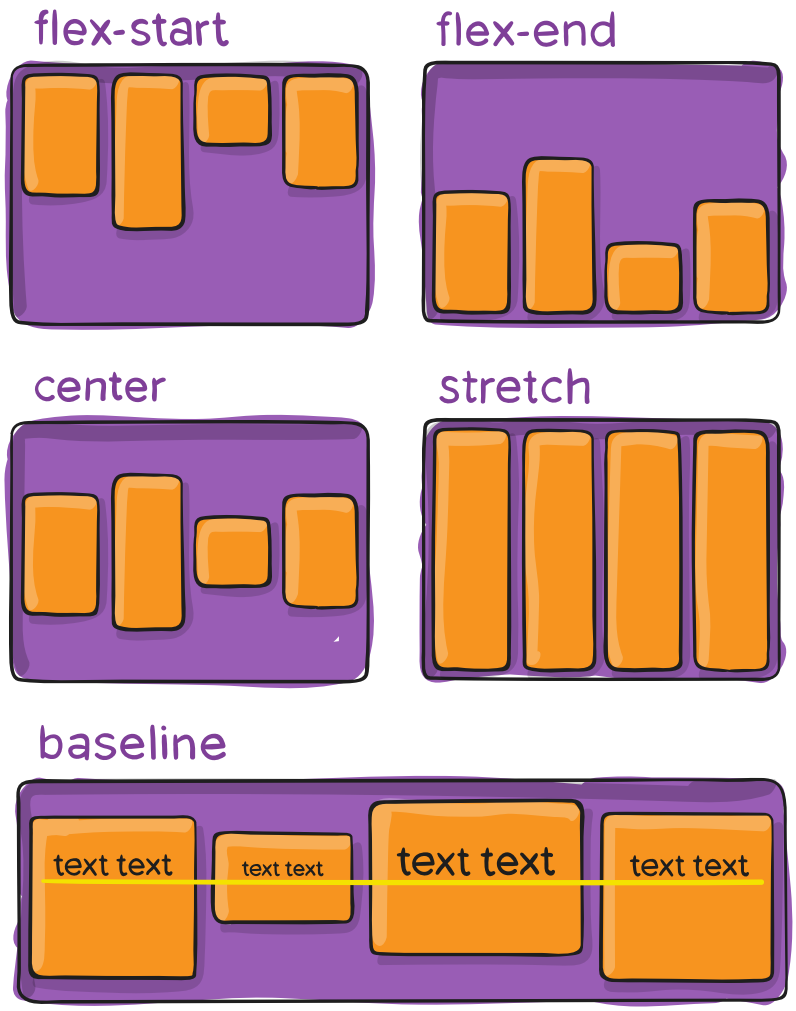
```
.flex-container {
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  justify-content: center;
}
```

Prueba agregando una altura al contenedor:

```
.flex-container {
    background-color: rgb(158, 158, 255);
    display: flex;
    flex-direction: column;
    flex-wrap: wrap;
    justify-content: space-evenly;
    height: 500px;
}
```

align-items (eje transversal)

- Esto define el comportamiento predeterminado de cómo se distribuyen los elementos flexibles a lo largo del **eje transversal** en la línea actual.



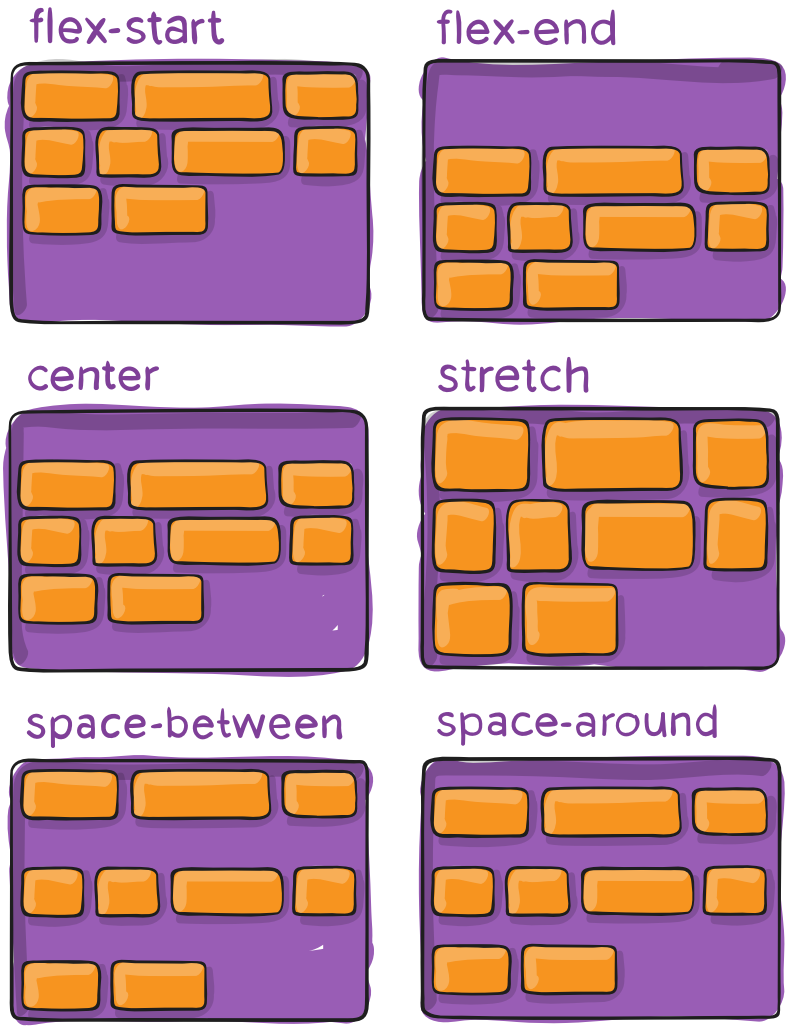
- stretch
- flex-start
- flex-end
- center
- baseline

```css
.flex-container {
    background-color: rgb(158, 158, 255);
    display: flex;
    flex-direction: column;
    flex-wrap: wrap;
    justify-content: space-evenly;
    align-items: flex-end;
    height: 500px;
}
```

```css
.flex-child {
    background-color: #fb7813;
    text-align: center;
    border: solid 5px black;
    margin: 10px;
    padding: 10px;
    /* width: 70px; */
}
```

align-content

- Esto alinea las líneas de un contenedor flexible cuando hay espacio adicional en el **eje transversal**, de forma similar a como justify-content se alinean los elementos individuales dentro del **eje principal**.
- Esta propiedad solo tiene efecto en contenedores flexibles de varias líneas.
- y además **flex-wrap** tiene que ser: **wrap** o **wrap-reverse**

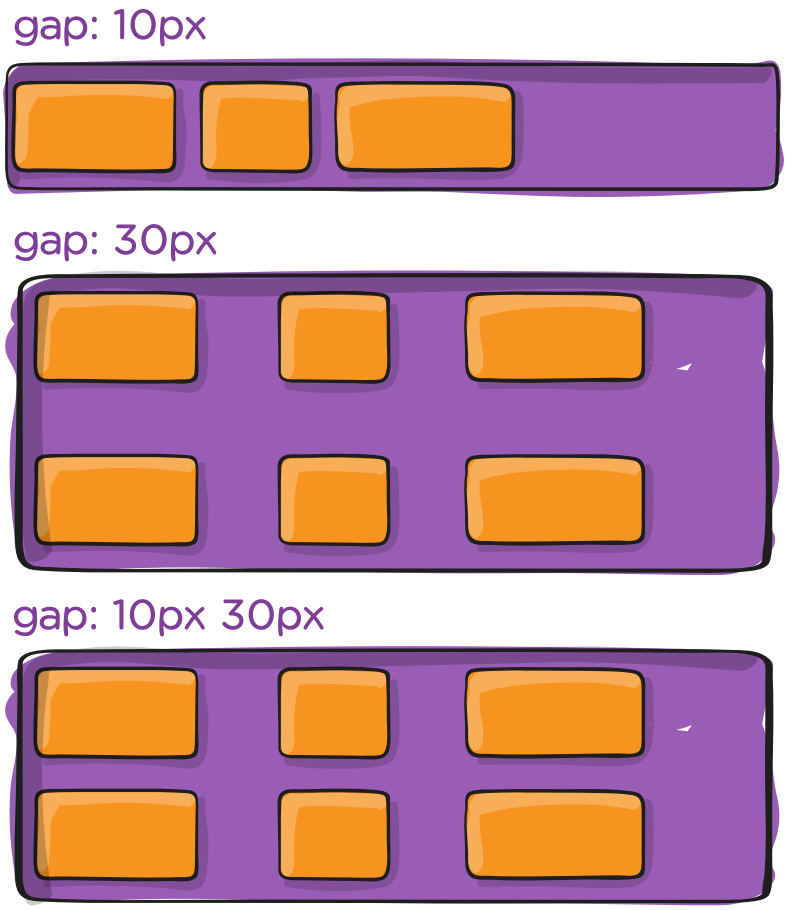


- flex-start
- flex-end
- space-between
- space-around
- space-evenly

```
.flex-container {
    background-color: rgb(158, 158, 255);
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    /* justify-content: space-evenly; */
    /* align-items: center; */
    align-content: space-between;
    height: 500px;
}
```

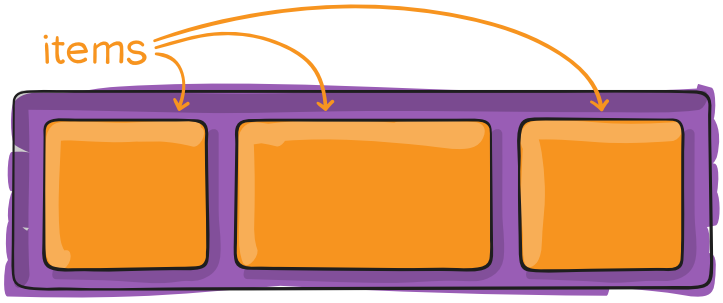
Gap

- controla explícitamente el espacio entre elementos flexibles. Se aplica ese espacio solo entre elementos que no están en los bordes exteriores.

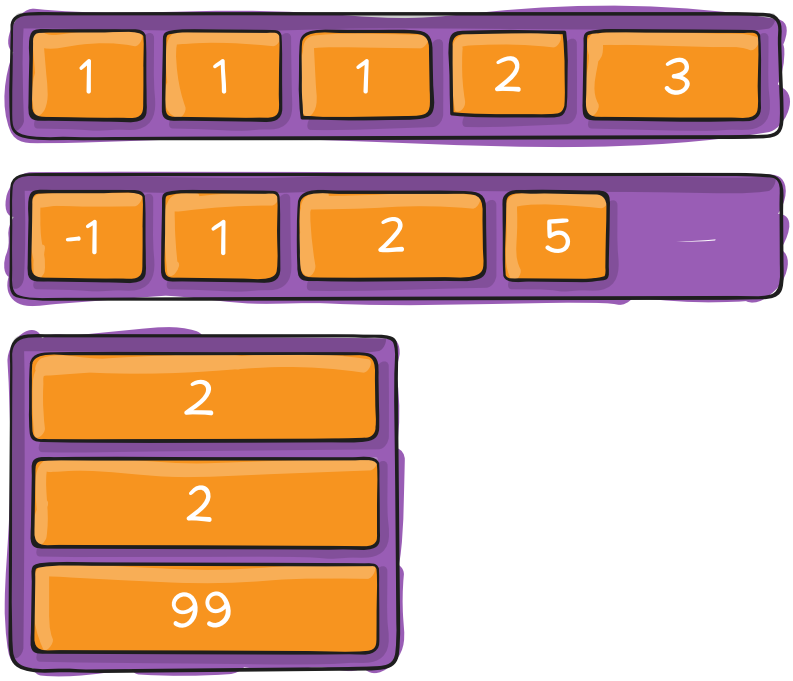



```
.flex-container {
    display: flex;
    gap: 10px;
    gap: 10px 20px; /* row-gap column gap */
    row-gap: 10px;
    column-gap: 20px;
}
```

Propiedades para los items o child

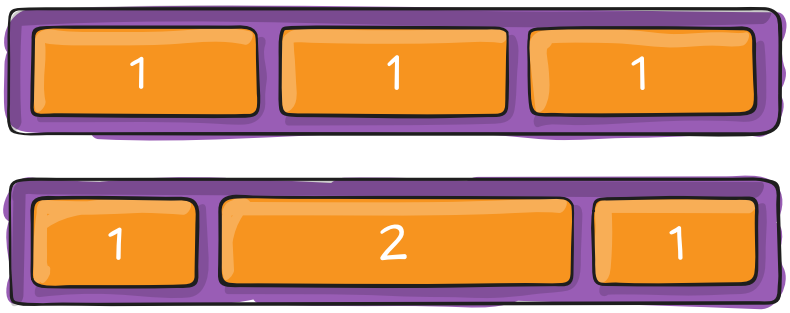


order




```
.flex-child:nth-child(3) {
    order: 1;
}
.flex-child:nth-child(2) {
    order: 2;
}
.flex-child:nth-child(1) {
    order: 3;
}
```

flex-grow (crecimiento flexible)



- Si todos los elementos se han establecido en `flex-grow:1`, el espacio restante en el contenedor se distribuirá por igual a todos item.
- Si uno de los item tiene un valor de 2, el espacio restante ocuparía el doble de espacio que los demás (o lo intentará, al menos).

```
.flex-container {
    background-color: rgb(158, 158, 255);
    display: flex;
    flex-flow: row wrap;
    gap: 10px;
}

.flex-child {
    background-color: #fb7813;
    text-align: center;
    border: solid 5px black;
    margin: 10px;
    padding: 10px;
    width: 50px;
}

.flex-child:nth-child(1) {
    flex-grow: 1;
}
.flex-child:nth-child(2) {
    flex-grow: 2;
}
.flex-child:nth-child(3) {
    flex-grow: 1;
}
```

flex-shrink (decrecimiento flexible)

- Esto define la capacidad de que un elemento flexible se encoja si es necesario.
- Por defecto `flex-shrink: 1;`, esto hace que todos disminuyan proporcionalmente, si aumentamos dicho número, este elemento disminuirá proporcionalmente más rápido que sus hermanos.
- Con `flex-shrink: 0;` hacemos que nuestro item no se reduzca de su tamaño establecido.

```
.flex-child:nth-child(1) {
    flex-grow: 1;
    width: 300px;
    flex-shrink: 0;
}
.flex-child:nth-child(2) {
    flex-grow: 2;
}
.flex-child:nth-child(3) {
    flex-grow: 1;
}
```

Prueba sacando la clase `flex-wrap: wrap;`

flex-basis

- Esto define el tamaño predeterminado de un elemento antes de que se distribuya el espacio restante.
- Obligamos a un item a partir de una proporción determinada:

```
.flex-child:nth-child(1) {
    flex-grow: 1;
    flex-basis: 300px;
}
```

flex

- El valor predeterminado es **flex: 0 1 auto;**
- Esta es la abreviatura de **flex-grow**, **flex-shrink** y **flex-basis** combinados.
- Los parámetros segundo y tercero (flex-shrink y flex-basis) son opcionales.
- Pero si lo configura con un solo valor numérico, como flex: 5; cambia **flex-basis a 0%**
- Se recomienda utilizar esta propiedad abreviada en lugar de establecer las propiedades individuales.

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

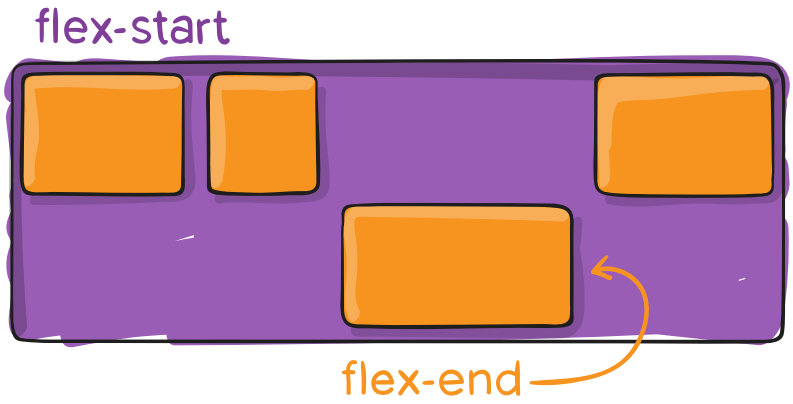
```
.flex-child:nth-child(1) {
    flex: 1 1 300px;
}
.flex-child:nth-child(2) {
    flex: 2;
}
.flex-child:nth-child(3) {
    flex: 1;
}
```

Notas

- `flex: 1 1 0;` Al establecer flex-grow y flex-shrink en 1 tanto para la imagen como para el contenido, nos aseguramos de que ambos elementos crezcan o se encojan para llenar el contenedor. Dado que ambos parten del mismo origen (0), cada uno crecerá o encogerá para ocupar la mitad del contenedor cada uno.
- `flex: 1 1 auto;` Sin embargo, si configuro el origen en automático para la imagen y el contenido, es justo suponer que la imagen ya tiene un ancho inherente, por lo que la imagen y el texto comenzarán a crecer y reducirse desde diferentes "puntos de partida".
- más info aquí
- flex-grow: 1 no tendrá efecto si existe un max-width.
- flex-shrink: 1 no tendrá efecto si existe un min-width.

align-self

Esto permite que se anule la alineación predeterminada (o la especificada por `align-items`) para elementos flexibles individuales.




```
.flex-container {
    background-color: rgb(158, 158, 255);
    display: flex;
    flex-flow: row nowrap;
    gap: 10px;
    height: 300px;
    align-items: flex-start;
}

.flex-child:nth-child(2) {
    flex: 2;
    align-self: center;
}
```

```
<div class="border flex-container h-700">
  <div class="item flex-1 aling-self">item 1</div>
  <div class="item flex-1">item 2</div>
  <div class="item flex-1">item 3</div>
</div>
```

Notas

- inline-flex
- Alinear icono y texto vertical

Alinear icono y texto vertical

```
.button {
  display: inline-flex;
  align-items: center;
}
```

Práctica

- práctica en clases
- Preview práctica
- Código
- FrontendMentor

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>CSS Flexbox</title>
        <link
            rel="stylesheet"
            href="https://necolas.github.io/normalize.css/8.0.1/normalize.css"
        />
        <link rel="stylesheet" href="style.css" />
    </head>
    <body>
        <main class="flex-container">
            <article class="flex-child">
                <div class="card-child-1"></div>
                <div class="card-child-2">
                    <div class="card-body">
                        <p>Perfume</p>
                        <h1>Gabrielle Essence Eau</h1>
                        <p>
                            Lorem ipsum dolor sit amet consectetur adipisicing
                            elit. Voluptatem assumenda modi nobis facilis
                            debitis adipisci commodi nam, ducimus impedit
                            accusantium voluptates maiores veniam perferendis
                            autem necessitatibus magni esse id voluptatum.
                        </p>
                    </div>
                </div>
            </article>
        </main>
    </body>
</html>
```

```
.flex-container {
    height: 100vh;
    padding: 0 20px;
    background-color: rgb(253, 201, 150);
    display: flex;
    justify-content: center;
    align-items: center;
```

```
}

.flex-child {
    border-radius: 10px;
    overflow: hidden;
    width: 500px;
    background-color: aliceblue;
    display: flex;
    flex-flow: row wrap;
}

.card-child-1 {
    min-width: 250px;
    min-height: 300px;
    background-image: url("https://belcorp.esika.com/cl/wp-
content/uploads/sites/3/2021/04/como-se-crean-perfumes-portada.jpg");
    background-position: center;
    background-size: cover;
    background-repeat: no-repeat;
    margin: 0;
    flex: 1 0 0;
}

.card-child-2 {
    min-width: 250px;
    flex: 1 0 0;
}

.card-body {
    padding: 20px;
}
```