¿Qué es CSS FlexBox?

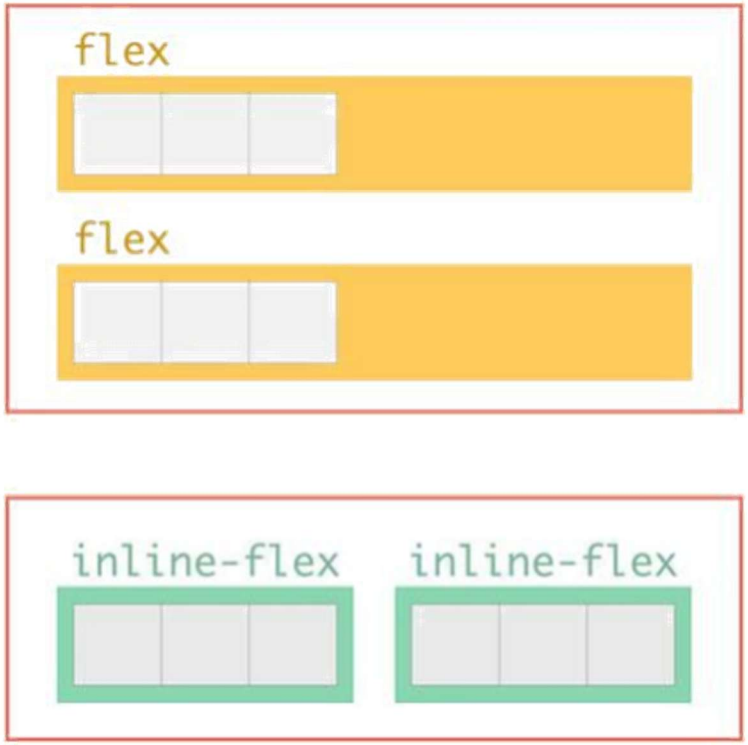
Rita de la Torre

CSS FlexBox es una técnica de CSS3 que consiste en un conjunto de propiedades y valores que van a permitir personalizar a un alto nivel de detalle la disposición de los elementos dentro de un contenedor flexible definido con la propiedad `display: flex;` o `display: inline-flex;`

Estas propiedades permiten situar y alinear fácilmente a sus elementos hijos a lo largo de dos ejes.

Los elementos hijos (llamados **flexbox items**) se distribuyen a lo largo del contenedor en una dirección: horizontal (row o fila que es la principal y la disposición **por defecto**) o vertical (columna o columna que es el eje secundario). Esta distribución se puede cambiar.

Es decir, Flex tiene un eje principal y un eje secundario o transversal. Es importante entenderlo al inicio ya que las propiedades de flex van a ser todas **relativas al eje principal**.

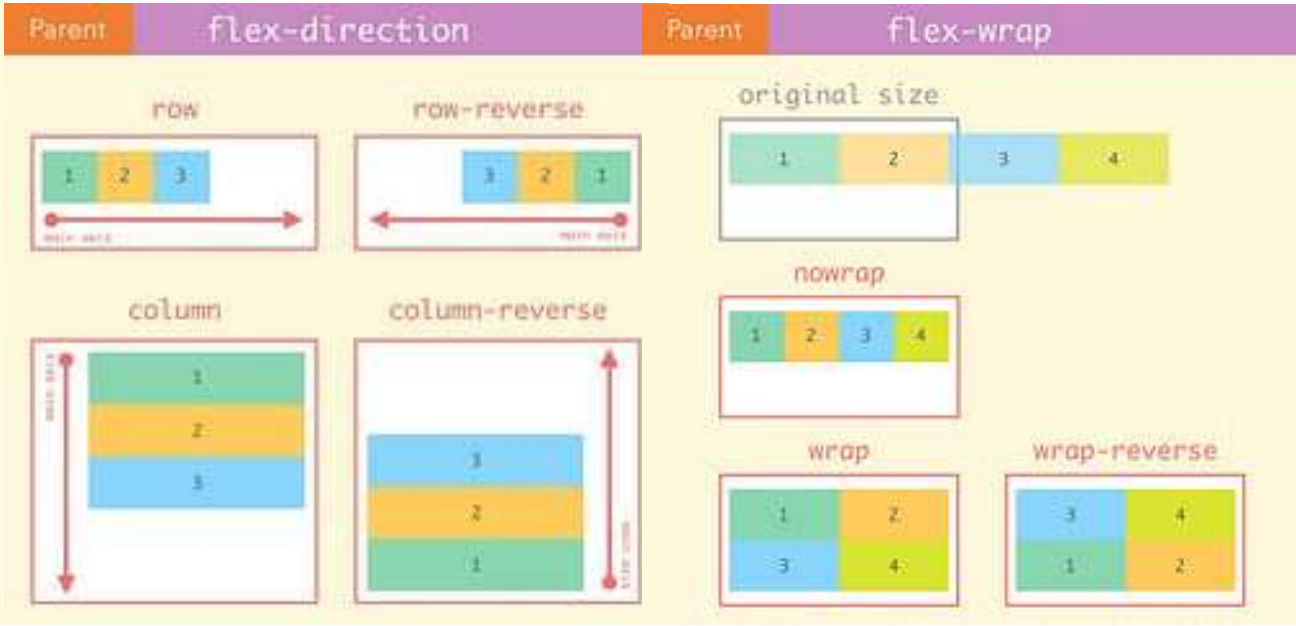


Propiedades de un contenedor padre tipo Flex

Rita de la Torre

La propiedad `flex-flow` es una propiedad atajo para las propiedades individuales `flex-direction` y `flex-wrap`. Ejemplo:


```
/* flex-flow = flex-direction + flex-wrap */
flex-flow: row nowrap;
```



Disposición por defecto de un contenedor Flex

Rita de la Torre


```
.contenedor-flex {
  display: flex
  flex-direction: row; /* distribución de los ítems */
  flex-wrap: wrap;
  /* controla lo que el navegador debe hacer si los hijos no caben en la línea
     sus valores pueden ser, wrap, nowrap o wrap-reverse */
}
```

Antes de continuar... IMPORTANTE:

La propiedad `display: flex` **no se hereda**, sólo afecta a los elementos hijos directos. Si un contenedor tiene `display: flex`, sus HIJOS DIRECTOS se convierten en flex items.

Sin embargo, si se desea que un hijo dentro de un contenedor flex también se comporte como un contenedor flexible, hay que asignarle explícitamente `display: flex`. Lo mismo ocurre con las demás propiedades de un flex container, como `flex-flow`, `justify-content`, `align-items`, y `align-content`: no se heredan automáticamente, sino que deben definirse donde sean necesarias.

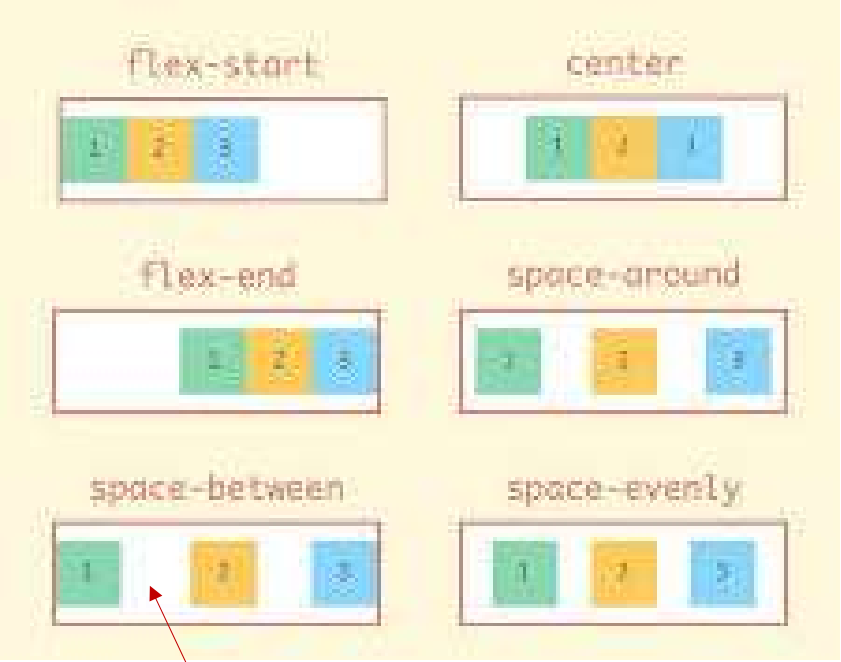
Los elementos hijos **por defecto** tienen `display: block` (para elementos en bloque como div, p, section, etc.) o `display: inline` (para elementos en línea como span, a, strong, etc.). Y su propiedad `position` es `static` por defecto.

Tanto un **flex container** como un **flex item** pueden tener la propiedad `position` modificada. La flexibilidad de Flexbox no impide que estos elementos sean posicionados de manera absoluta, fija, relativa o incluso sticky. Sin embargo, **si un flex ítem se define como** `absolute` o `fixed`, ya no responderá a las reglas de distribución de flexbox porque queda fuera del flujo del documento.

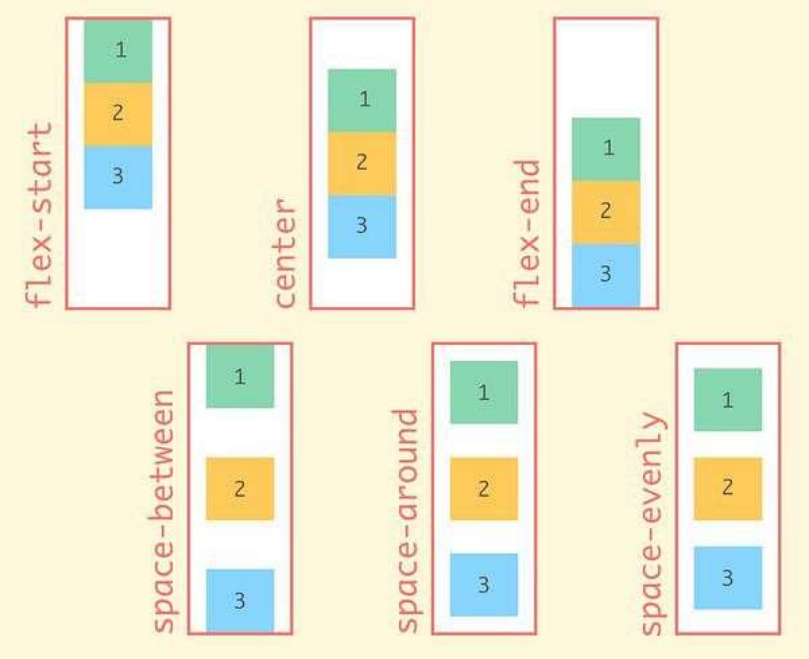
Alineación de los hijos eje principal: **justify-content**

Rita de la Torre

Si `flex-direction: row;`



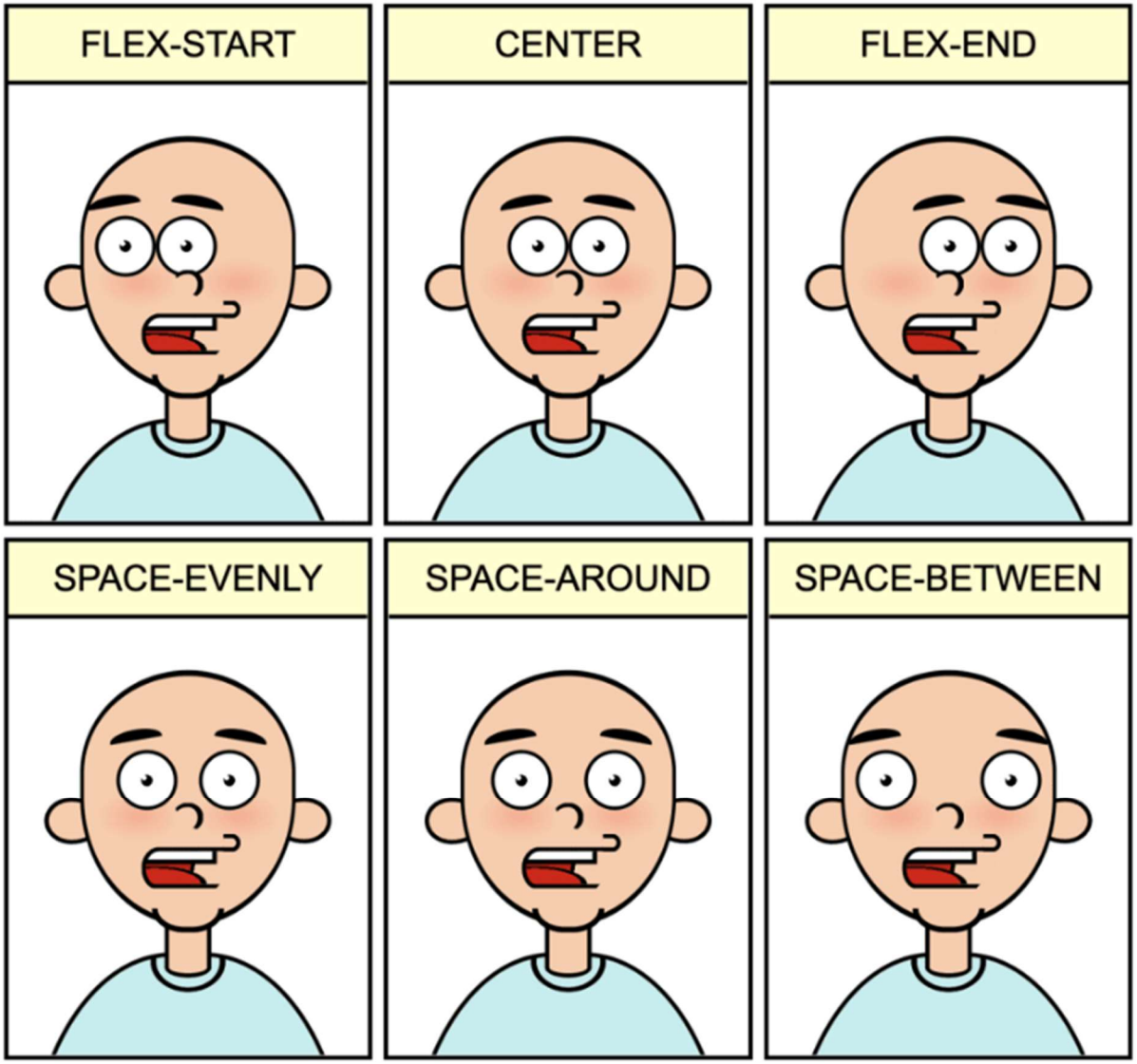
Si `flex-direction: column;`

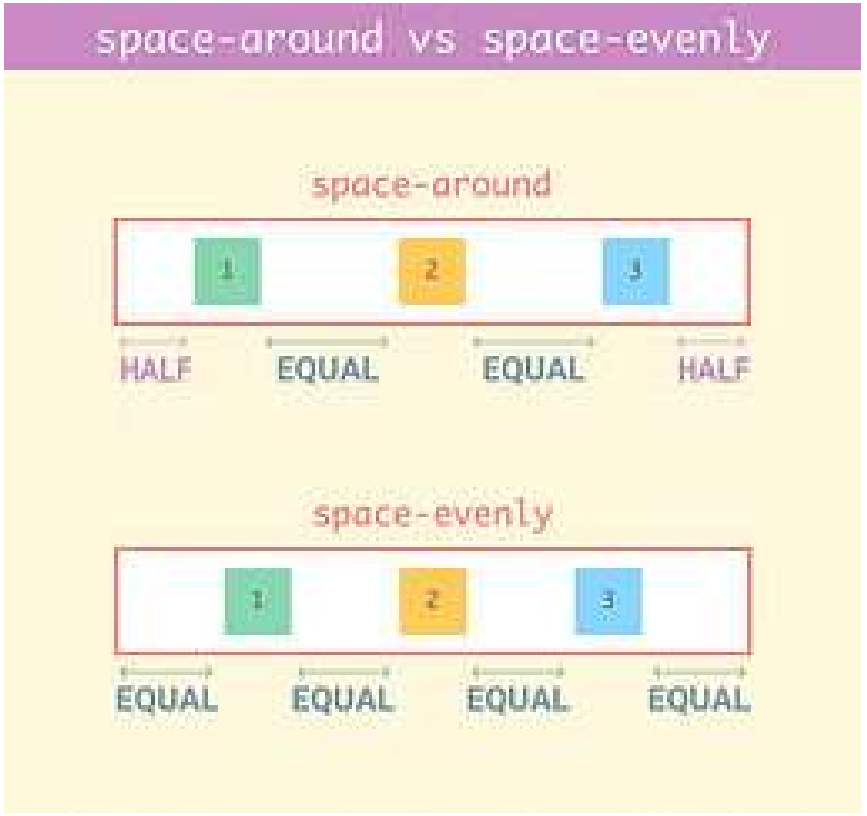



```
.contenedor-flex {
  display: flex
  flex-direction: row; /* distribución de los ítems */
  flex-wrap: wrap;
  justify-content: space-between;
}
```

justify-content comic

Rita de la Torre

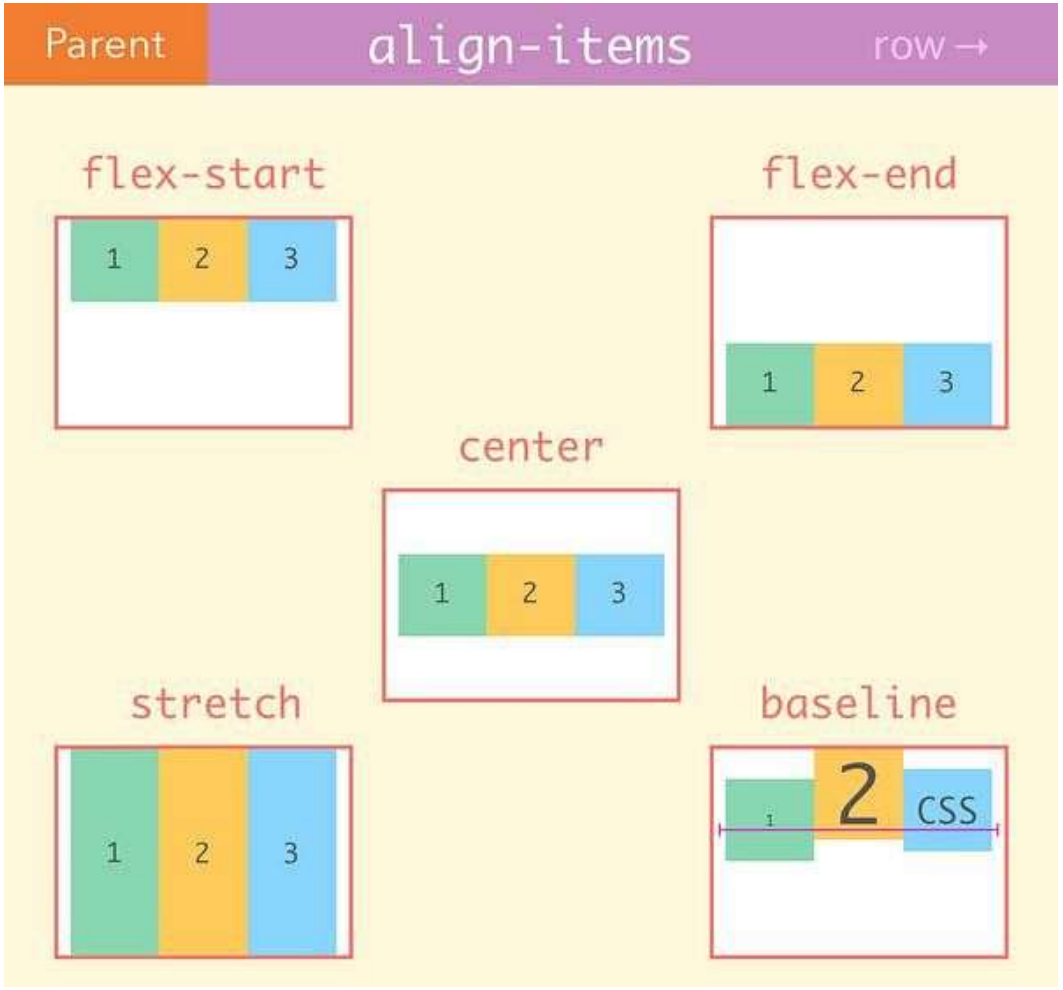




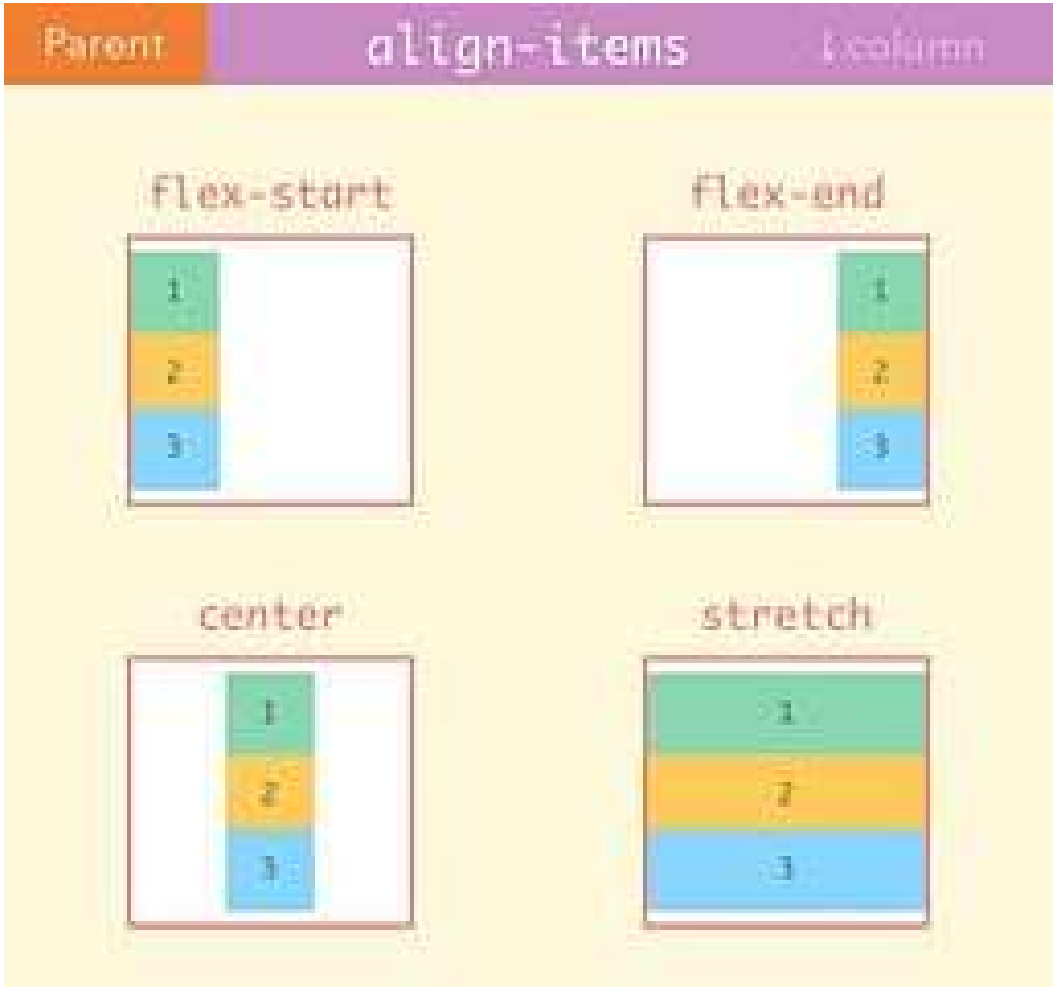
Alineación del eje secundario: **align-items**

Rita de la Torre

Si `flex-direction: row;`

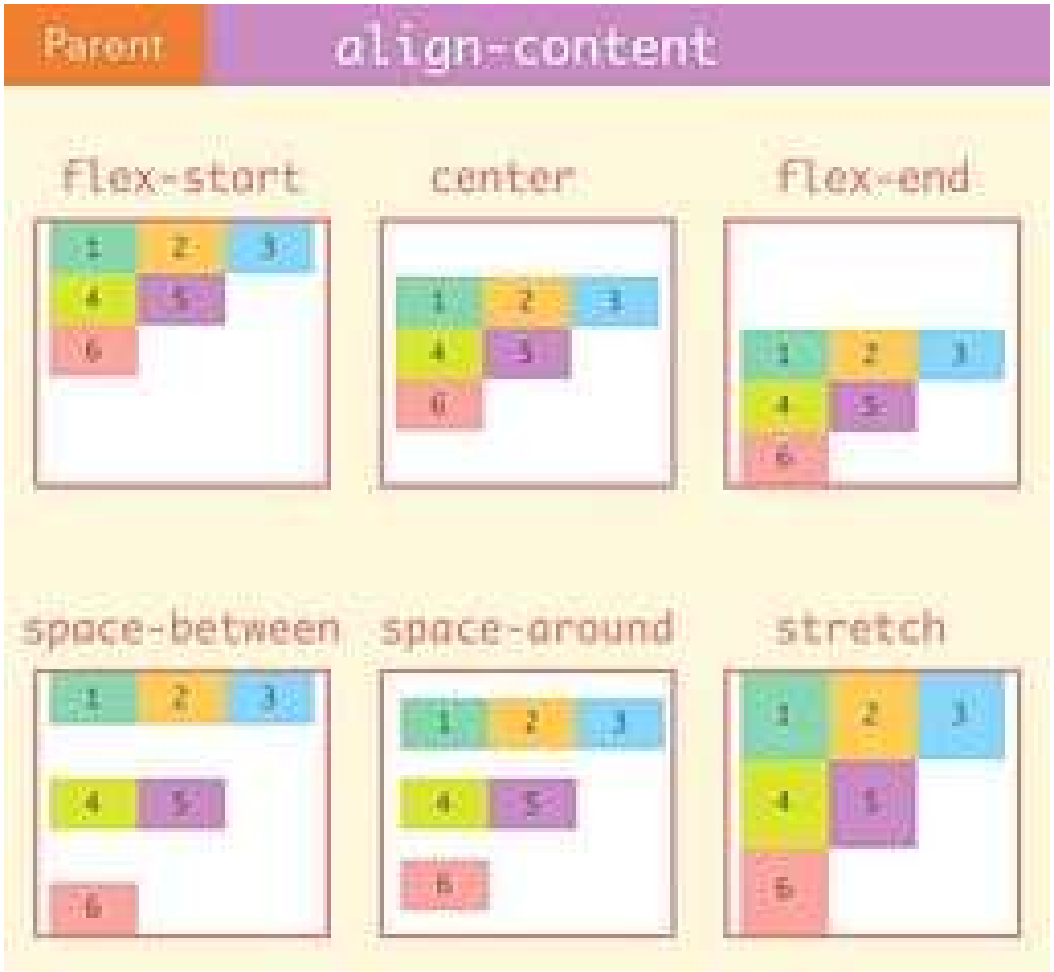


Si `flex-direction: column;`



Propiedad: **align-content**

Rita de la Torre

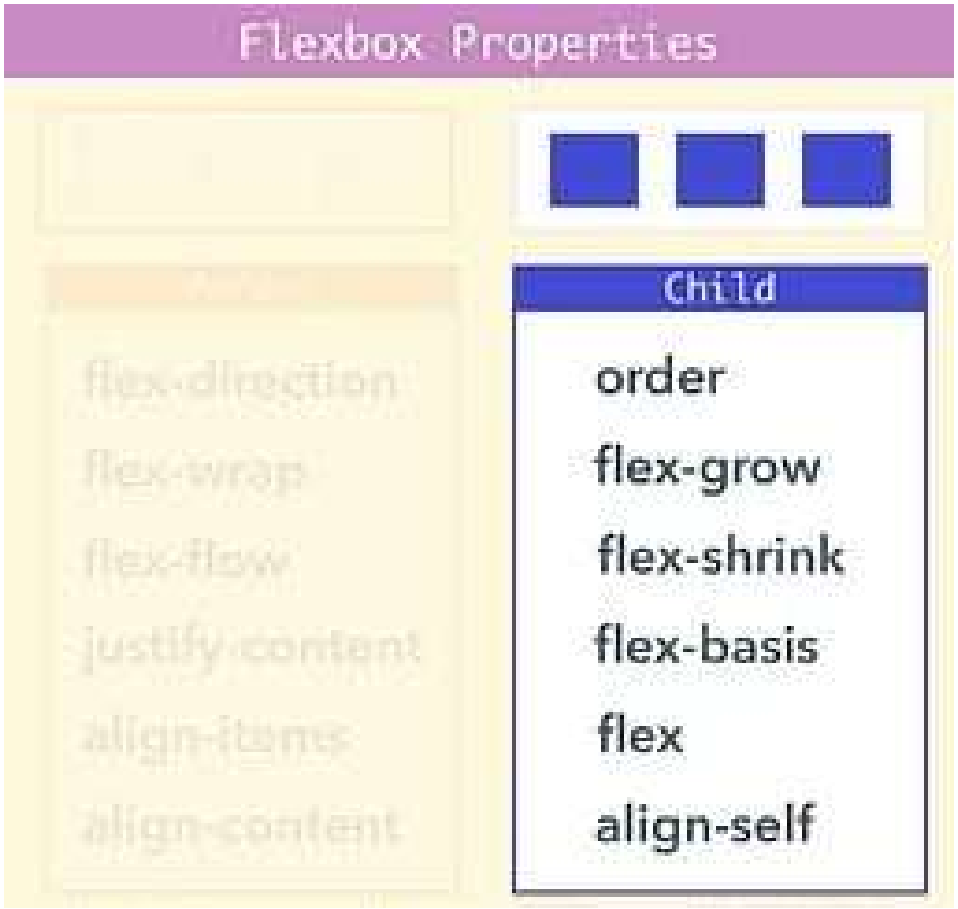


La propiedad `align-content` indica cómo se muestra los `flex-items` si se muestra más de una fila o columna.

Además Flexbox tiene la propiedad `gap` que permite controlar el espacio entre los flex items sin necesidad de usar márgenes. Ejemplo:

```
.contenedor {
  display: flex;
  gap: 15px;
  /* Espacio de 15px entre cada item */
}
```

https://www.youtube.com/watch?v=GShRBWz8jIM



`order` especifica el orden utilizado para disponer los elementos en el contenedor padre, en caso de querer romper el flujo natural del documento.

`flex-grow` y `flex-shrink` permiten especificar si crecen y decrecen (valor 1) en su dirección principal, los elementos hijos directos de un contenedor flexible (que tiene asignado `display:flex`) o no lo hacen (valor 0).

`flex-basis` establece el tamaño inicial de un elemento flexible (hijos directos de un contenedor flexible). Puede ser en px, %, auto, etc.

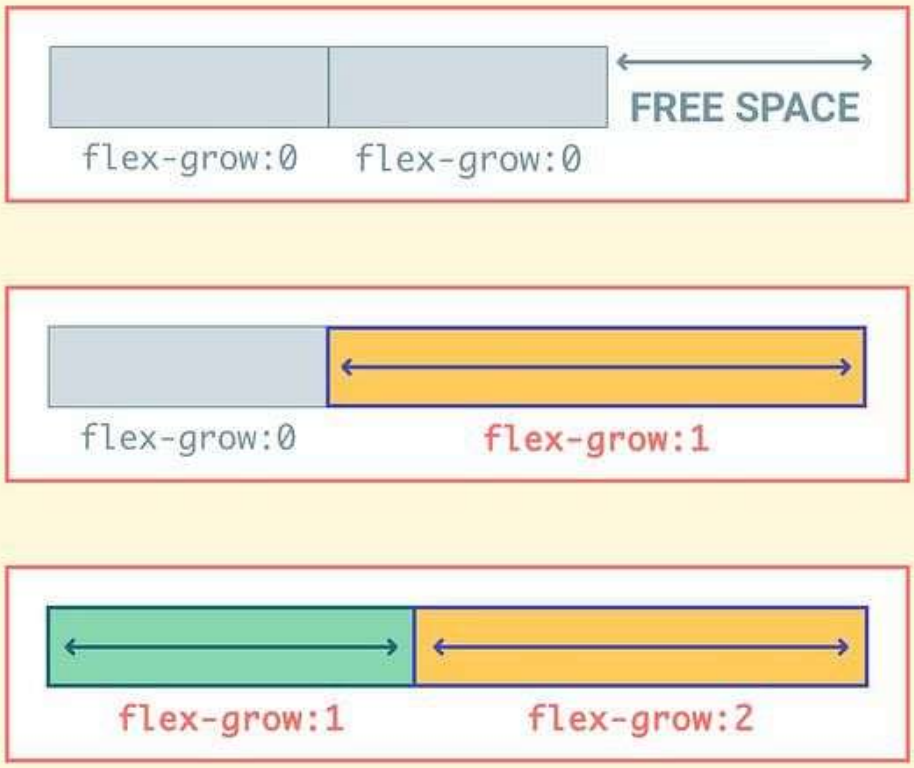
La propiedad `flex` es un atajo para las propiedades `flex-grow` + `flex-shrink` + `flex-basis`. Ejemplo: `flex: 0 1 auto;`

`align-self` anula el valor `align-items` de un flex ítem, reasignando otro.

Propiedad **flex-grow**

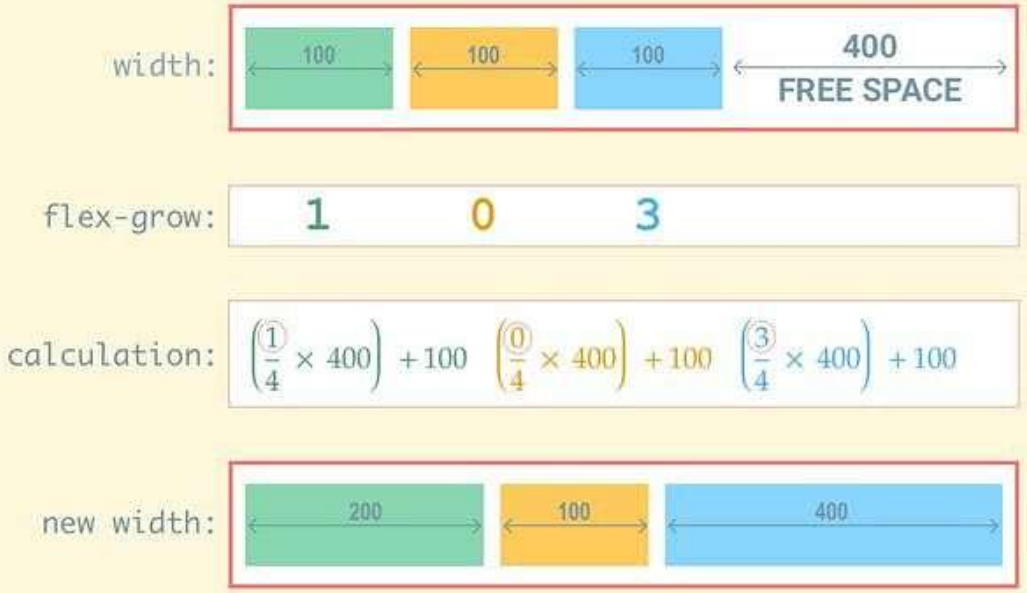
Rita de la Torre

Child	flex-grow



flex-grow calculation

$$\text{new width} = \left(\frac{\text{flex grow}}{\text{total flex grow}} \times \text{free space}\right) + \text{width}$$

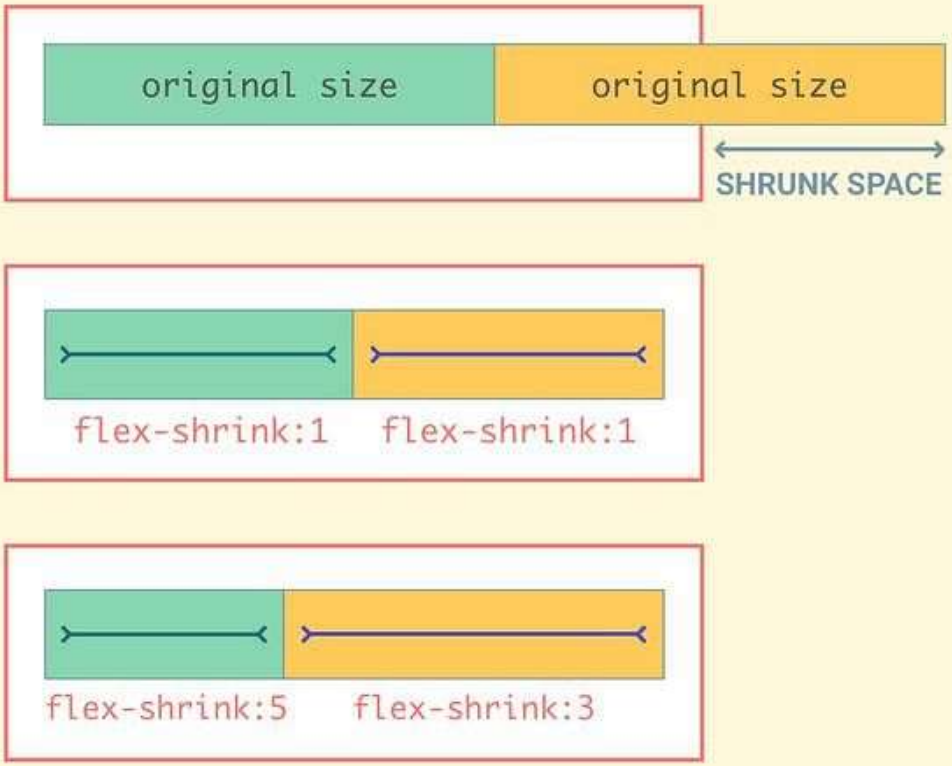


Propiedad **flex-shrink**

Rita de la Torre

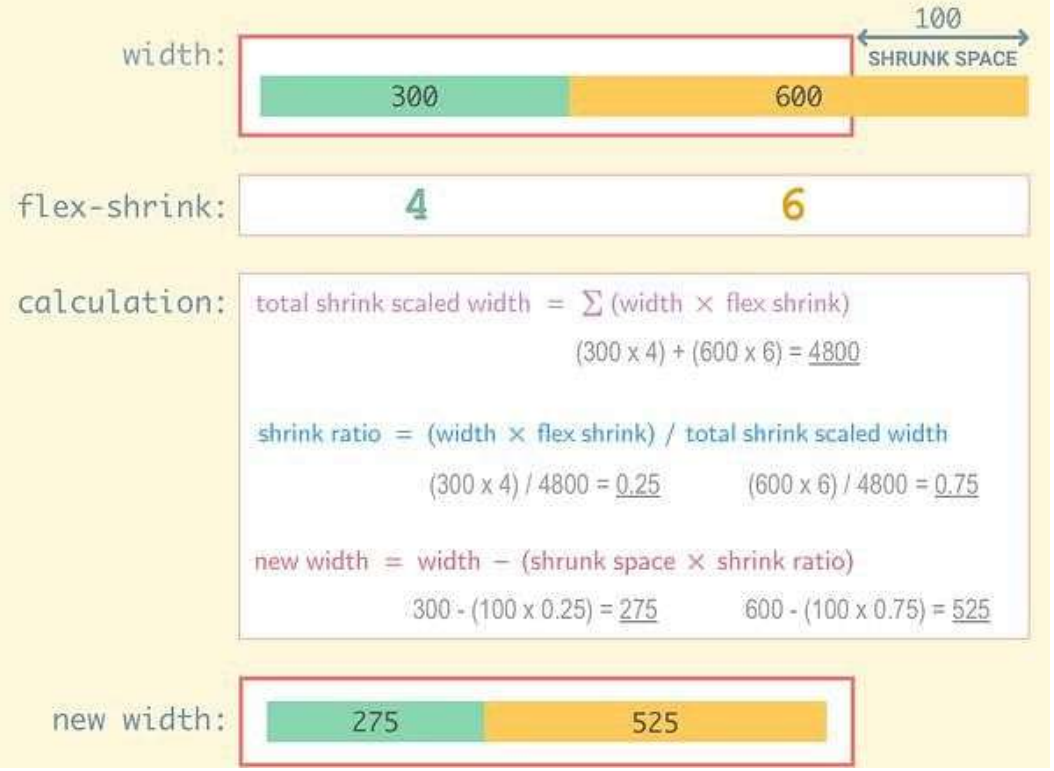
Child

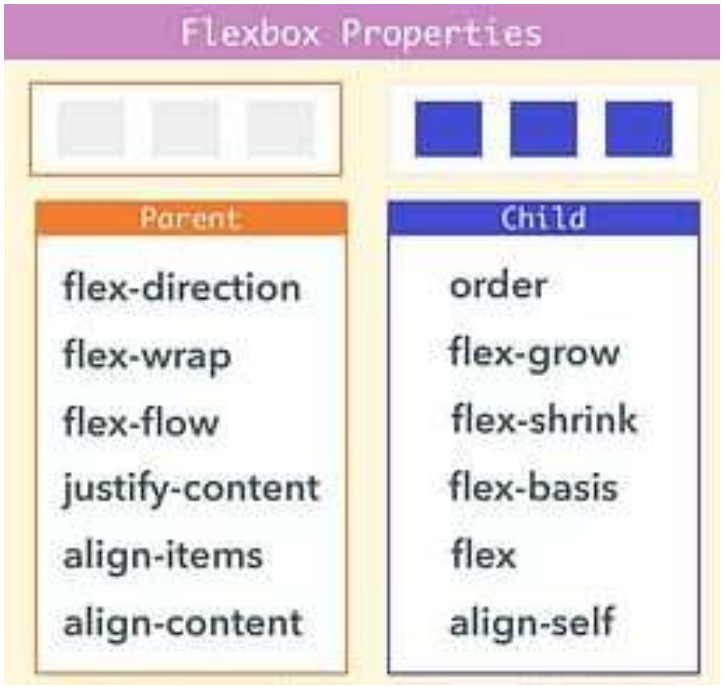
flex-shrink

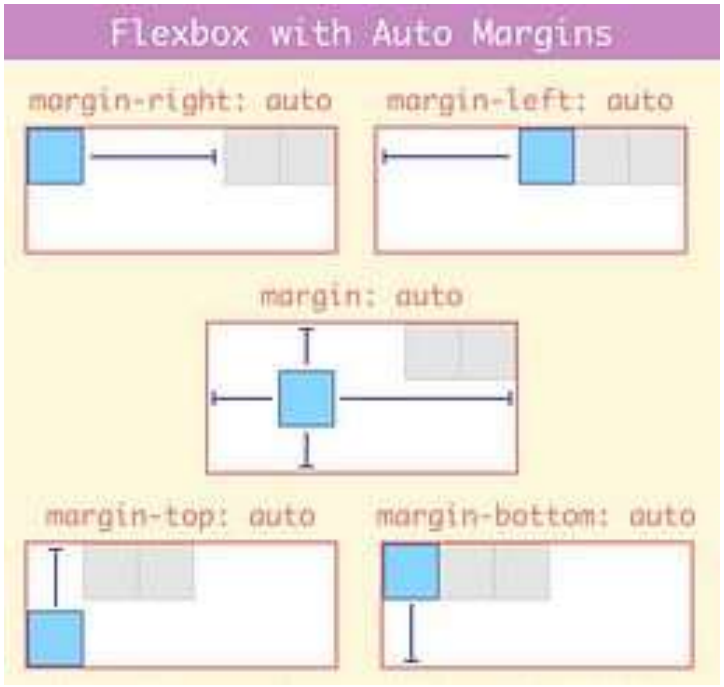


flex-shrink calculation

new width = width − (shrunk space × shrink ratio)







Imagenes: https://www.samanthaming.com/flexbox30/

Ejemplo: **Centrar elementos con Flexbox**

Rita de la Torre

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

Copia el ejemplo, intenta crear más **flex-ítems**, agrega o cambia propiedades hasta que entiendas cómo funciona esta técnica.

Una vez la entiendas, trata de rehacer el ejercicio de **GrInterest** con CSS Flexbox.

```css
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

.contenedor {
    display: flex;
    justify-content: center;
      /* Centra horizontalmente */
    align-items: center;
      /* Centra verticalmente */
    height: 100vh;  /* Altura de la pantalla */
    background-color: lightblue;
}

.item {
    width: 100px;
    height: 100px;
    background-color: coral;
    margin: 10px;
    display: flex;
    justify-content: center;
    align-items: center;
    font-size: 20px;
    color: white;
    font-weight: bold;
}
```