Flexbox CSS: Guia Completo, Elementos y Ejemplos

J

juliana-amoasei

02/09/2021

Flexbox tiene como meta ser una forma más eficiente de crear diseños, alinear y distribuir espacios entre ítems en un contenedor, incluso cuando las dimensiones de esos ítems son desconocidas y/o dinámicas (en virtud de eso el término "flex").

Aprendamos los fundamentos de CSS Flexbox para la alineación y el posicionamiento, y cómo usar sus funcionalidades correctamente.

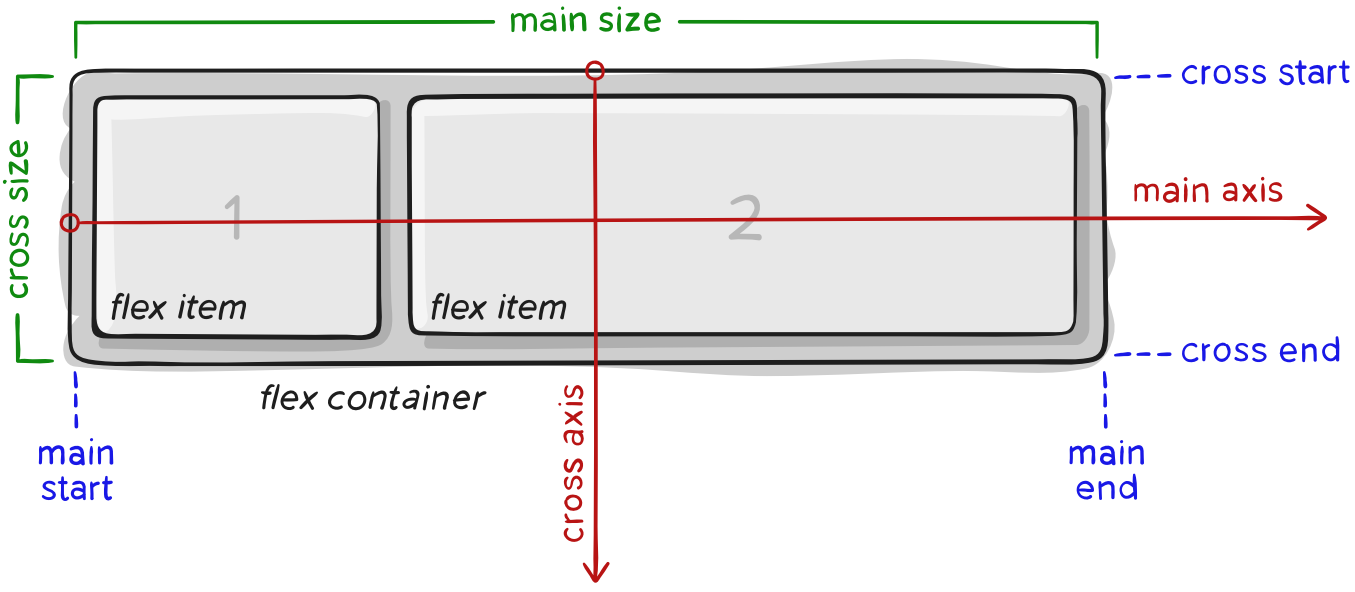
Qué es el *Flexbox*

Durante mucho tiempo, las únicas herramientas disponibles para crear diseños CSS y posicionar elementos con buena compatibilidad entre browsers eran float y position. Sin embargo, estas herramientas tienen algunas limitaciones muy frustrantes, especialmente cuando se trata de responsividad. Algunas tareas que consideramos básicas en un diseño, como centrar verticalmente un elemento secundario en relación con un elemento principal o hacer que los elementos secundarios ocupen la misma cantidad de espacio, o que las columnas tengan el

mismo tamaño independientemente de la cantidad de contenido interno, fueron imposibles. o muy difícil de manejar con floats o position, al menos de forma práctica y *flexible*. La herramienta Flexbox (de *Flexible Box*) fue creada para hacer estas tareas más sencillas y funcionales: los secundarios de un elemento con Flexbox se pueden posicionar en cualquier dirección y pueden tener dimensiones flexibles para adaptarse.

Elementos

Flexbox es un módulo completo y no una propiedad única; algunos de ellos deben declararse en el contenedor (el elemento principal, que llamamos de *flex container*), mientras que otros deben declararse en los elementos secundarios (el *flex ítems*). Si el diseño "estándar" se basa en las direcciones `block` e `inline`, el diseño Flex se basa en direcciones de "flex flow". A continuación se muestra un diagrama de la especificación, que explica la idea central detrás del diseño Flex.



Los ítems se distribuirán en el diseño siguiendo el eje principal o transversal.

- Eje principal: el eje principal de un *flex container* es el eje primario y a lo largo de él son insertados los *flex ítems*. **Precaución**: El eje principal no es

necesariamente horizontal; Dependerá de la propiedad `flex-direction` (vea abajo).

- *Main-start | main-end*: los *flex ítems* se insertan en el contenedor empezando por el lado *start*, dirigiéndose hacia el lado *end*.

- Tamaño principal: El ancho o alto de un *flex ítem*, dependiendo de la dirección del contenedor, es el tamaño principal del ítem. La propiedad de tamaño principal de un *flex ítem* puede ser tanto `width` como `height`, dependiendo de cuál esté en la dirección principal.

- Eje transversal: El eje perpendicular al eje principal se llama eje transversal. Su dirección depende de la dirección del eje principal.

- *Cross-start | cross-end*: Líneas flex se llenan con ítems y se agregan al contenedor, comenzando desde el lado *cross start* del *flex container* hacia el lado *cross end*.

- *Cross size*: El ancho o alto de un *flex ítem*, dependiendo de lo que haya en la dimensión transversal, es el *cross size* del ítem. La propiedad *cross size* puede ser el ancho o el alto del ítem, lo que se encuentre en la transversal.

  Flex container es el elemento que involucra su estructura. Tu defines que un elemento es un Flex Container con la propiedad `display` y valores `flex` o `inline-flex`.

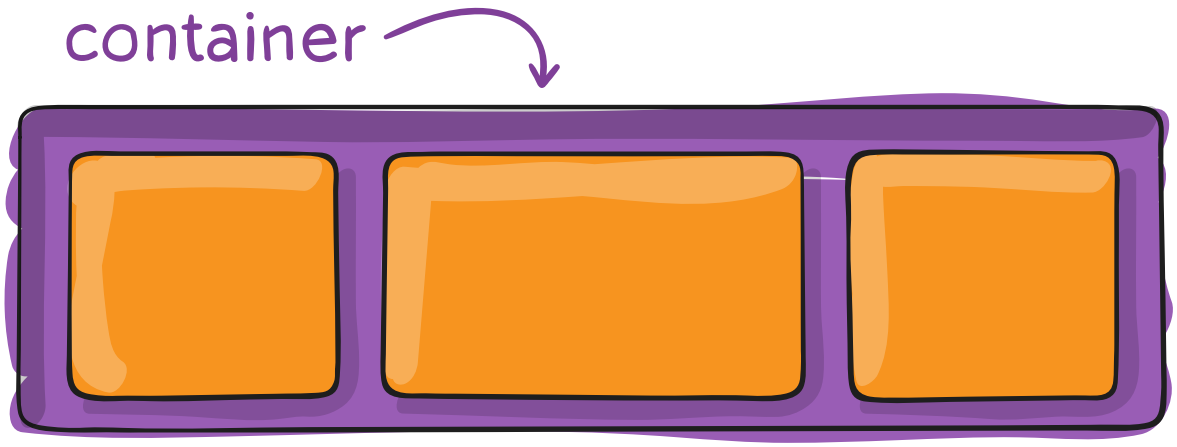
  ```
  <div class="flex-container">

  <div>1</div>

  <div>2</div>

  <div>3</div>

  </div>

  .flex-container {
  ```

```
display: flex;

}
```

Flex Ítem son elementos secundarios del flex container.

Ejes o Axes son las dos direcciones básicas que existen en un Flex Container: *main axis*, o eje principal, y *cross axis*, o eje transversal.

Propiedades del elemento principal



Cuando usamos el *Flexbox*, es muy importante saber qué propiedades se declaran en el elemento principal (por ejemplo, una div que contendrá los elementos a alinear) y cuáles se declararán en los elementos secundarios. A continuación, se muestran las propiedades que deben declararse utilizando el elemento principal como selector (para alinear los elementos secundarios):

Display

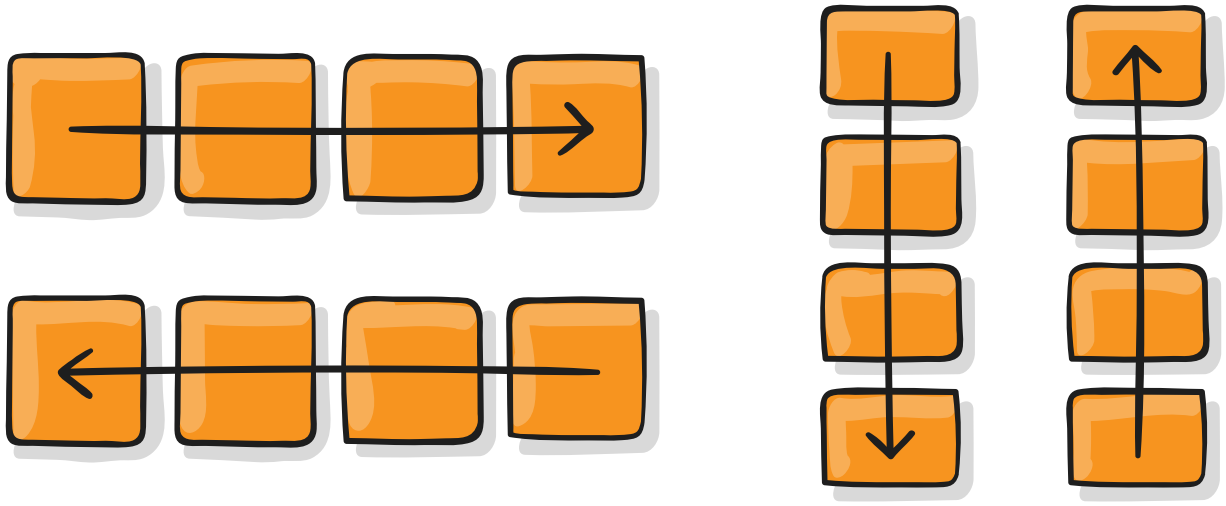
Esta propiedad define un *flex container*; inline o block dependiendo de los valores pasados. Coloca todos los elementos secundarios directos en un contexto Flex.

```
.container {
```

```
display: flex; */\* or inline-flex \*/*

}
```

Tenga en cuenta que la propiedad CSS `columns` no tiene efecto en un *flex container*.

Flex-direction



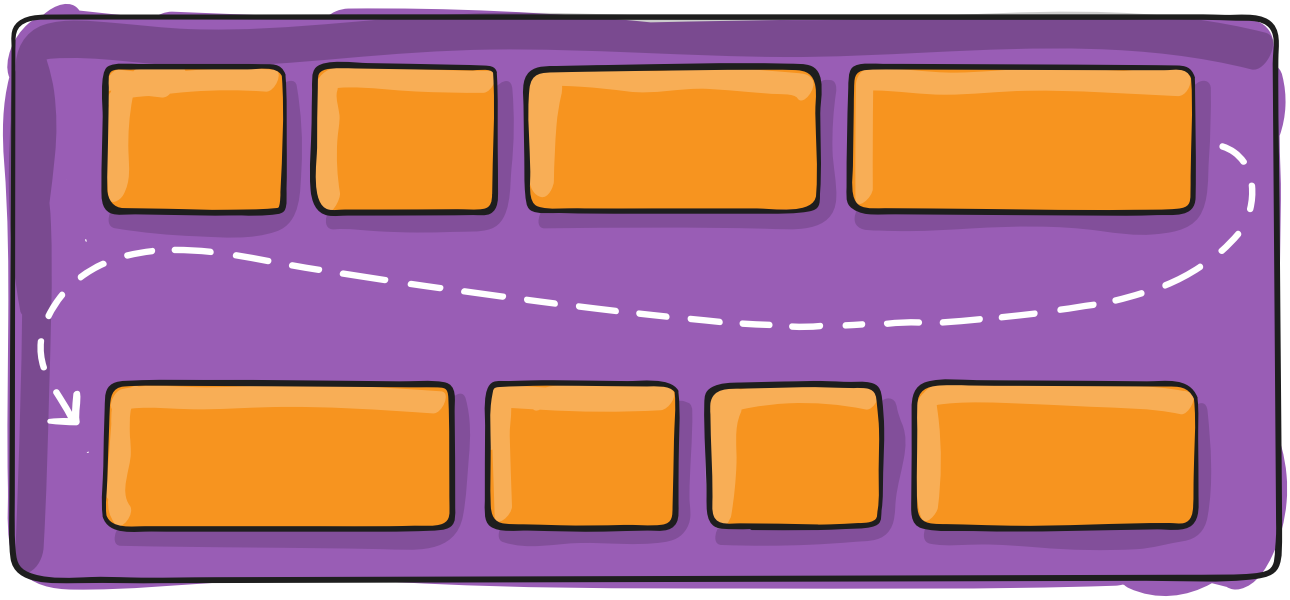
Establece el eje principal, definiendo así la dirección en la que los *flex ítems* están alineados en el *flex container*. Flexbox es (con la excepción de un wrapping opcional) un concepto de diseño unidireccional. Piensa en los *flex ítems* inicialmente posicionaids o en líneas horizontales o en columnas verticales.

```
.flex-container { flex-direction: row | row-reverse | column | column-reverse; }
```

- `row` (estándar): de la izquierda a la derecha en `ltr` (left to right), de la derecha a la izquierda en rtl (right to left)
- `row-reverse`: de la derecha a la izquierda en `ltr`, de la izquierda a la derecha en `rtl`
- `column`: mismo que `row`, pero de arriba a abajo

- `column-reverse`: mismo que `row-reverse` pero de abajo hacia arriba

flex-wrap



Por estándar, los *flex ítems* todos intentarán encajarse en una sola línea. Con esta propiedad puedes modificar este comportamiento y permitir que los ítems pasen a la siguiente línea según sea necesario.

```
.flex-container {

flex-wrap: nowrap | wrap | wrap-reverse;

}
```

- `nowrap` (estándar): todos los *flex ítems* estarán en una sola línea
- `wrap`: los *flex ítems* se dividirán en múltiples líneas, de arriba a abajo
- `wrap-reverse`: los *flex ítems* se dividirán en múltiples líneas de abajo hacia arriba

flex-flow

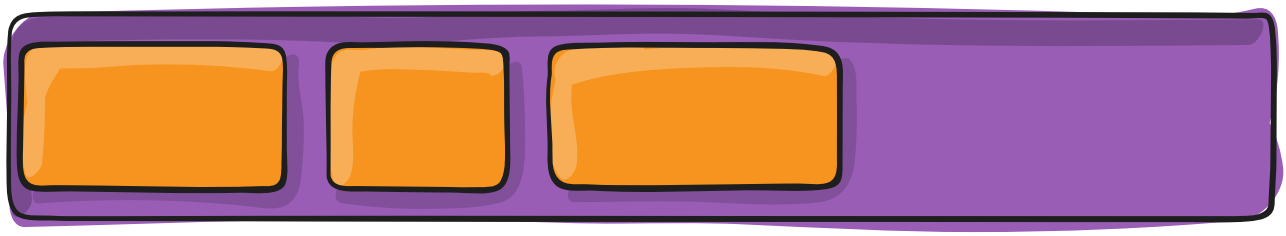
- La propiedad **flex-flow** es una propiedad *shorthand* (una misma declaración incluye varios valores relacionados con más de una

propiedad) que incluye flex-direction y flex-wrap. Determina cuáles serán los ejes principal y transversal del contenedor. El valor estándar es row nowrap.

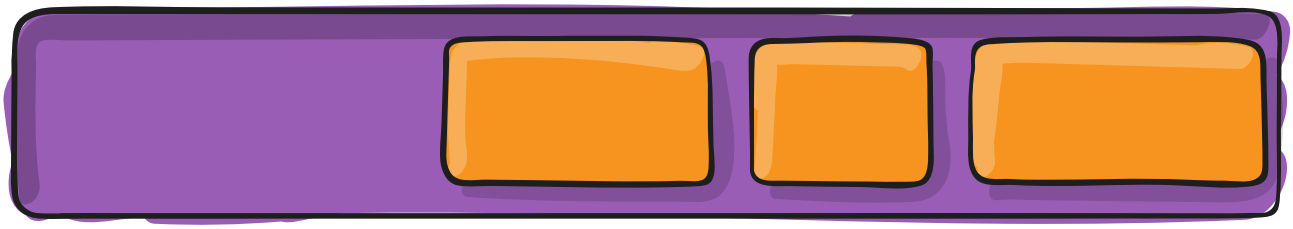
```
.flex-container { flex-flow: row nowrap | row wrap | column nowrap | column wrap; }
```

Justify-content

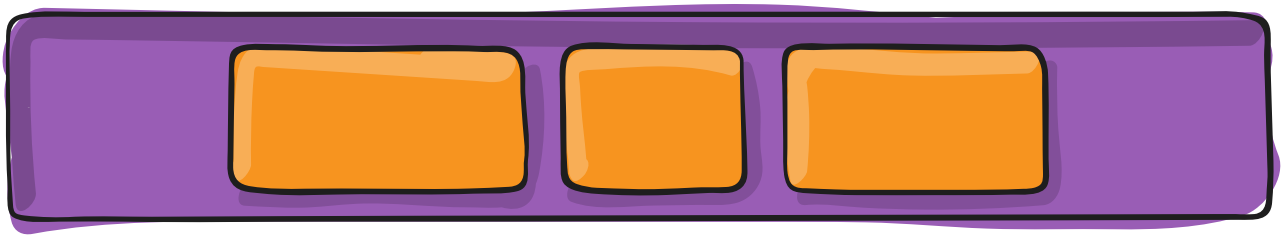
flex-start



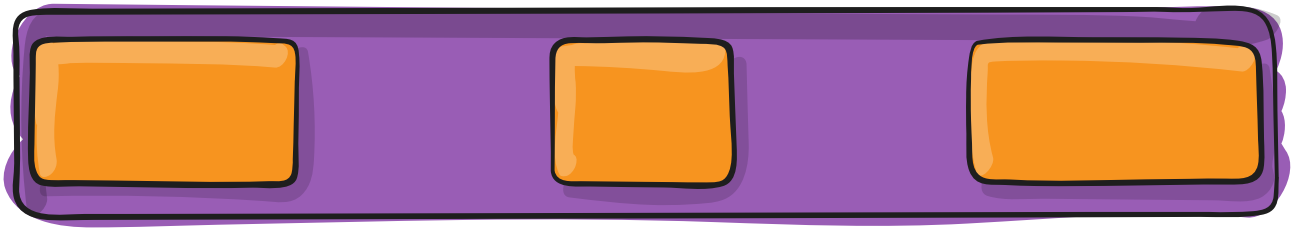
flex-end



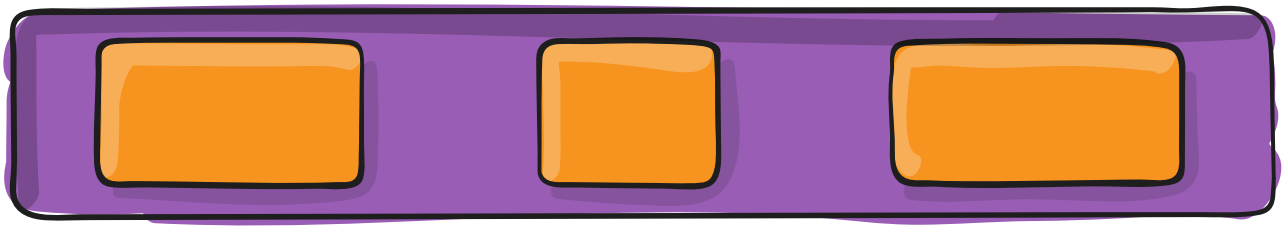
center



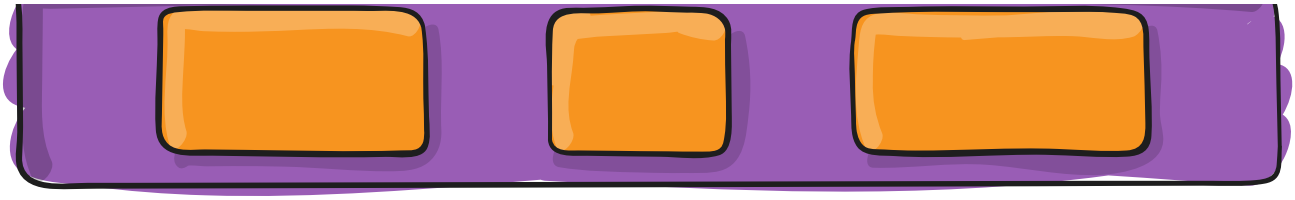
space-between



space-around



space-evenly



Esta propiedad define la alineación de los ítems a lo largo del eje principal. Ayuda a distribuir el espacio libre que queda en el contenedor, ya sea que todos los flex ítems de una línea sean inflexibles o flexibles, pero ya hayan alcanzado su tamaño máximo. También ejerce cierto control sobre la alineación de los ítems cuando sobrepasan el límite de la línea.

```
.flex-container {

justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly;

}
```

- `flex-start` (estándar): los ítems se alinean a lo largo del borde de inicio (start) de acuerdo con la `flex-direction` del contenedor.
- `Flex-end`: los ítems se alinean a lo largo del borde final (end) de acuerdo con la `flex-direction` del contenedor.
- `start`: los ítems se alinean a lo largo del borde de inicio de la dirección del writing-mode (modo de escritura).
- `end`: los ítems se alinean a lo largo del borde final de la dirección del writing-mode (modo de escritura).
- `left`: los ítems están alineados a lo largo del borde izquierdo del contenedor, a menos que esto no tenga sentido con el `flex-direction` que se está utilizando. En este caso, se comporta como `start`.
- `right`: los ítems están alineados a lo largo del borde derecho del contenedor, a menos que esto no tenga sentido con el `flex-direction`

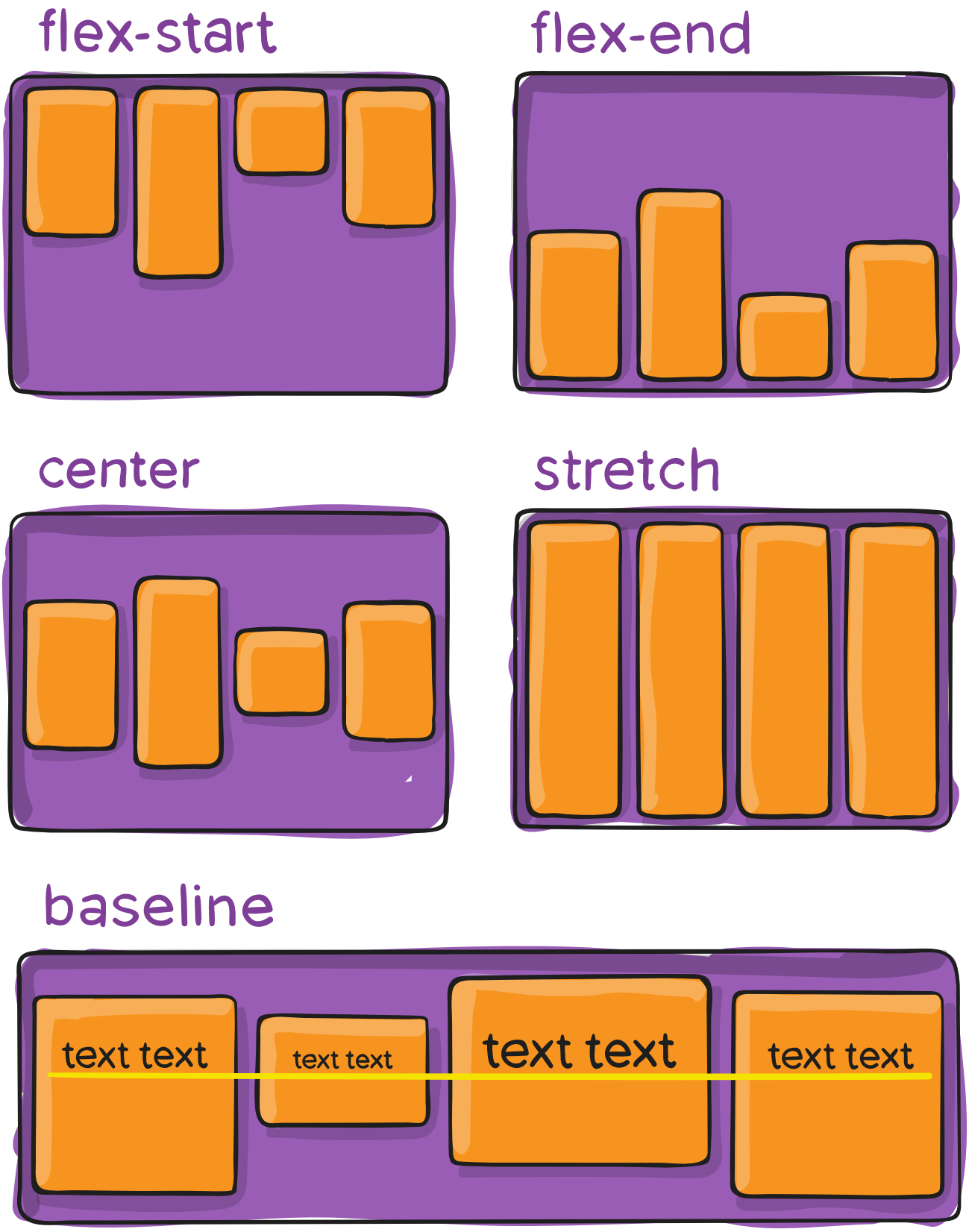
que se está utilizando. En este caso, se comporta como `start`.

- `center:` los ítems están centrados en la línea.
- `Space-between:` los ítems se distribuyen uniformemente a lo largo de la línea; el primer ítem al lado del borde inicial de la línea, el último ítem al lado del borde final de la línea.
- `Space-around:` los ítems se distribuyen en línea con el mismo espacio entre ellos. Tenga en cuenta que visualmente el espacio puede no ser igual, ya que todos los ítems tienen la misma cantidad de espacio en ambos lados: el primer ítem solo tendrá una unidad de espacio a lo largo del borde del contenedor, pero dos unidades de espacio entre él y el siguiente ítem, porque el ítem siguiente también tiene su propio espaciamiento que se está aplicando.
- `Space-evenly:` los ítems se distribuyen de manera que el espacio entre dos elementos cualesquiera en la línea (incluso entre los ítems y los bordes) sea igual.

Nota: el soporte dado por los navegadores para estos valores es difuso. Por ejemplo, space-between no tiene soporte en ninguna versión de Edge (hasta la elaboración de este tutorial) y start / end / left / right aún no se han implementado en Chrome. Para obtener tablas detalladas, consulte MDN. Los valores más seguros son flex-start, flex-end y center.

También hay dos palabras clave adicionales que puedes usar junto con estos valores: `safe y unsafe`. Safe asegura que, independiente de cómo hagas este tipo de posicionamiento, no sea posible "empujar" un elemento y hacer con que sea renderizado hacia afuera de la pantalla (por ejemplo, sobre el tope) de una manera que haga con que el contenido sea imposible de mover con el desplazamiento de la pantalla (el CSS llama a esto de "pérdida de datos").

Align-ítems



Establece el comportamiento estándar de cómo *flex ítems* están alineados según el eje transversal (*cross axis*). En cierto modo, funciona de manera

similar al justify-content, pero en el eje transversal (perpendicular al eje principal).

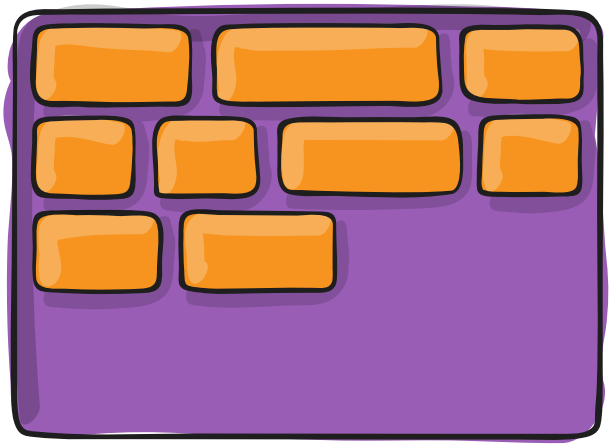
```
. .flex-container {

align-items: stretch | flex-start | flex-end | center | baseline;

}
```

- `Stretch` (estándar): estira los ítems para llenar el contenedor, respetando el `min-width/max-width`).
- `Flex-start/start/self-start:` ítems se posicionan al inicio del eje transversal. La diferencia entre ellos es sutil y se refiere a las reglas de `flex-direction` o `writing-mode`.
- `center`: ítems se centran en el eje transversal.
- `baseline:` ítems se alinean de acuerdo con sus baselines.

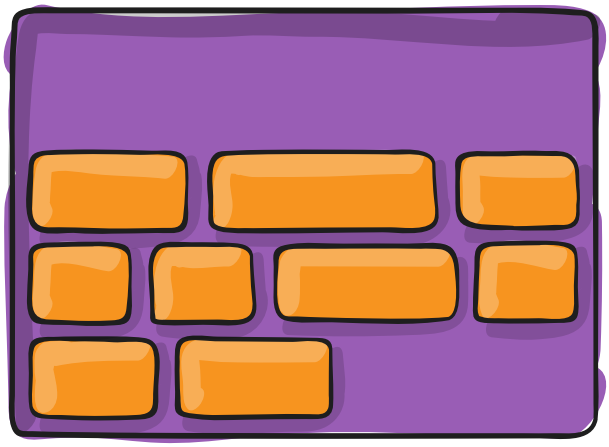
Los modificadores `safe y unsafe` se puede usar junto con todas estas palabras clave (por favor verifique el soporte de cada navegador) y sirven para evitar cualquier alineación de elementos que haga con que el contenido sea inaccesible (por ejemplo, fuera de la pantalla).

align-content

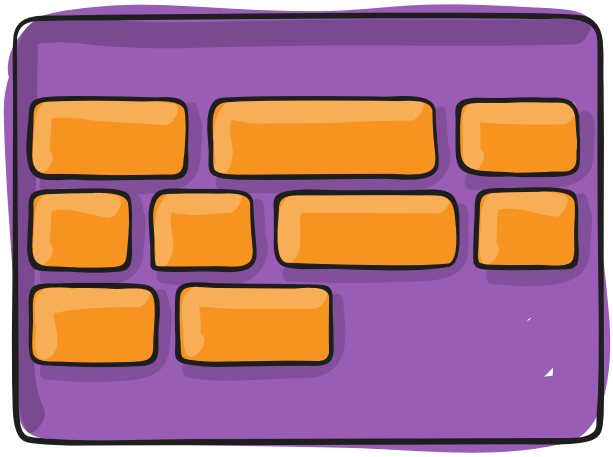
flex-start



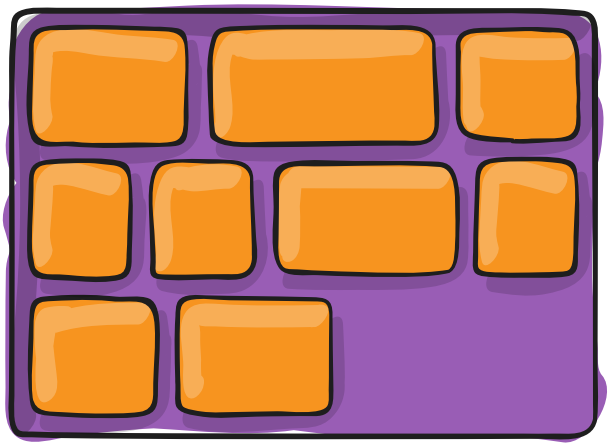
flex-end



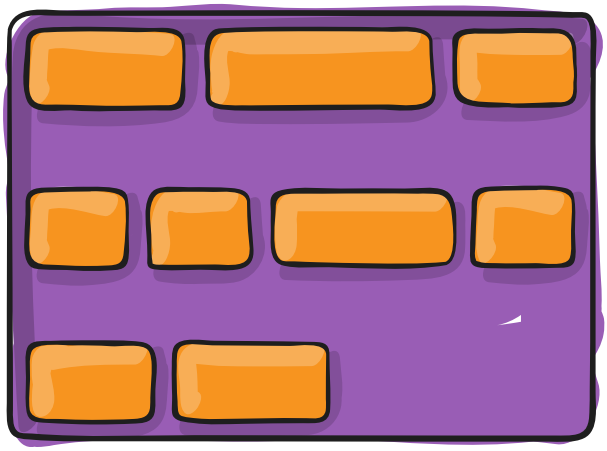
center



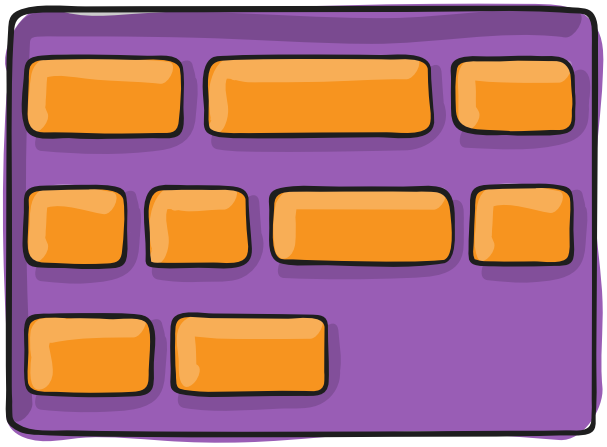
stretch



space-between



space-around



Organiza las líneas dentro de un flex container cuando hay espacio adicional en el eje transversal, similar a cómo `justify-content` alinea ítems

individuales dentro del eje principal.

Importante: Esta propiedad no tiene efecto cuando solo hay una línea de flex ítems en el contenedor.

```
.flex-container {

align-content: flex-start | flex-end | center | space-between | space-around | stretch;

}
```

- `flex-start / start:` ítems alineados con el inicio del contenedor. El valor (con mayor soporte de los navegadores) flex-start se guía por la flex-direction, mientras que start se guía por la dirección del writing-mode.
- `Flex-end/end:` ítems alineados con el final del contenedor. El valor (con mayor soporte de los navegadores) flex-end se guía por la flex-direction, mientras que end se guía por la dirección del writing-mode.
- `center`: ítems centrados en el contenedor.
- `space-between`: ítems distribuidos uniformemente; la primera línea al inicio del contenedor y la última línea al final del contenedor.
- `space-around`: ítems distribuidos uniformemente con el mismo espaciamiento entre cada línea.
- `space-evenly:` elementos distribuidos uniformemente con el mismo espaciamiento entre ellos.
- `stretch` (estándar): ítems en cada línea se estiran para ocupar el espacio remanente entre ellas.

Los modificadores `safe y unsafe` se puede usar junto con todas estas palabras clave (por favor verifique el soporte de cada navegador) y sirven

para evitar cualquier alineación de elementos que haga con que el contenido sea inaccesible (por ejemplo, fuera de la pantalla).

Propiedades de los elementos secundarios

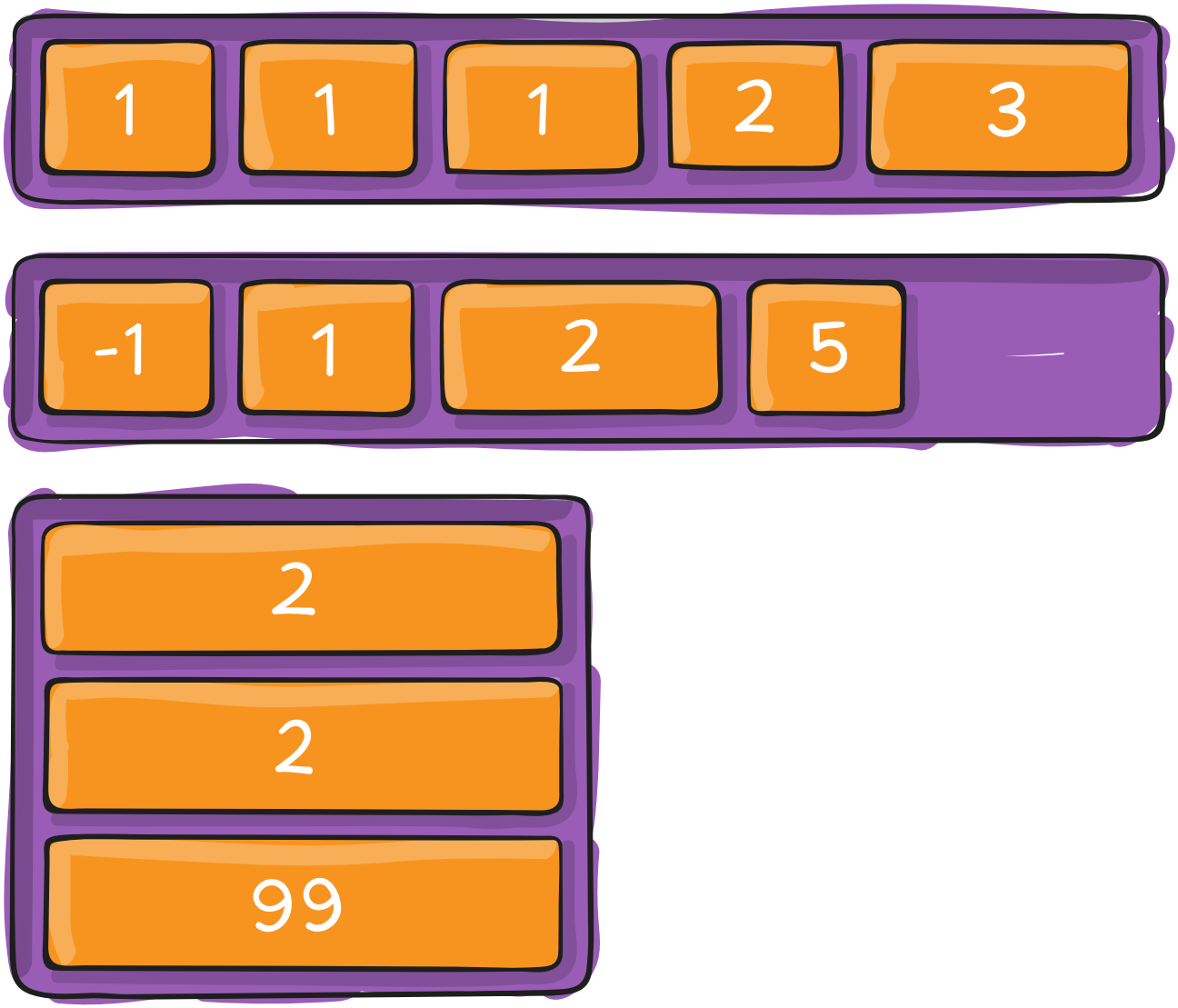
A continuación, veremos propiedades que deben declararse teniendo como selector los elementos secundarios, es decir:

1
2
3
Esto significa que, donde hay un elemento principal con propiedad *flex* (o *flex-container*), también puede asignar propiedades flex específicas a elementos secundarios (*flex-ítem*).

Puede definir las propiedades a continuación para solo uno de los elementos secundarios a través de un identificador, como una clase específica.

order

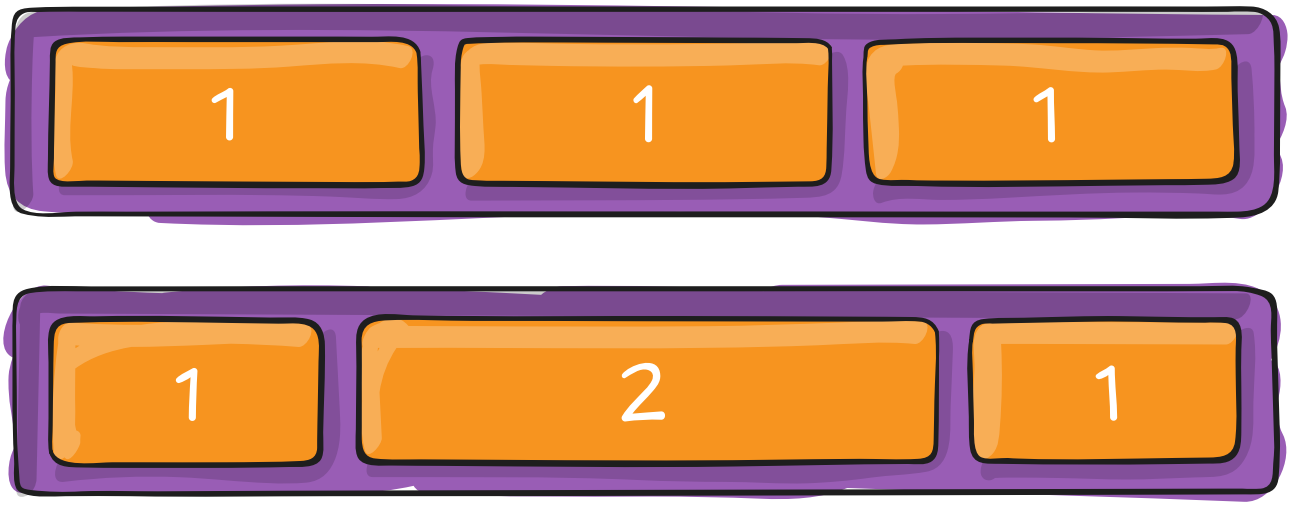
Determina el orden en que aparecerán los elementos.



Por estándar los flex ítems, se organizan en la pantalla en orden de código. Pero la propiedad order controla el orden en que aparecen en el contenedor.

```
.flex-item {

order: <número>; */\* el valor estándar es 0 \*/*

}
```

Flex-grow



Define la habilidad de un flex ítem de crecer, según el caso. El valor de esta propiedad es un valor numérico sin indicación de unidad, que se utiliza para calcular la proporción. Este valor dicta la cantidad de espacio disponible en el contenedor que ocupará el ítem.

Si todos los ítems tienen flex-grow establecido en 1, el espacio remanente en el contenedor se distribuirá de manera uniforme entre todos. Si uno de los ítems tiene el valor 2, ocupará el doble de espacio en el contenedor que los demás (o al menos intentará hacerlo).

```
.flex-item {

flex-grow: <numero>; */\* el valor default(estándar) es 0 \*/*

}
```

La propiedad no acepta valores negativos.

flex-shrink

Establece la habilidad de un flex ítem para contraerse, según el caso. `.flex-item {`

`flex-shrink: <número>; */\* el valor estándar es 0 \*/* }`

La propiedad no acepta valores negativos.

flex-basis

Establece el tamaño estándar de un elemento antes de que se distribuya el espacio remanente del contenedor. Puede ser un largo (por ejemplo, 20%, 5rem, etc.) o una palabra clave. La palabra clave auto significa "observe mis propiedades de altura o ancho" (lo que era hecho por la palabra clave `t main-size`, que fue depreciada). La palabra clave `content` significa "establezca el tamaño según el contenido interno del ítem"; esta palabra clave aún no tiene mucho soporte, por lo que no es fácil de probar, al igual que sus relacionadas: `max-content, min-content y fit-content.`

```
.flex-item {

flex-basis: flex-basis: | auto; */\* el valor estándar es auto \*/*

}
```

Con el valor 0, el espacio adicional alrededor del contenido no se considera. Con el valor de auto, el espacio adicional se distribuye con base en el valor de `flex-grox` del ítem.

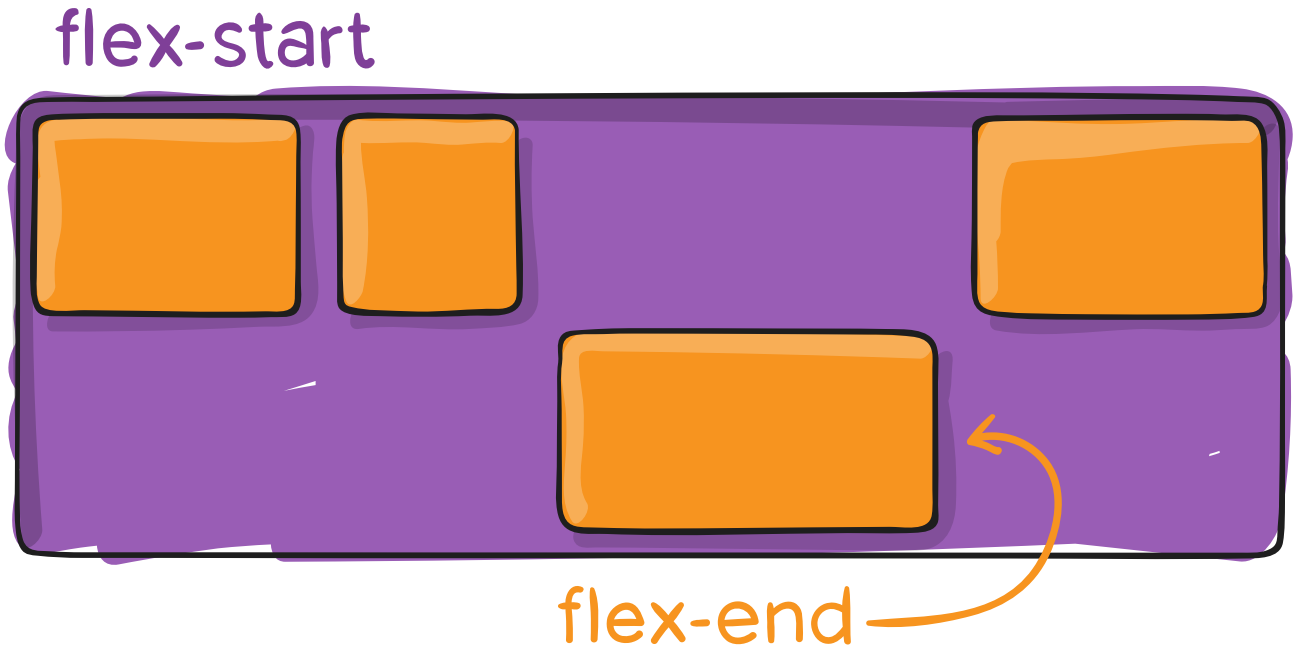
flex

Esta es la propiedad *shorthand* para `flex-grow, flex-shrink y flex-basis`, combinadas. El segundo y tercer parámetros (`flex-shrink y flex-basis`) son opcionales. El estándar es `0 1 auto`, pero si lo defines con un solo número, es equivalente a `0 1`.

```
.item {

flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]

}
```

Se recomienda que utilices esta propiedad *shorthand* en lugar de definir cada una de las propiedades por separado. *Shorthand* establece los demás valores de forma inteligente.

align-self



Permite que la alineación estándar (o lo que esté definido por align-items) se sobrescriba para ítems individuales.

Por favor consulte la explicación de la propiedad align-items para entender cuáles son los posibles valores.

```
. .item {

align-self: auto | flex-start | flex-end | center | baseline |
stretch;

}
```

¡Importante!

- CSS solo ve la jerarquía de principal-secundario; no aplicará propiedades Flex a elementos que no estén directamente relacionados;
- Para que las propiedades funcionen en elementos secundarios, los principales deben tener propiedad `display: flex;`.
- Las propiedades `float, clear y vertical-align` no tienen ningún efecto en flex-items.

¿Vamos a practicar?

Flexbox Froggy

Enlaces útiles

Al igual que con cualquier otra funcionalidad nueva que hemos aprendido, es fundamental practicar mucho e investigar siempre que tengamos dudas. A continuación se muestran algunos enlaces útiles.

Este contenido se basa en gran parte en el tutorial de CSS Tricks, ¡con nuestros agradecimientos! Guarda en tus favoritos para consultar siempre que lo necesites.

La guía CSS Tricks también aborda algunos otros aspectos importantes de Flex: soporte de los navegadores, errores, propiedades relacionadas, prefijos y etc. Si tiene alguna duda que no se haya abordado en este artículo, puedes consultar estos temas relacionados en el enlace.

- Flexbox en MDN
- Flexbox en W3Schools

¿Te gustó la publicación y quieres saber más? Aquí en **Alura** tenemos una **formación front-end** donde aprenderás más sobre **HTML y CSS.**

Juliana Amoasei