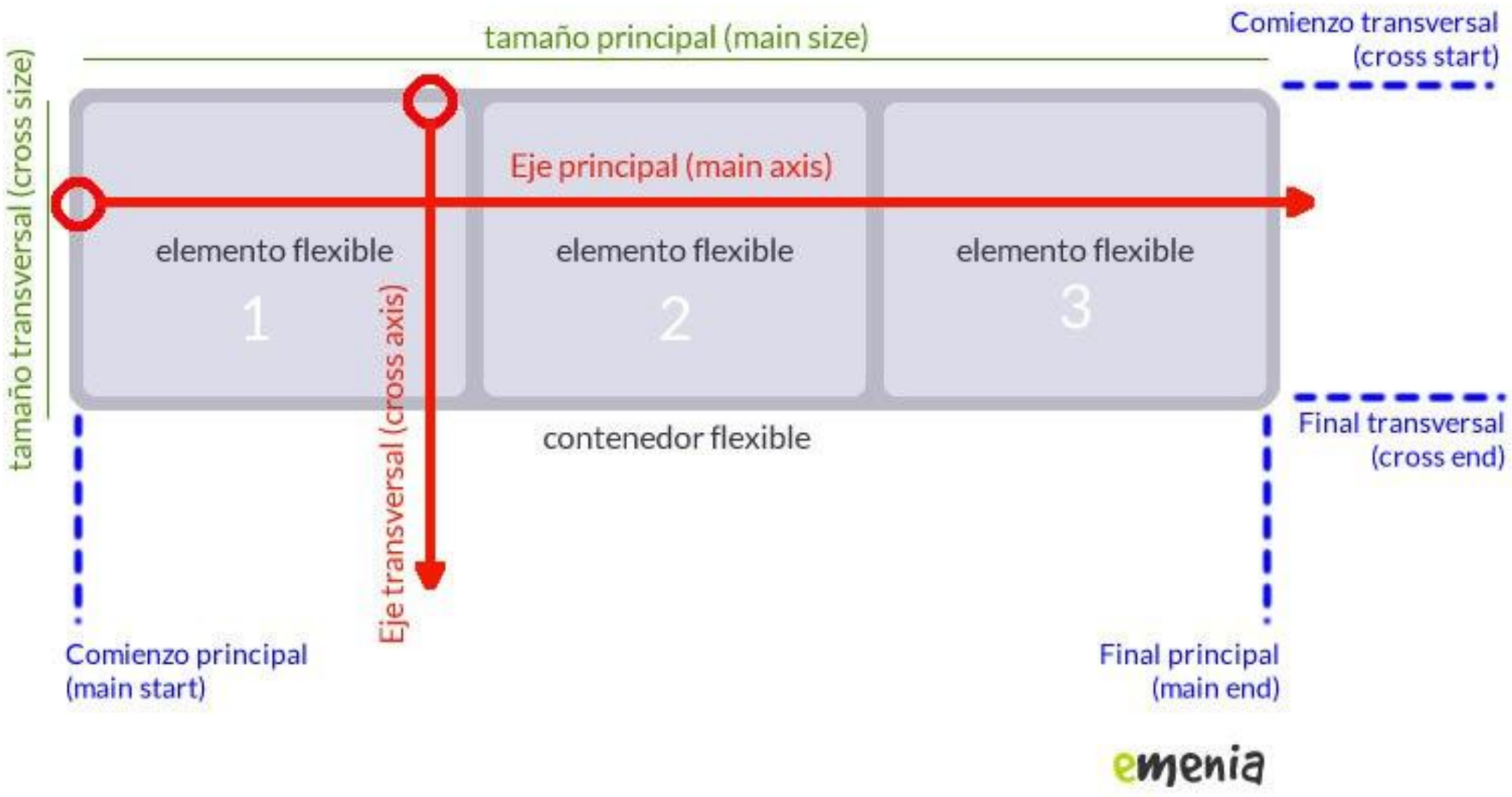
Cajas flexibles CSS3

FlexBox

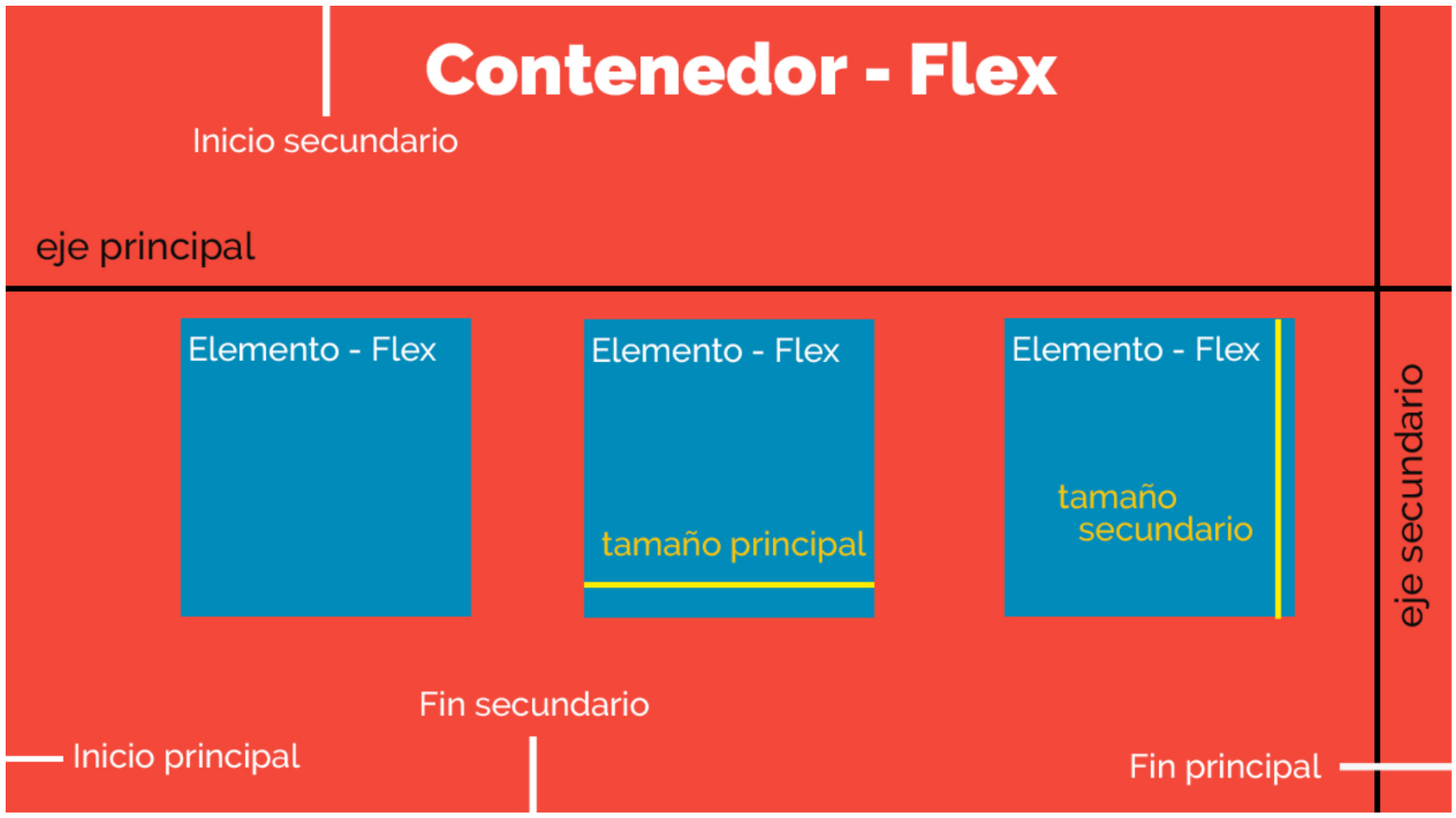
display: flex

Lo que caracteriza un diseño flexible es su habilidad para alterar el ancho y alto de sus elementos para ajustarse lo mejor posible al espacio disponible en cualquier dispositivo. Un contenedor flexible expande sus elementos para rellenar el espacio libre, o los comprime para evitar que rebasen el área prevista.

La declaración display: flex; **define un "contenedor flexible" y convierte de forma automática a sus "hijos" directos en "elementos flexibles"**. Un contenedor flexible tiene un Eje principal (main axis), que es la dirección en la cual se posicionan los elementos flexibles. Y tiene un Eje transversal, perpendicular al Eje principal. Ambos ejes tienen una serie de propiedades que controlan cómo se posiciona cada elemento flexible en relación a los demás. Puedes además poner contenedores flexibles uno dentro de otro, ya que la propiedad display no se hereda de forma automática.



Flexbox es un nuevo módulo de diseño en CSS3 para mejorar la forma en que hacemos **Responsive Design**, evitando así el uso de **float** (No uses float para hacer responsive design). Gracias a FlexBox facilitamos la forma en que posicionamos elementos, es más simple y usamos menos código.
Para empezar a trabajar con FlexBox primero tienes que entender la estructura, FlexBox está constituido por un padre **(Contenedor - Flex)** y sus hijos **(Elemento - Flex)**.



Contenedor - Flex: Es el elemento "padre" que contiene nuestros elementos "hijos", para definirlo se usa "flex" o "inline-flex" en la propiedad display.

Elemento - Flex: Estos son los elementos "hijos" que tendrán un comportamiento automático dependiendo lo que defina el elemento "padre".

Ejes: Cada diseño "FlexBox" está compuesto por dos ejes:
Eje principal: define la posición horizontal de los **elementos - Flex**
Eje secundario: define la posición vertical de los **elementos - Flex**.

Direcciones: Las partes de **inicio principal / fin principal** e **inicio secundario / fin secundario** del **Contenedor - Flex** definen el origen y final del flujo de **elementos - Flex**.

Dimensiones: Esto equivale a la anchura **(tamaño principal)** y la altura **(tamaño secundario)** del **elemento - Flex** que depe

Actualmente Flexbox es compatible con todos los navegadores.

Nota: float, clear y vertical-align no tienen efecto en un **elemento - flex**.

Definiendo FlexBox

Para empezar a usar FlexBox solo tenemos que definir flex en la propiedad display del elemento padre:

```
<section class="padre">
</section>
```

```
.padre{
    display: flex;}
```

Nota: Esto es lo único que necesitamos para configurar el contenedor principal y automáticamente todos sus hijos se convertirán en elementos **flex**.

Propiedades del contenedor - flex (Padre)

El **contenedor - flex (Padre)** tiene una serie de propiedades interesantes con las cuales podemos ir empezando a trabajar.

flex-direction

Valor por defecto: **row**

Esta propiedad define las direcciones del **eje principal**, es decir, hacia donde se moverán los **Elementos - Flex**, tanto horizontalmente como verticalmente.

Horizontalmente:

```
.padre {
  flex-direction: row;
}
```

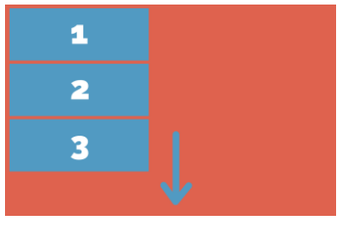
Los **elementos - flex** se apilan en una fila de izquierda a derecha.

```
.padre {
  flex-direction: row-reverse;
}
```

Los **elementos - flex** se apilan en una fila de derecha a izquierda

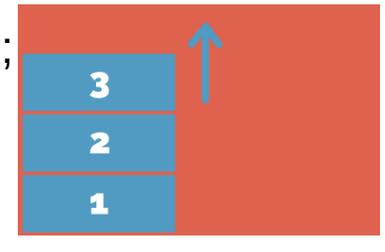
Verticalmente:

```
.padre {
  flex-direction: column;
}
```



Los **elementos - flex** se apilan en una columna de arriba abajo.

```
.padre {
  flex-direction: column-reverse;
}
```



Los **elementos - flex** se apilan en una columna de abajo arriba.

flex-wrap

Valor por defecto: **nowrap**

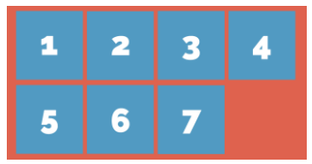
Al manejar Flexbox el concepto inicial es fijar los **elementos - flex** en una sola línea, pero con la propiedad **flex-wrap** controla

```
.padre {
  flex-wrap: nowrap;
}
```

Los **elementos - flex** se muestran en una fila y se pueden encoger dependiendo de su **contenedor - flex**.

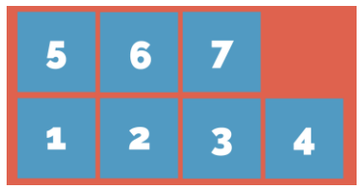

```
.padre {
  flex-wrap: wrap;
}
```

Los **elementos - flex** se muestran en varias filas (si es necesario), de izquierda a derecha y de arriba abajo.




```
.padre {
  flex-wrap: wrap-reverse;
}
```

Los **elementos - flex** se muestran en varias filas (si es necesario), de izquierda a derecha y de abajo arriba.



flex-flow

Valor por defecto: **row nowrap**

Esta propiedad es una forma rápida para establecer las propiedades flex-direction y flex-wrap.

```
.padre {
  flex-flow: <flex-direction> || <flex-wrap>;
}
```

Ejemplo:

```
.padre {
  flex-flow: column-reverse wrap;
}
```

justify-content

Valor por defecto: **flex-start**
La propiedad justify-content alinea los **elementos - flex** a lo largo del **eje principal** de la línea actual **(contenedor - flex)**.

```
.padre {
  justify-content: flex-start;
}
```

Los **elementos - flex** están alineados con el lado izquierdo del **contenedor - flex.**


```
.padre {
  justify-content: flex-end;
}
```

Los **elementos - flex** están alineados con el lado derecho del **contenedor - flex**.


```
.padre {
  justify-content: center;
}
```

Los **elementos - flex** están alineados al centro del **contenedor - flex**.


```
.padre {
  justify-content: space-between;
}
```

Los **elementos - flex** tienen la misma distancia entre ellos, pero el primero y el último **elemento - flex** están alineados con los bordes del **contenedor - flex**.


```
.padre {
  justify-content: space-around;
}
```

Los **elementos - flex** tienen la misma distancia entre ellos, incluso el primero y el último **elemento - flex**.

align-items

Valor por defecto: **stretch**

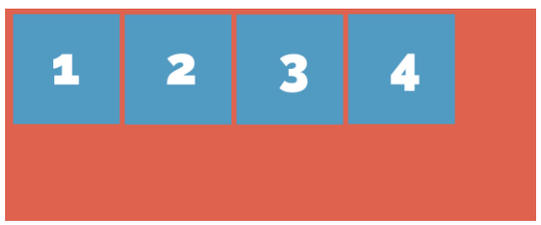
La propiedad align-items es muy similar a la propiedad justify-content, pero este es en la dirección del **eje secundario**.

```
.padre {
  align-items: stretch;
}
```

Los **elementos - flex** ocupan toda la altura (o anchura) de **inicio secundario** a **fin secundario**.

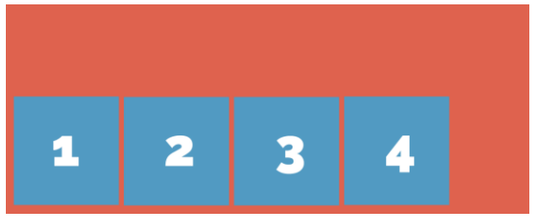

```
.padre {
  align-items: flex-start;
}
```

Los **elementos - flex** se apilan al **inicio secundario** del **contenido - flex**



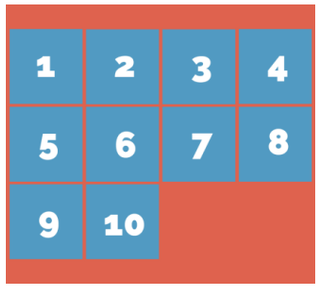

```
.padre {
  align-items: flex-end;
}
```

Los **elementos - flex** se apilan al **fin secundario** del **contenido - flex**.



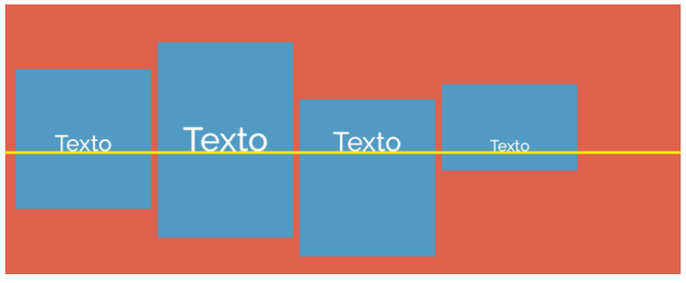

```
.padre {
  align-items: center;
}
```

Los **elementos - flex** se apilan al centro del **eje secundario**.




```
.padre {
  align-items: baseline;
}
```

Los **elementos - flex** se alinean de manera que sus líneas base quedan iguales.



align-content

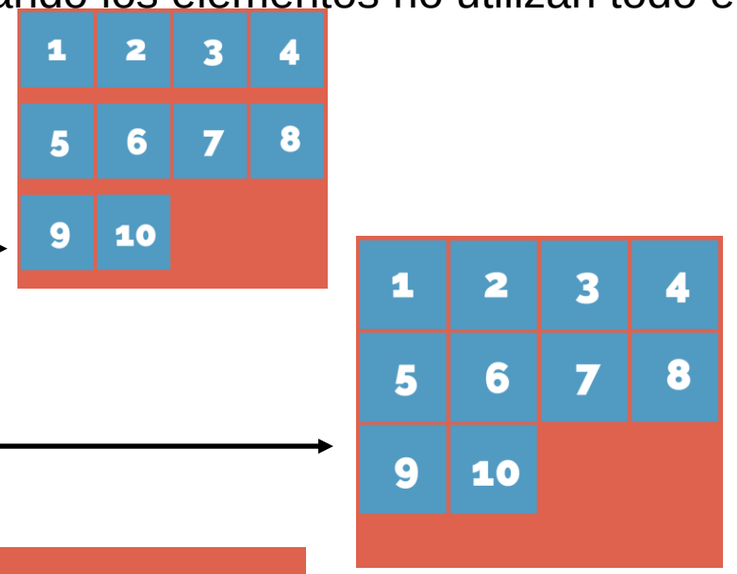
__Nota:__ Esta propiedad solo funciona cuando el __contenedor - flex__ tiene varias líneas de __elementos - flex__. Si solo existe una sola línea esta propiedad no tiene efecto.

Valor por defecto: **stretch**

La propiedad **align-content** alinea las líneas de un **contenedor - flex** cuando los elementos no utilizan todo el espacio dispo

```
.padre {
  align-content: stretch;
}
```

Los **elementos - flex** se muestran con espacio distribuido después de cada fila de **elementos - flex**.

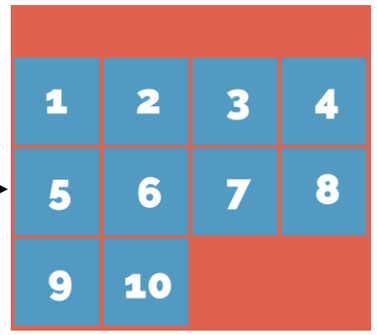



```
.padre {
  align-content: flex-start;
}
```

Los **elementos - flex** se apilan hacia el **inicio secundario** del **contenedor - flex**

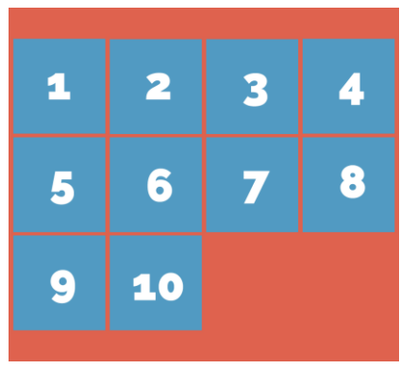

```
.padre {
  align-content: flex-end;
}
```

Los **elementos - flex** se apilan hacia el **fin secundario** del **contenedor - flex**



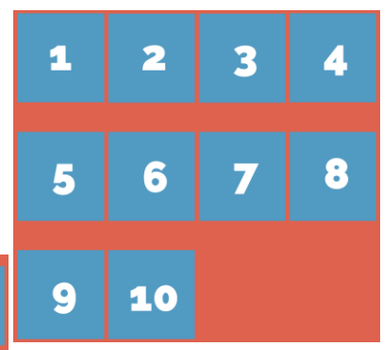

```
.padre {
  align-content: center;
}
```

Las filas se apilan en el centro del **eje secundario**.



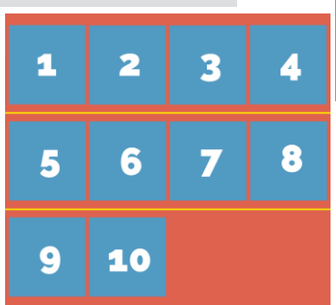

```
.padre {
  align-content: space-between;
}
```

Las filas de los **elementos - flex** tienen la misma separación entre ellos, pero la primera y última fila están alineadas con los bordes del **contenedor - flex**.




```
.padre {
  align-content: space-around;
}
```

Los **elementos - flex** tienen la misma separación en cada fila de **elementos - flex**



Propiedades de los elementos - flex (Hijos)

Al igual que el **contenedor - flex**, los **elementos - flex** tienen propiedades interesantes, con las cuales podremos hacer cosas aun más increíbles.

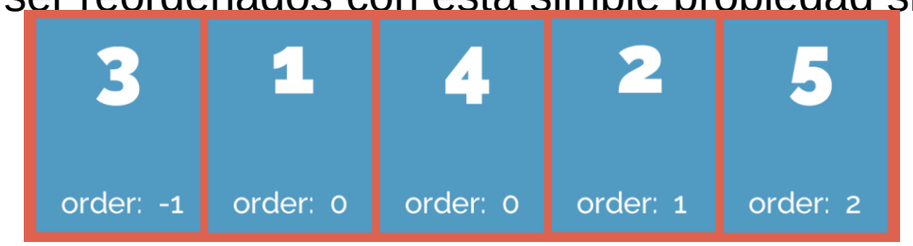
order

Valor por defecto: **0**

Con esta propiedad controlamos el orden de los **elementos - flex** que están dentro del **contenedor - flex**.

Los **elementos - flex** pueden ser reordenados con esta simple propiedad sin la necesidad de reestructurar el código HTML.

```
.hijo {
  order: <integer>;
}
```



flex-grow

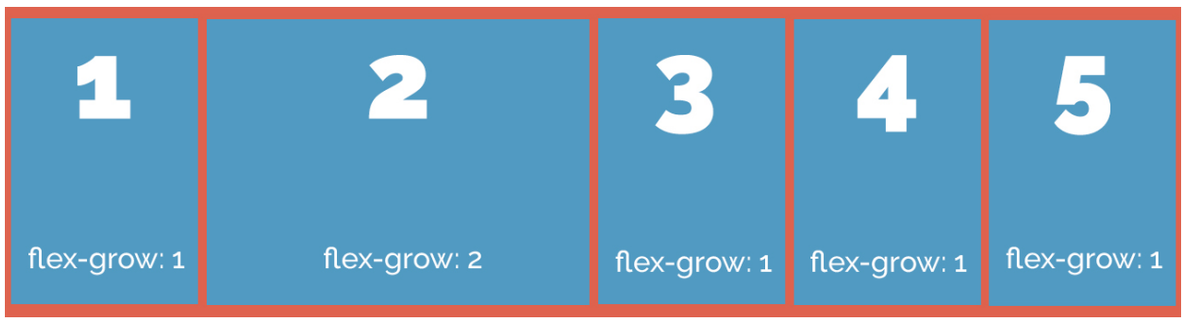
Valor por defecto: **0**

Esta propiedad especifica el factor que determina hasta que punto un **elemento - flex** crecerá en relación con el resto de los **el**

```
.hijo {
  flex-grow: <number>;
}
```

Si los **elementos - flex** tienen el mismo valor para flex-grow, todos van a tener el mismo tamaño, según el **contenedor - flex**.

Al definir que en el segundo **elemento - flex** la propiedad flex-grow va a tener un valor de **dos**, este va a tomar un ancho que equivale a la suma de dos elementos.



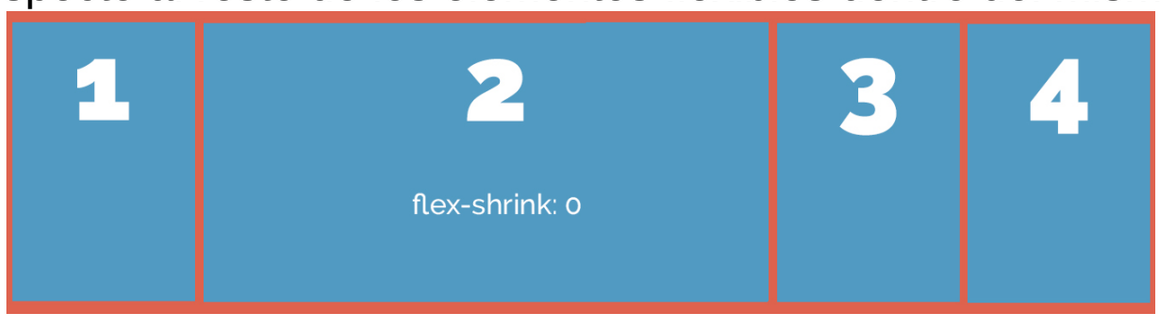
flex-shrink

Valor por defecto: 1

Esta propiedad especifica cómo el **elemento - flex** se reducirá con respecto al resto de los elementos flexibles dentro del mism

```
.hijo{
  flex-shrink: <number>;
}
```

Por defecto, todos lo **elementos - flex** se pueden reducir, pero si le ponemos el valor de 0 no se van a encoger, si no, que van a mantener el tamaño original.



flex-basis

Valor por defecto: **auto**

Esta propiedad tiene los mismos valores que las propiedades **width** y **height** y especifica el tamaño principal del **elemento -**

```
.hijo {
  flex-basis: auto | <width>;
}
```

Aquí especificamos que el quinto **elemento - flex** dicta el tamaño inicial del elemento.

flex

Valor por defecto: 0 1 auto

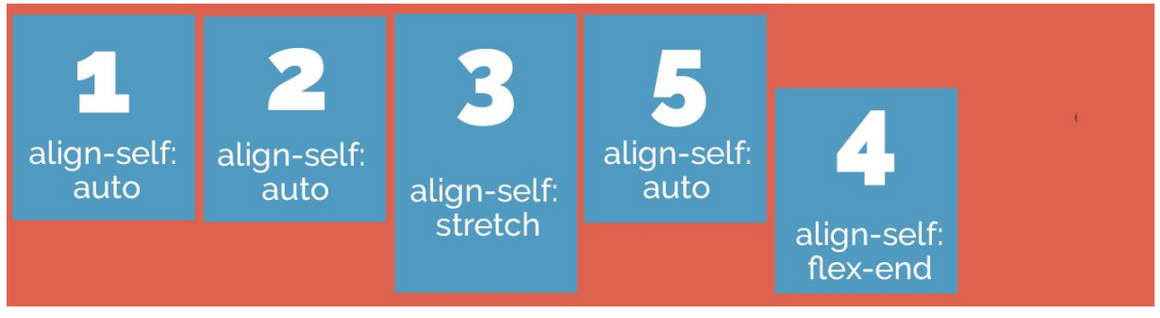
Con esta propiedad solo es la abreviatura de **flex-grow**, **flex-shrink** y **flex-basis**. Entre otros valores que también se pueden configurar para auto (1 1 auto) y none (0 0 auto).

align-self

Valor por defecto: **auto**

Esta propiedad permite la alineación por defecto (o la especifica por **align-items**) para ser anulados por **elementos - flex** individuales.

```
.hijo {
  align-self: auto | flex-start | flex-end | center | baseline | stretch;
}
```



El tercer y quinto **elemento - flex** han anulado la alineación a través de la propiedad **align-self.**

Ejemplo

Creamos un catálogo de viajes donde cada viaje tiene un título, una foto, una descripción y un botón de "Más información". Lo que queremos es que cada entrada tenga el mismo tamaño (anchura y altura), que la foto esté sobre el texto y que el botón de "Más información" esté siempre alineado abajo. Además, cuando cambiemos el ancho de pantalla se ajustará todo perfectamente sin necesidad de haber calculado complejos porcentajes. Con FlexBox conseguir esto es facilísimo!

XHTML

```
<section id='viajes'>
    <div class='viajes-item uno'>
        <h2>Malta, la isla mediterránea por conocer</h2>
        <p>Dignissim placerat vel aenean porta, magna ac scelerisque facilisis, rhoncus est! Urna
        <img src='images/malta.jpg' alt='' />
        <button>MÁS INFO</button>
    </div>
    <div class='viajes-item dos'>
        <h2>Polonia, un país por descubrir</h2>
        <p>Dignissim placerat vel aenean porta, magna ac scelerisque facilisis.</p>
        <img src='images/oporto.jpg' alt='' />
        <button>MÁS INFO</button>
    </div>
    <div class='viajes-item tres'>
        <h2>Un paseo por el viejo Oporto</h2>
        <p>Cras quis mattis. Elementum, lectus magna, amet dis dis pulvinar scelerisque proin mon
        <img src='images/polonia.jpg' alt='' />
        <button>MÁS INFO</button>
    </div>
</section>
```

He puesto una clase a cada viaje para numerarlos en el HTML. Su función es simplemente jugar con ellos y ver qué fácil es cambiarlos de orden con FlexBox

Los contenedores flexibles: flex e inline-flex

```
#viajes {
    display: flex;
}
```

el contenedor (**#viajes**) será un contenedor flexible y los elementos dentro del contenedor serán elementos flexibles que se adaptarán a los espacios disponibles que les deje el contenedor. Podríamos haber puesto también:

```
#viajes {
    display: inline-flex;
}
```

en cuyo caso se comportaría en relación a otros elementos de la página de manera semejante a **display: inline**. En caso contrario se comporta igual que **display: block**.

Orientación: flex-flow, flex-direction y flex-wrap

nos va a permitir decirle al navegador cómo se van a alinear los elementos que están dentro del contenedor (#viajes).

flex-flow = flex-direction + flex-wrap

flex-flow: row wrap. De esta manera establecemos que los elementos dentro del contenedor (los .viajes-item) se van a alinear en fila (row). flex-flow lo podríamos haber dividido en dos: **flex-direction** y **flex-wrap**

```
#viajes {
    display: flex;
    flex-flow: row wrap;
}
```

→

```
#viajes {
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
}
```

Hemos visto antes que el eje principal es la dirección principal, pero hay que tener en cuenta que esta no es siempre horizontal. La dirección del contenedor se puede cambiar con la propiedad **flex-direction**, que *especifica cómo se sitúan los elementos flexibles dentro del contenedor*. Sus posibles valores son:

- **row**: se alinean en filas.
- **row-reverse**: en filas, pero con el orden inverso.
- **column**: se alinean en columnas.
- **column-reverse**: en columnas, pero con el orden inverso.

Es decir, que en móviles lo más lógico para nuestro ejemplo es tener una lista de una columna en lugar de varias columnas en horizontal. Esto se consigue en el ejemplo con flex-direction: column;.
En tamaños grandes se usaría flex-direction: row; para lograr que se distribuya horizontalmente. El valor por defecto de flex-direction es row, por lo que si no usáramos flex-direction los elementos aparecerían distribuidos horizontalmente.

para probar **flex-direction:column** vamos a cambiar la orientación cuando el navegador tenga un ancho de 500px o menor. Así lo veremos mejor en los móviles:

```
@media all and (max-width: 500px) {
    #viajes {
        flex-direction: column;
    }
}
```

→ ***Para todos los dispositivos y un ancho máximo de 500 px***

Probad a reducir el ancho del navegador y veréis el cambio al llegar a los 500px de ancho.

La propiedad **flex-wrap** controla si el contenedor flexible (en este caso #viajes) tiene una sola línea o múltiples líneas, así c

- **nowrap**: El contenedor consta de una sola línea.
- **wrap**: El contenedor tiene múltiples líneas.
- **wrap-reverse**: El contenedor tiene múltiples líneas que se colocan en orden inverso.

En este caso, al ser múltiples líneas, hemos puesto wrap.

Ahora vamos a ver las propiedades que podemos asignar a los elementos que están dentro del contenedor (los .viaje

```
.viajes-item {
    display: flex;
    flex-direction: column;
}
```

Orden: La propiedad order

Podemos establecer el orden en el que aparecen los componentes de una caja flexible. Por defecto aparecerán tal y como aparecen en el código HTML (equivale a order: 0), pero eso se puede cambiar

```
.uno {
    background-color: #bacee3;
    order: 3;
}
.dos{
    background-color: #b6ebb3;
    order: 2;
}
.tres {
    background-color: #7ebdbb;
    order: 1;
}
```

Vamos a aprovechar el **media-query** que iniciamos antes para volver a poner todo en su orden original (order: 0 para los .viajes-item) para anchos inferiores a 500px:

```
@media all and (max-width: 500px) {
   #viajes {
      flex-direction: column;
      }
   }
   .viajes-item {
      order: 0;
      width: auto;
   }
```

Dentro de cada viaje queremos poner la imagen encima de todo. ¿Cómo lo conseguimos? Como por defecto todos tienen order: 0 si ponemos a la imagen order: -1 se pondrá la primera.

```
.viajes-item > img {
    order: -1;
}
```

Flexibilidad: La propiedad flex

Con la propiedad flex podemos establecer cómo crece o decrece un elemento flexible dentro del contenedor en relación a los demás. En el ejemplo que estamos siguiendo las tres columnas tienen el mismo ancho, por ello veréis en el código que todos tienen flex:1:

```
.uno {
    background-color: #bacee3;
        flex: 1;
    }
.dos{
    background-color: #b6ebb3;
        flex: 1;
    }
.tres {
    background-color: #7ebdbb;
    flex: 1;
    }
```

Si hubiéramos querido que .uno ocupara el doble que los otros dos habríamos hecho:

```
.uno {
        background-color: #bacee3;
        flex: 2;
    }
```

Esta propiedad se puede volver más compleja, porque puede tener tres parámetros: **flex-grow**, **flex-shrink** y **flex-basis**.

- **flex-grow**: Especifica el factor de crecimiento, es decir, cuanto crecerá el elemento en relación a los demás cuando hay espacio disponible del contenedor a ocupar. Por defecto es ‘0’, que es el valor que dimos en el ejemplo anterior a los tres elementos.
- **flex-shrink**: Determina el factor de reducción, es decir, cuanto decrecerá el elemento en relación a los demás cuando hay espacio negativo en el contenedor (el contenedor es más pequeño de los anchos combinados de los elementos que hay en su interior). Por defecto es ‘1’.
- **flex-basis**: Toma el mismo valor que la propiedad ‘width’ y establece el tamaño inicial del elemento antes de distribuir el espacio libre de acuerdo con los ratios de flex-grow o flex-shrink. Cuando se omite, su valor es ‘main-size’ (anteriormente, ‘auto’).

Supongamos que tenemos un contenedor al que llamaremos “A” que tiene 300px de ancho. Hacemos que este contenedor sea flexible: **A display**

Supongamos que este contenedor tiene en su interior dos elementos, B y C, que no tienen un ancho especificado. Vamos a establecer los siguientes valores para ambos de la propiedad flex: **flex-grow**, **flex-shrink**, **flex-basis**:

```
B { flex: 3 1 100px;}
C { flex: 1 2 100px;}
```

Como hemos establecido un flex-basis para cada elemento de 100px, nos quedarán aún 100px libres sin ocupar (300px del contenedor menos 100px del elemento B y menos 100px del elemento C). ¿Como se reparte ese espacio disponible entre los elementos B y C? En función del flex-grow: 3 partes para el elemento B (75px) y una parte para el elemento C (25px). Es decir, de inicio el elemento B ocupará 100px+75px = 175px de los 300px disponibles que mide el elemento A, y el elemento C ocupará 100px+25px=125px de los 300px disponibles del elemento A.

supongamos que el elemento contenedor A mide 170px y no 300px. Eso quiere decir que habrá espacio negativo, porque los elementos B y C tienen un flex-basis de 100px cada uno, es decir, 200px, que es 30px mayor que los 170px del contenedor. En este caso el ratio que se usa es el flex-shrink, que recordemos que era 1 para el elemento B y 2 para el elemento C. Esos 30px se restarán del ancho de los elementos B y C en función de dicho ratio: al elemento B se le quitarán 10px y al elemento C se le quitarán 20px.

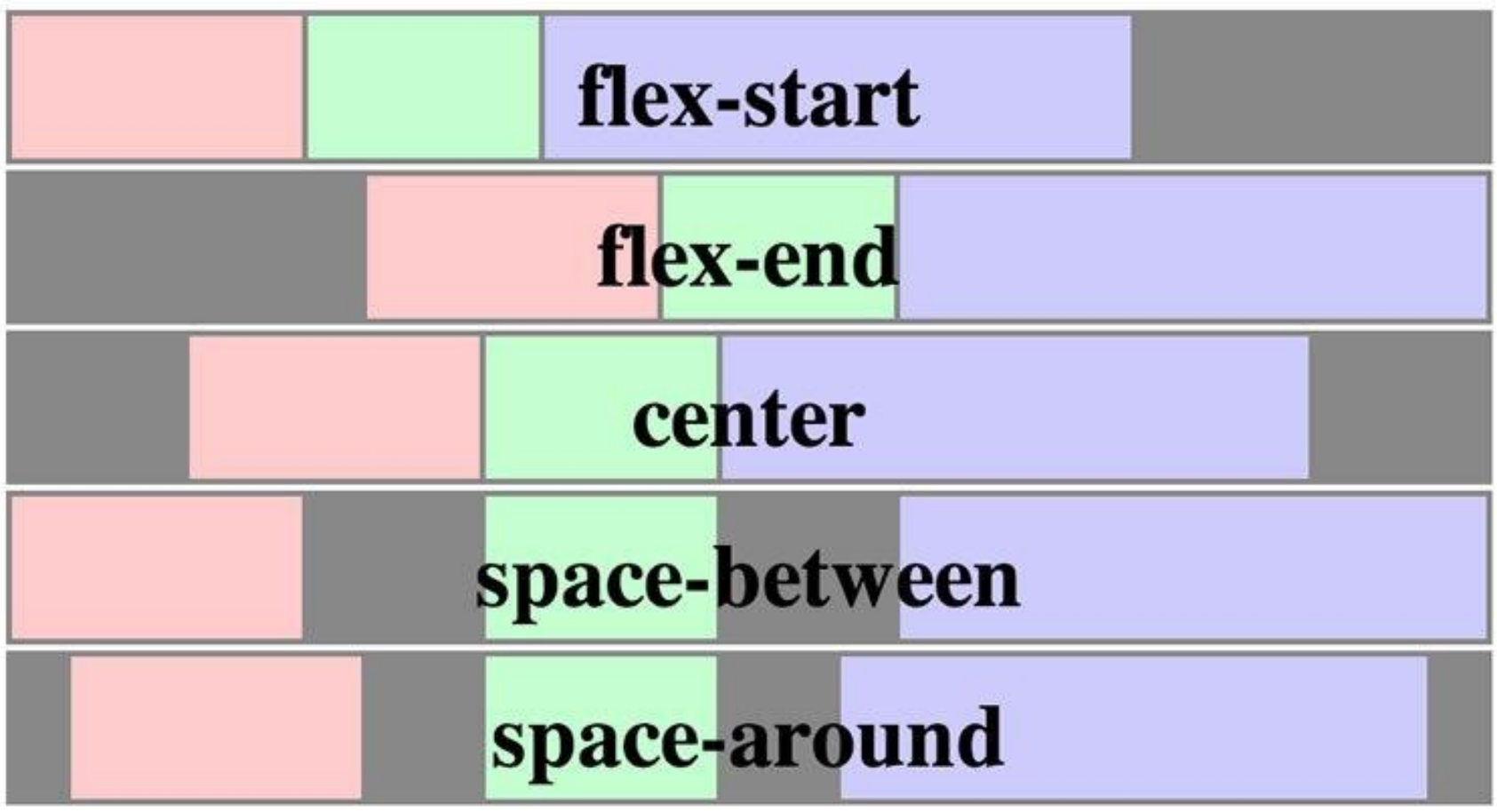
Alineación de los elementos flexibles

Podemos alinear los elementos flexibles en:

- el Eje principal con justify-content
- el Eje transversal con align-items y align-self.

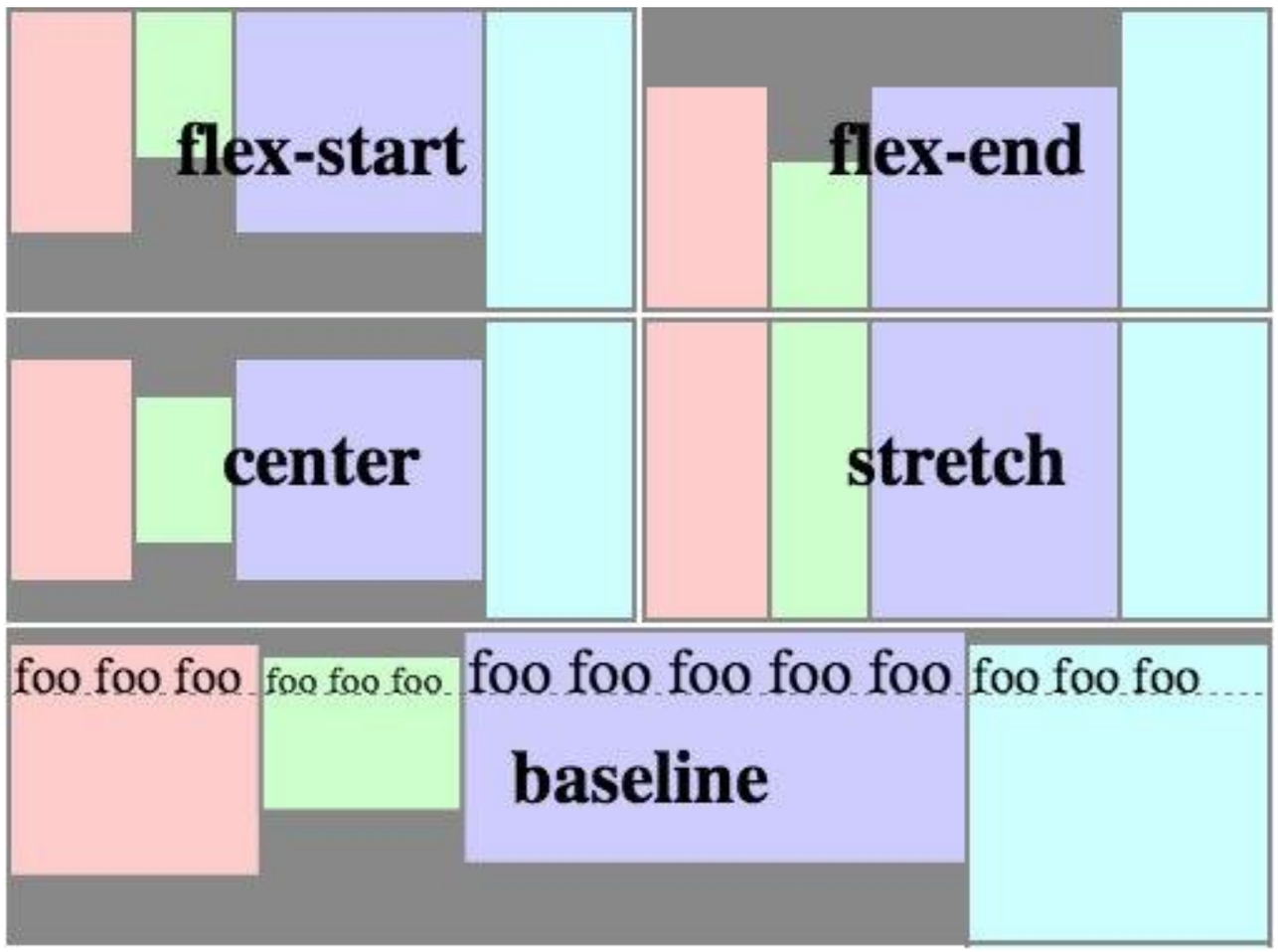
Si hay espacio extra dentro de un contenedor flexible la propiedad justify-content puede definir cómo se usa ese espacio que sobra. Las opciones son:

- **flex-start:** se distribuyen todos pegados al inicio.
- **flex-end:** se distribuyen todos pegados al final.
- **center:** se distribuyen todos alineados al centro.
- **space-between:** se distribuyen ocupando todo el espacio disponible, con separaciones iguales entre ellos, pero sin dejar espacio al inicio y al final.
- **space-around:** se distribuyen ocupando todo el espacio disponible, con separaciones iguales entre ellos, dejando espacio al inicio y al final.



Podemos alinear los elementos flexibles en el eje transversal con align-items y con align-self. La primera establece el valor por defecto para todos, la segunda sirve para ser aplicada a elementos individuales sobre-escribiendo align-items para ese elemento.
Los valores posibles son:

- **auto:** Sólo se puede aplicar en align-self y equivale al valor de align-items del elemento padre, o a stretch si el elemento no tiene padre.
- flex-start
- flex-end
- center
- baseline
- stretch



En el ejemplo inicial de nuestra agencia de viajes, queríamos que el botón de “Más info” estuviera alineado en la parte inferior de cada caja flexible. Como todas tienen la misma altura, los tres botones estarán igualmente a la misma altura.

Eso lo conseguimos aplicando a la imagen margin-top: auto

```
.viajes-item > button {
    margin-top: auto;
    width: 30%;
}
```

Además, queremos que tanto la imagen como el botón estén centrados dentro de su contenedor. Para ello utilizamos **align-self:**

```
.viajes-item > img {
    order: -1;
    align-self: center;
}
.viajes-item > button {
    margin-top: auto;
    width: 30%;
    align-self: center;
}
```

Ojo: Hay propiedades de CSS que no funcionarán en un elemento que se encuentre dentro de un contenedor flexible, como float, clear, column- o vertical-align.