Capítulo 4 Flexbox

4.1 Introducción ¿Qué es flexbox?

Flexbox (o el módulo de cajas flexibles) es probablemente uno de los más completos y eficaces módulos de maquetación. Todo lo que era complicado en versiones anteriores de CSS (como centrar verticalmente o diseñar estructuras que se redimensionan con elegancia) con flexbox (CSS3) es ya una tarea muy fácil.

¿Cómo crear una caja flex?

Flexbox representa un modelo básico de maquetación que supone la existencia de una caja padre llamada contenedor flexible o caja flex. Los elementos hijos situados dentro del contenedor flexible llevan el nombre de elementos o **ítems** flex.

Los elementos flex tienen la capacidad de redimensionarse y colocarse dentro de la caja flex con facilidad. También tienen la capacidad de alinearse tanto horizontalmente como verticalmente y todo esto puede ser muy interesante a la hora de diseñar páginas web adaptativas.

La propiedad **display** está por ahí desde CSS1, pero es en el CSS3 cuando adquiere la capacidad de transformar una caja cualquiera en un contenedor flex. Para esto tiene que tomar una de estas valores: **flex o flex-inline.**

```
.flex-container{display:flex;} o
.flex-container{display:flex-inline;}
```

```
.flex-container{
        display: flex;
}
.flex-container-inline{
        display: inline-flex;
}
```

Propiedades del contenedor flex

4.2. La propiedad flex-direction

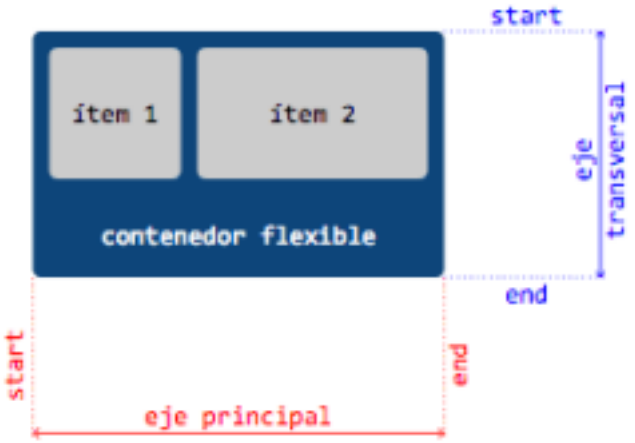
La propiedad **flex-direction** es una propiedad del contenedor flex.

En una caja flex podemos colocar los elementos en cualquier dirección: horizontalmente (**row**) o verticalmente (**column**), en el sentido lógico o en el sentido contrario (**-reverse**).

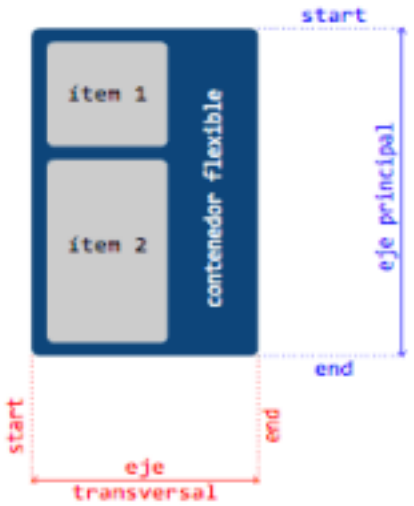
Para hacerlo, utilizamos la propiedad flex-direction, que establece cuál es el eje principal de la caja y por lo tanto la dirección de los elementos hijos.

2

flex-direction:row



flex-direction:column



Muy importante: flex-direction: row establece el eje horizontal como eje principal (*main axis*), y el eje vertical como eje transversal (*cross axis*). Si flex-direction: column pasa todo lo contrario: el eje vertical es el eje principal (*main axis*) mientras que el eje horizontal es el eje transversal (*cross axis*) .
Esto es realmente muy importante ya que las propiedades de flexbox controlan la alineación de los ítems flex a lo largo de estos ejes.

La propiedad flex-direction puede tomar una de estos valores:

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

row (el valor por defecto): coloca los elementos flex horizontalmente (*row = fila*). En idiomas como el castellano, con un sistema de escritura de izquierda a derecha (*ltr – left to right*), los elementos flex se colocan también de izquierda a derecha.

row-reverse: coloca los elementos flex horizontalmente pero en sentido contrario (de derecha a izquierda en idiomas como el castellano.)

column: coloca los elementos flex verticalmente y de arriba abajo (*tb - top to bottom*).

column-reverse: coloca los elementos flex verticalmente y de abajo arriba (*bottom to top*).

```
.flex-container.flex-row{

                flex-direction: row;
        }
.flex-container.row-reverse{
                flex-direction: row-reverse;
        }
.flex-container.flex-column{
                flex-direction: column;
                height: 150px;}
```

4.3 La propiedad flex-wrap

La propiedad flex-wrap es una propiedad del contenedor flex y especifica si puede haber un cambio de línea (wrap) o no (nowrap). El valor por defecto es nowrap. La propiedad flex-wrap puede tomar una de estas valores:

.contenedor { flex-wrap: nowrap | wrap | wrap-reverse; }

nowrap (el valor por defecto): los elementos flex aparecen en una sola línea. En ciertas circunstancias los elementos flex aparecen redimensionados para que puedan acomodarse dentro del contenedor flex. En otras circunstancias el contenedor flex puede desbordar (overflow).

wrap: indica al CSS que puede haber cambio de línea. Los elementos flex aparecen colocados en varias líneas.

wrap-reverse: igual que wrap pero las líneas de elementos flex aparecen ordenadas en sentido contrario.

flex-wrap: nowrap *(el valor por defecto)*

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

```
.flex-container{
        width: 250px;
        height: 140px;
        padding:5px;
        margin: 10px auto;
        background-color:#124678;
        display: flex;
}
.flex-item{
        display: inherit;
        width:50px;
        height:50px;
        background-color:#ccc;
        margin:5px;
```

```
    }
    .flex-item p{
            width:100%;

5
            text-align:center;
            align-self: center;
            margin:0; }

    .flex-container.nowrap{
           flex-wrap: nowrap;
    }

    <div class="flex-container nowrap">

             <div class="flex-item"><p>1</p></div>

             .....

 </div>
```

flex-wrap: wrap

Si flex-wrap: wrap, el CSS entiende que puede haber un cambio de línea. Los elementos flex aparecen colocados en varias líneas, tantas como sea necesario.

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

flex-wrap: wrap-reverse

Si flex-wrap: wrap-reverse, el CSS entiende que puede haber un cambio de línea. Los elementos flex aparecen colocados en varias líneas, pero en orden

contrario.

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

La propiedad flex-flow

La propiedad flex-flow es una propiedad del contenedor flex.

Para que podamos escribir menos código, el CSS nos permite abreviar las dos propiedades: flex-direction y flex-wrap (opcional) en una sola: flex-flow. El valor por defecto es row nowrap.

.contenedor { flex-flow: flex-direction [flex-wrap]; }

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

4.4 La propiedad align-items

Podemos controlar el alineamiento de los elementos de una caja flexible a lo largo de su eje transversal con align-items.

La propiedad align-items es una propiedad del contenedor flex y puede tomar una de estas valores:

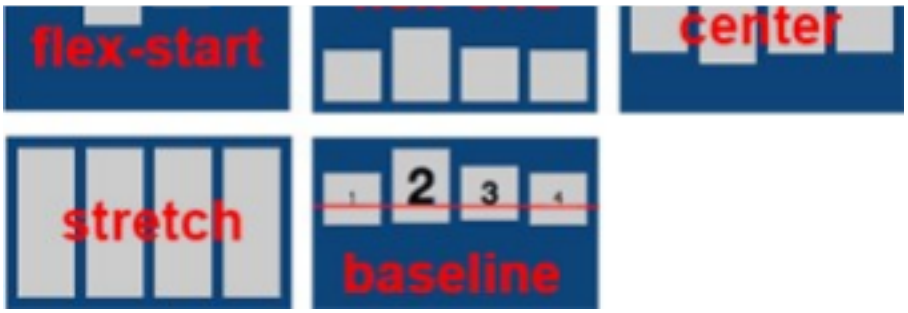
.contenedor { align-items: flex-start | flex-end | center | baseline | stretch; }

flex-start: los elementos aparecen agrupados al principio (*start*) del eje transversal.

flex-end: los elementos aparecen agrupados al final (*end*) del eje transversal.

center: los elementos aparecen agrupados al centro (*center*) de la caja. 7

stretch (el valor por defecto): los elementos aparecen estirados (*stretched*) para ocupar el espacio restante.



baseline: los elementos aparecen alineados relativamente a su línea de base (*baseline*).

align-items: flex-start

Si queremos que los elementos (*items*) aparezcan agrupados al principio (*start*) del eje transversal de la caja flex utilizamos align-items: flex-start;

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

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

.flex-container.flex-start{

        align-items: flex-start;
}
```

align-items: flex-end

Si queremos que los elementos (*items*) aparezcan agrupados al final (*end*) del eje transversal de la caja flex utilizamos align-items: flex-end;

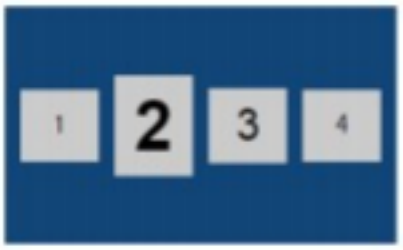
```
.flex-container.flex-end{
align-items: flex-end;
}
```

align-items: center

Si queremos que los elementos (*items*) aparezcan agrupados en el centro (*center*) de la caja flex utilizamos align-items: center;

```
.flex-container.center{

align-items: center;
}
```

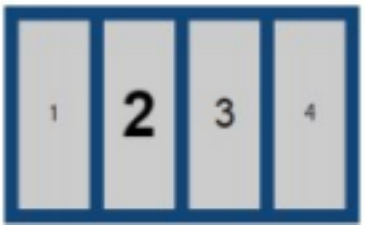


align-items: stretch

Si queremos que los elementos (*items*) de la caja flex aparezcan estirados (*stretched*) ocupando el espacio restante, utilizamos align-items: stretch;

Nota: Si los items tienen Altura (heigth) no se podrán estirar.

```
.flex-container.stretch{
align-items: stretch;
}
```



align-items: baseline

Si utilizamos align-items: baseline; los elementos (*items*) aparecen distribuidos uniformemente: al principio, en el centro y al final del contenedor flex.

```
.flex-container.baseline{
        align-items: baseline;
}
```

4.5 La propiedad justify-content

Podemos controlar el alineamiento de los elementos de una caja flexible (*flexbox*) a lo largo de su eje principal con justify-content.

La propiedad justify-content es una propiedad del contenedor flex y puede tomar una de estas valores:

.contenedor { justify-content: flex-start | flex-end | center | space-between | space around; } flex-start (el

valor por defecto): los elementos aparecen agrupados al principio (*start*) del eje principal.

flex-end: los elementos aparecen agrupados al final (*end*) del eje principal. center: los

elementos aparecen agrupados al centro (*center*).

space-between: los elementos aparecen distribuidos uniformemente: al principio, en el centro y al final del contenedor flex.

space-around: los elementos aparecen distribuidos uniformemente, y con un espacio igual

entre ellos.

10

justify-content: flex-start

Si queremos que los elementos (*items*) aparezcan agrupados al principio (*start*) del eje principal de la caja flex utilizamos justify-content: flex-start;

```
.flex-container{
        display: flex;
}
.flex-container.flex-start{

        justify-content: flex-start;
}
<div class="flex-container flex-start">
      <div class="flex-item"><p>1</p></div>
      <div class="flex-item"><p>2</p></div>
      <div class="flex-item"><p>3</p></div>
</div>
```

justify-content: flex-end

Si queremos que los elementos (*items*) aparezcan agrupados al final (*end*) del eje principal de la caja flex utilizamos justify-content: flex-end;

```
.flex-container.flex-end{
        -webkit-justify-content: flex-end;
        -ms-flex-pack: end; justify-content: flex-end;
}
```

justify-content: center

Si queremos que los elementos (*items*) aparezcan agrupados en el centro (*center*) de la caja flex utilizamosjustify-content: center;

11

```
.flex-container.center{
        justify-content: center;
}
```

justify-content: space-between

Si utilizamos justify-content: space-between; los elementos (*items*) aparecen distribuidos uniformemente: al principio, en el centro y al final del contenedor flex.

```
.flex-container.space-between{

        justify-content: space-between;
```

```
}
```

justify-content: space-around

Si utilizamos justify-content: space-around; los elementos (*items*) aparecen distribuidos uniformemente, y con un espacio igual entre ellos.

```
.flex-container.space-around{
        justify-content: space-around;
}
```

4.6 La propiedad align-content

Podemos controlar el alineamiento de los elementos de una caja flexible (*flexbox*) a lo largo de su eje principal con justify-content o a lo largo de su eje transversal con align items.

Pero, a veces, los elementos de la caja flex pueden ocupar varias líneas (vea flex-wrap). En este caso podemos controlar el alineamiento de los elementos flex utilizando la propiedad align-content.

La propiedad align-content es una propiedad del contenedor flex y puede tomar una de estos valores:

.contenedor { align-content: flex-start | flex-end | center | space-between | space-around | stretch; }

flex-start: los elementos aparecen agrupados al principio (*start*) del eje transversal.

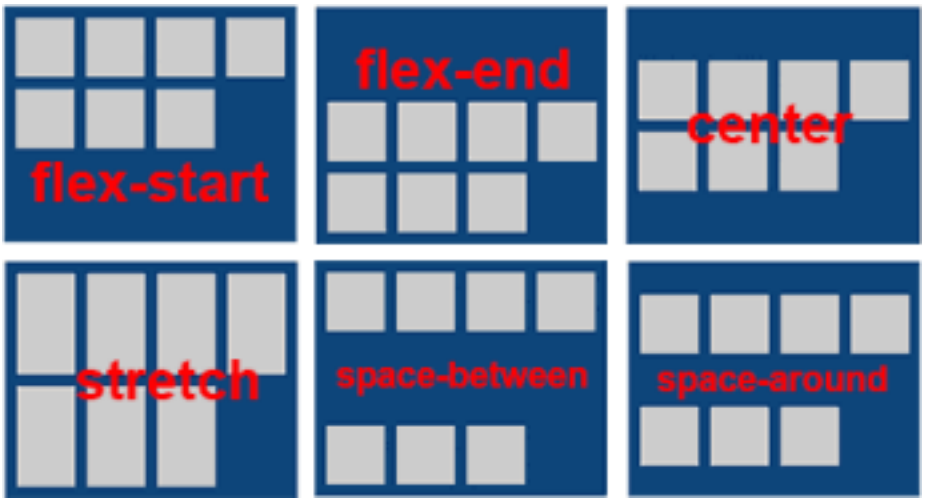
flex-end: los elementos aparecen agrupados al final (*end*) del eje transversal. center: los

elementos aparecen agrupados al centro (*center*).

stretch (el valor por defecto): los elementos aparecen estirados (*stretched*) para ocupar el espacio restante.

space-between: los elementos aparecen distribuidos uniformemente: al principio, en el centro y al final del contenedor flex.

space-around: los elementos aparecen distribuidos uniformemente, y con un espacio igual entre ellos.



align-content: flex-start

Si queremos que los elementos (*items*) aparezcan agrupados al principio (*start*) del eje transversal de la caja flex utilizamos `align-content: flex start;`

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

13

```
.flex-container{
	flex-wrap: wrap;
}
.flex-container.flex-start{

	align-content: flex-start;
}
<div class="flex-container flex-start">
	<div class="flex-item"><p>1</p></div>
	...

</div>
```

align-content: flex-end

Si queremos que los elementos (*items*) aparezcan agrupados al final (*end*) del eje transversal de la caja flex utilizamos `align-content: flex-end;`.

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


align-content: center

Si queremos que los elementos (*items*) aparezcan agrupados en el centro (*center*) de la caja flex utilizamos `align-content: center;`

```
.flex-container.center{
        align-content: center;
}
```

align-content: stretch

Si queremos que los elementos (*items*) de la caja flex aparezcan estirados (*stretched*) ocupando el espacio restante, utilizamos align-content: stretch;

```
.flex-container.stretch{
        align-content: stretch;
}
```


align-content: space-between

Si utilizamos align-content: space-between; los elementos (*items*) aparecen distribuidos uniformemente: al principio, en el centro y al final del contenedor flex.

```
.flex-container.space-between{
        align-content: space-between;
}
```

align-content: space-around

Si utilizamos align-content: space-around; los elementos (*items*) aparecen distribuidos uniformemente, y con un espacio igual entre ellos.

```
.flex-container.space-around{

        align-content: space-around;
}
```

16

4.7 La propiedad align-self

La propiedad align-self reposiciona elementos individuales relativamente al eje transversal de la caja. Generalmente se trata de elementos posicionados con align-items.

La propiedad align-self es una propiedad de los ítems del contenedor flex y puede tomar una de estas valores:

.item { align-self: flex-start | flex-end | center | baseline | stretch; }

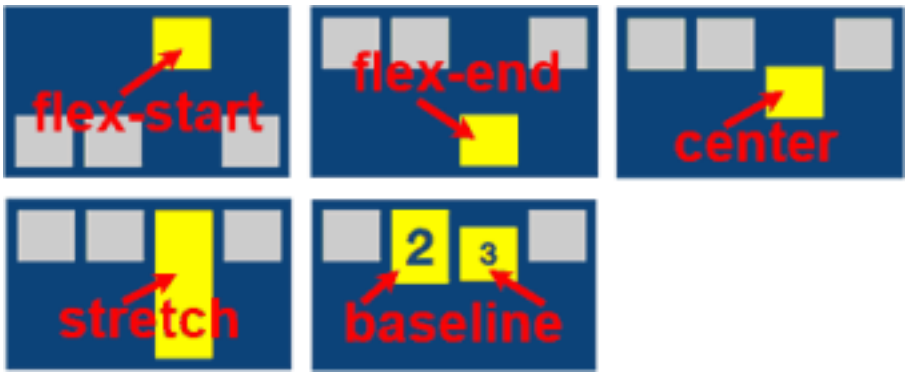
flex-start: el elemento aparece al principio (*start*) del eje transversal.

flex-end: el elemento aparece al final (*end*) del eje transversal.

center: el elemento posicionado aparece en el centro (*center*) de la caja flex.

stretch (el valor por defecto): el elemento aparece estirado (*stretched*) para ocupar el espacio restante.

baseline: los elementos aparecen alineados relativamente a su línea de base (*baseline*).



align-self: flex-start

Si queremos que el elemento (*item*) aparezca al principio (*start*) del eje transversal de la caja flex utilizamos align-self: flex-start;

```
.flex-container{
        display: flex;
        align-items: flex-start;
}
.flex-item:nth-of-type(3){background-color:yellow;}

.flex-container .flex-item *{

17
        width:100%;
        text-align:center;
        align-self: center;
        margin:10px;}

.flex-container .flex-start{

        align-self: flex-start;
}
<div class="flex-container">
<div class="flex-item"><p>1</p></div>
        <div class="flex-item"><p>2</p></div>
        <div class="flex-item flex-start"><p>3</p></div>
        <div class="flex-item"><p>4</p></div>
</div>
```

align-self: flex-end

Si queremos que el elemento (*item*) aparezca al final (*end*) del eje transversal de la caja flex utilizamos align-self: flex-end;

```
.flex-container .flex-end{
        align-self: flex-end;
}
```


align-self: center

Si queremos que el elemento (*item*) aparezca en el centro (*center*) de la caja flex utilizamos align-self: center;

```
.flex-container .center{
        align-self: center;
}
```

align-self: stretch

Si queremos que el elemento (*item*) aparezca estirado (*stretched*) ocupando el espacio restante, utilizamos align-self: stretch;

```
.flex-container .stretch{
        align-self: stretch;
}
```

19

align-self: baseline

Si utilizamos align-self: baseline; los elementos (*items*) aparecen alineados relativamente a su línea de base (*baseline*).

```
.flex-container .baseline{
        align-self: baseline;
        }
```

4.8 La propiedad flex

La propiedad flex es una propiedad de los ítems del contenedor flex.

Para que podamos escribir menos código, el CSS nos permite abreviar las tres propiedades: flex-grow, flex-shrink(opcional) y flex-basis (opcional) en una sola: flex. El valor por defecto es flex: 0 1 auto.

```
.item { flex: flex-grow [flex-shrink] [flex-basis]; }
.item {
 flex: 0 1 auto;
}
```

20
Veamos las tres propiedades:

La propiedad flex-grow

La propiedad flex-grow es una propiedad de los ítems del contenedor flex.

La propiedad flex-grow establece cuánto puede crecer un elemento flex en relación al resto de elementos de la misma caja flex. Su valor es un número.

El valor por defecto de flex-grow: 0, lo que quiere decir que el elemento no puede crecer. Si todos los ítems de una caja tienen el mismo valor de flex-grow, por ejemplo flex-grow:1; quiere decir que todos los ítems tienen que crecer en igual proporción, hasta ocupar todo el espacio disponible.

```
.flex-item{
 flex-grow:0;
}
```

Veamos un ejemplo: los contenedores .flex-container tienen una anchura height: 250px. Anidados dentro de cada contenedor hay 4 ítems (elementos flex) cuya anchura declarada es

de 10%. Esto genera un espacio restante de 60%.

Queremos que el segundo contenedor .flex-container quede completamente ocupado. Decidimos que el segundo ítem crecerá ocupando una parte del espacio restante (flex grow:1), y el cuarto elemento dos partes (flex-grow:2). Los demás elementos siguen igual.

```
.flex-container{
        width: 250px;
        height: 70px;
        padding:5px;
        margin: 10px auto;
        background-color:#124678;
        display: flex;
}
[class^="flex-item"] p{
        width:100%;
        text-align:center;
        align-self: center;
        margin:0; }
.flex-item{
        display: inherit;
        width:10%;
```



```
        height:50px;
        background-color:#ccc;
        margin:5px;
}

.flex-item.flex-grow1{

        flex-grow:1;
}
.flex-item.flex-grow2{

        flex-grow:2;
}
<div class="flex-container">
```

```
 <div class="flex-item"><p>1</p></div>
 <div class="flex-item"><p>2</p></div>
 <div class="flex-item"><p>3</p></div>
 <div class="flex-item"><p>4</p></div>
</div>


<div class="flex-container">
 <div class="flex-item"><p>1</p></div>
 <div class="flex-item flex-grow1"><p>2</p></div>
<div class="flex-item"><p>3</p></div>
 <div class="flex-item flex-grow2"><p>4</p></div>
</div>
```

La propiedad flex-shrink

La propiedad flex-shrink es una propiedad de los ítems del contenedor flex.

La propiedad flex-shrink establece cuánto puede disminuir un elemento flex en relación al resto de elementos de la misma caja flex. Su valor es un número.
El valor por defecto de flex-shrink: 1, lo que quiere decir que los elementos de una caja flex disminuirán en igual proporción, por tal de acomodarse dentro de la caja.

```
.flex-item{
 flex-shrink:1;
}
```

Veamos un ejemplo: los dos contenedores .flex-container tienen una anchura height: 250px. Anidados dentro de cada contenedor hay 4 ítems (elementos flex) cuya anchura declarada es de 40%, y no puede haber cambio de línea (por defecto flex-wrap: nowrap). Esto genera un exceso de 60%. Si no hacemos nada, todos los ítems disminuirán por igual (el primer contenedor).

En el segundo contenedor .flex-container decidimos que el tercer ítem disminuirá dos veces más que los demás elementos (flex-shrink: 2;)

```
.flex-item1{
        display: inherit;
        width:40%;
        height:50px;
        background-color:#ccc;
        margin:5px;
}

.flex-item1.flex-shrink2{
        flex-shrink:2;
}
<div class="flex-container">
 <div class="flex-item1"><p>1</p></div>
 <div class="flex-item1"><p>2</p></div>
 <div class="flex-item1"><p>3</p></div>
 <div class="flex-item1"><p>4</p></div>
</div>



<div class="flex-container">
 <div class="flex-item1"><p>1</p></div>
 <div class="flex-item1"><p>2</p></div>
 <div class="flex-item1 flex-shrink2"><p>3</p></div>
 <div class="flex-item1"><p>4</p></div>
</div>
```

23

La propiedad flex-basis

La propiedad `flex-basis` es una propiedad de los ítems del contenedor flex.

La propiedad flex-basis especifica el valor inicial del tamaño principal de un elemento flex, antes de que esté redimensionado con flex-grow o flex-shrink.

Recuerde que: el tamaño principal de un elemento flex es la anchura en contenedores horizontales - donde flex-direction: row; o la altura en contenedores verticales - donde flex direction: column.

El valor por defecto de es flex-basis: auto. En este caso la anchura base del ítem es igual a la

anchura declarada (width) del elemento, o en su defecto, la anchura calculada por el navegador en base a su contenido.

```
.item {

 flex-basis: auto;
}
```

En el siguiente ejemplo los ítems flex (.flex-item) tienen una anchura width:10%. Para el cuarto elemento de la segunda caja (.flex-basis50) establecemos flex-basis:60%. Observamos como flex-basis sobrescribe la anchura declarada del elemento.

```
.flex-item {
 width:10%;
}
.flex-basis50 {
 flex-basis: 60%;
}
<div class="flex-container">
 <div class="flex-item"><p>1</p></div>
 <div class="flex-item"><p>2</p></div>
 <div class="flex-item"><p>3</p></div>
 <div class="flex-item flex-basis50"><p>4</p></div>
</div>
```



```
<div class="flex-container">
 <div class="flex-item"><p>1</p></div>
 <div class="flex-item"><p>2</p></div>
 <div class="flex-item"><p>3</p></div>
 <div class="flex-item"><p>3</p></div>
```

4.9 La propiedad order

Por defecto los elementos flex, como todos los elementos HTML aparecen en el mismo orden que en el código. En cajas flex podemos alterar este orden utilizando la propiedad order.

La propiedad order es una propiedad de los ítems del contenedor flex y su valor es generalmente un número entero, positivo o negativo.

.item { order: número | initial | inherit; }

El valor por defecto de order es 0. Todos los elementos flex con el mismo valor, aparecen en el mismo orden que en el código. En el siguiente ejemplo el tercer elemento .flex-item tiene el orden order:-1, lo que hace que se desplace delante de los demás elementos .flex-item que tienen el orden order: 0 (el valor por defecto).

Por de otra parte el primer elemento .flex-item tiene el orden order: 1, y por lo tanto aparece detrás de los demás elementos.

```
.flex-item:nth-of-type(3){ order:-1;
background-color:yellow;} .flex-item:nth-of-type(1){ order:
1; background-color:tomato;}
.flex-container{
        width: 250px;
        height:150px;
        padding:5px;
        margin: 10px auto;
        background-color:#124678;

25
        display: flex;
}
.flex-item{
        display: inherit;
        width:50px;
        background-color:#ccc;
        margin:5px;
}

.flex-container .flex-item *{
        width:100%;
        text-align:center;
        align-self: center;
        margin:10px;}

.flex-item:nth-of-type(3){ order:-1;
```

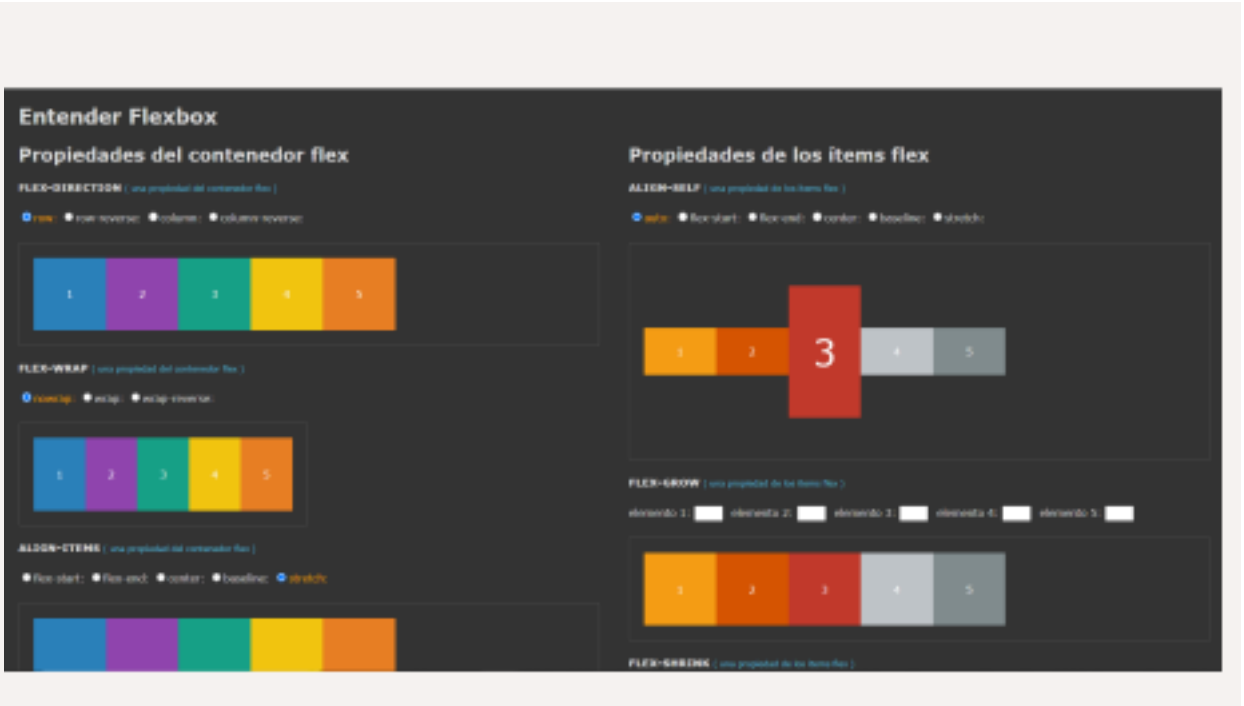
```
background-color:yellow;} .flex-item:nth-of-type(1){ order:1;
background-color:tomato;} <div class="flex-container">
 <div class="flex-item"><p>1</p></div>
 <div class="flex-item"><p>2</p></div>
 <div class="flex-item"><p>3</p></div>
 <div class="flex-item"><p>4</p></div>
</div>
```

advertencia: el orden depende en última instancia de la dirección de los elementos dentro del contenedor flex (establecida con flex-direction, flex-wrap o flex-flow). La única diferencia entre el siguiente ejemplo y el ejemplo anterior es la dirección de los elementos determinada por la propiedad flex-direction.

```
.flex-container.row-reverse{
        flex-direction: row-reverse;
```



```
        }
```



Recursos: Entender FLEX-BOX: Link Documentación en MDN: link