Flexbox

Introducción

¿Qué es flexbox?

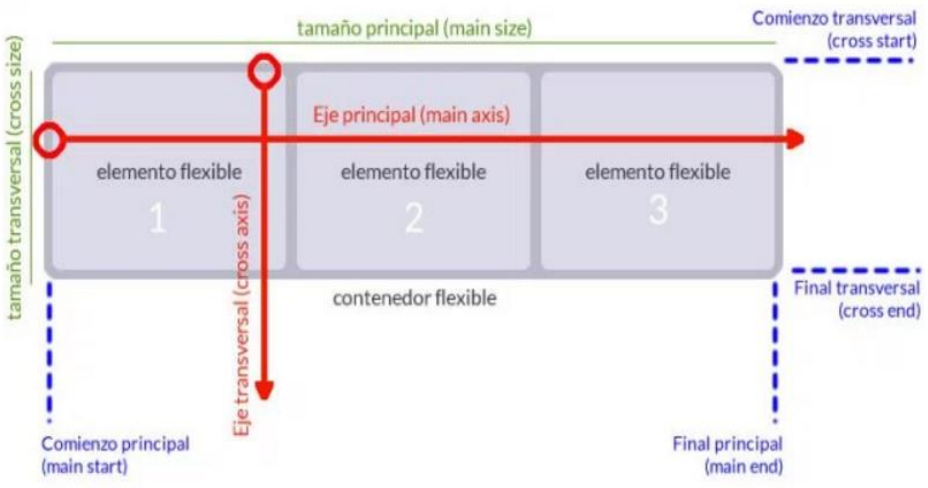
Flexbox (o el módulo de cajas flexibles) es probablemente uno de los más completos y eficaces módulos de maquetación. Todo lo que era complicado en versiones anteriores de CSS (como centrar verticalmente o diseñar estructuras que se redimensionen con elegancia) con flexbox (CSS3) es ya una tarea muy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

-

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

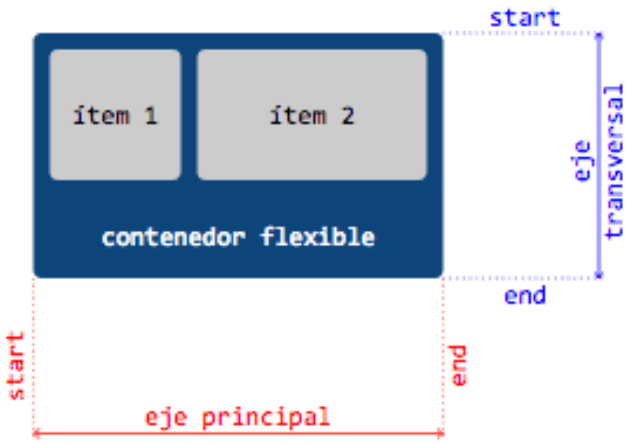
```CSS
.flex-container{
        display:-webkit-flex;
        display: -ms-flexbox;
        display: flex;
}
.flex-container-inline{
        display:-webkit-inline-flex;
        display: -ms-inline-flexbox;
        display: inline-flex;
}
```

1. La propiedad flex-direction

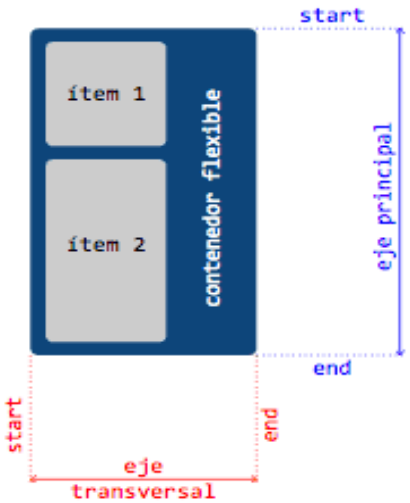
La propiedad `flex-direction` es una propiedad del contenedor flex.

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 cual es el eje principal de la caja y por lo tanto la dirección de los elementos hijos.

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.)

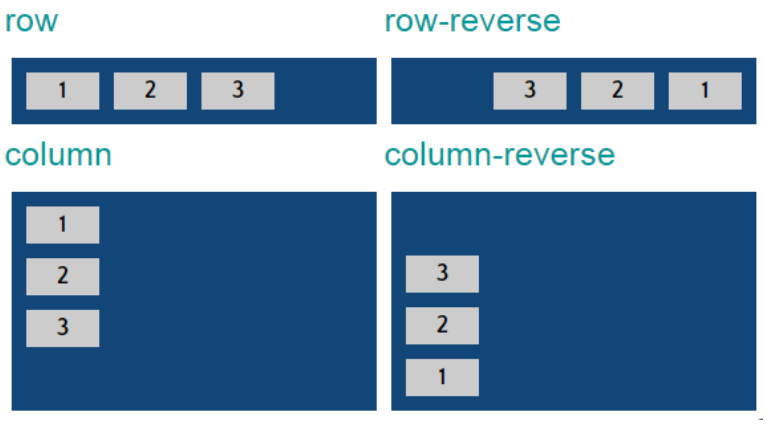
`column`: coloca los elementos flex verticalmente y de arriba abajo (*tb - top to bottom*).

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

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

```
.flex-container.flex-row{
                -webkit-flex-direction: row;
                -ms-flex-direction: row;
                flex-direction: row;
        }
.flex-container.row-reverse{
            -webkit-flex-direction: row-reverse;
                -ms-flex-direction: row-reverse;
                flex-direction: row-reverse;
        }
.flex-container.flex-column{
                -webkit-flex-direction: column;
                -ms-flex-direction: column;
                flex-direction: column;
                height: 150px;}
<div class="left">
    <h4>row</h4>
    <div class="flex-container flex-row">
      <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>
  </div>

  <div class="left">
    <h4>row-reverse</h4>
    <div class="flex-container row-reverse">
      <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>
</div>

<div class="left">
  <h4>column</h4>
  <div class="flex-container flex-column">
    <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>
</div>



  <h4>column-reverse</h4>
  <div class="flex-container column-reverse">
    <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>
```



2. 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.

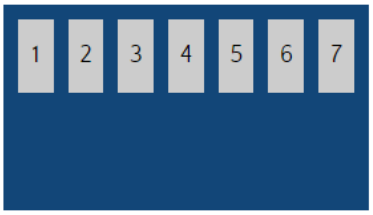
`wrap-reverse`: igual que `wrap` pero las líneas de elementos flex aparecen ordenadas en sentido contrario.

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

```
.flex-container.nowrap{
   -webkit-flex-wrap: nowrap;
   -ms-flex-wrap: nowrap;
   flex-wrap: nowrap;
```

```
}
.flex-container{
        width: 250px;
        height: 140px;
        padding:5px;
        margin: 10px auto;
        background-color:#124678;
        display:-webkit-flex;
        display: -ms-flexbox;
        display: flex;
}
.flex-item{
        display: inherit;
        width:50px;
        height:50px;
        background-color:#ccc;
        margin:5px;
}
.flex-item p{
        width:100%;
        text-align:center;
        -webkit-align-self: center;
        -ms-flex-item-align: center;
        align-self: center;
        margin:0; }

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

<div class="flex-container nowrap">
   <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 class="flex-item"><p>5</p></div>
```

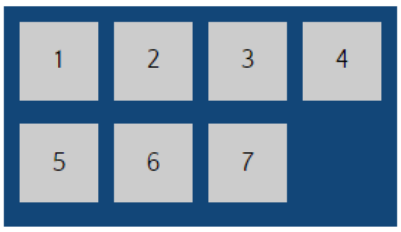
```
    <div class="flex-item"><p>6</p></div>
    <div class="flex-item"><p>7</p></div>
</div>
```



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{
    -webkit-flex-wrap: wrap;
    -ms-flex-wrap: wrap;
    flex-wrap: wrap;
}
```

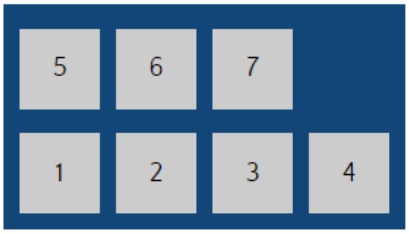


flex-wrap: wrap-reverse

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{
    -webkit-flex-wrap: wrap-reverse;
    -ms-flex-wrap: 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{
   -webkit-flex-flow: row nowrap;
   -ms-flex-flow: row nowrap;
   flex-flow: row nowrap;
}
```

3. 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; }

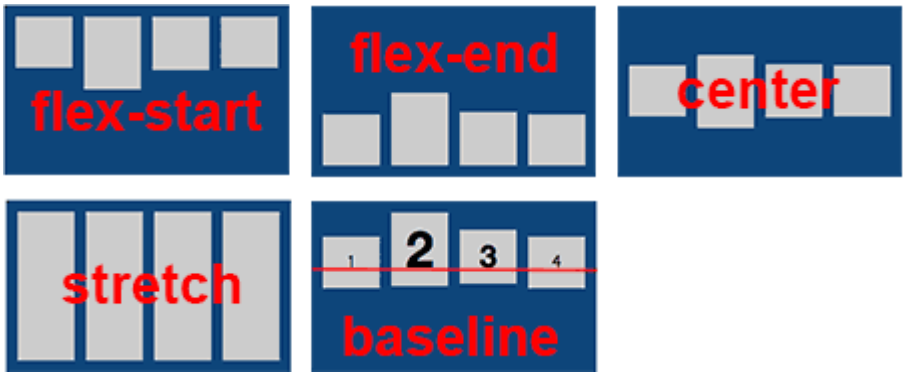
`flex-start`: los elementos aparecen agrupados al principio (*start*) del eje transversal.

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

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

`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*).



Para los usuarios de IE10

Para que los usuarios de IE10 tengan una experiencia similar utilizamos la propiedad `-ms-flex-align` cuyos posibles valores son:

`start` (en lugar de `flex-start`)
`end` (en lugar de `flex-end`)
`center`
`stretch`
`baseline`

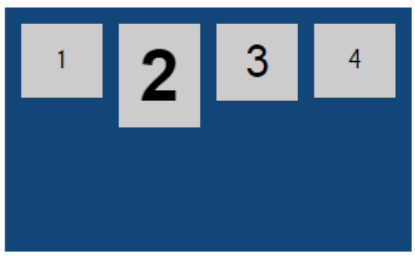
align-items: flex-start

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{
	-webkit-align-items: flex-start;
	-ms-flex-align: start;
	align-items: flex-start;
}
```

```
.flex-container{
	width: 250px;
	height:150px;
	padding:5px;
	margin: 10px auto;
	background-color:#124678;
	display:-webkit-flex;
	display: -ms-flexbox;
	display: flex;
```

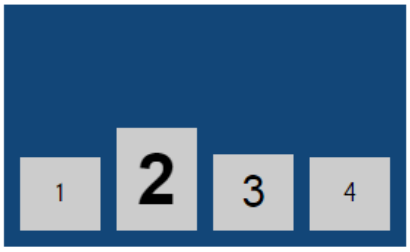
```
}
.flex-item{
        display: inherit;
        width:50px;

        background-color:#ccc;
        margin:5px;
}
.flex-container .flex-item p,
.flex-container .flex-item h1,
.flex-container .flex-item h3{
        width:100%;
        text-align:center;
        -webkit-align-self: center;
        -ms-flex-item-align: center;
        align-self: center;
        margin:10px;
        color:black; }

.flex-container.flex-start{
        -webkit-align-items: flex-start;
        -ms-flex-align: start;
        align-items: flex-start;
}
<div class="flex-container flex-start">
<div class="flex-item"><p>1</p></div>
        <div class="flex-item"><h1>2</h1></div>
        <div class="flex-item"><h3>3</h3></div>
        <div class="flex-item"><p>4</p></div>
</div>
```



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;`

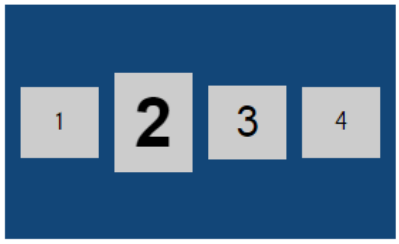
```
.flex-container.flex-end{
        -webkit-align-items: flex-end;
        -ms-flex-align: end;
        align-items: flex-end;
}
```



align-items: center

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

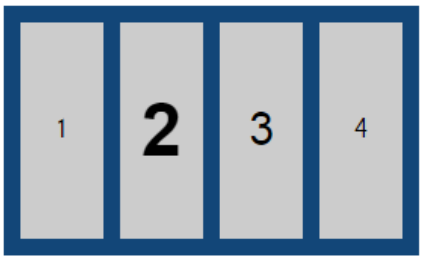
```
.flex-container.center{
        -webkit-align-items: center;
        -ms-flex-align: center;
        align-items: center;
}
```



align-items: stretch

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

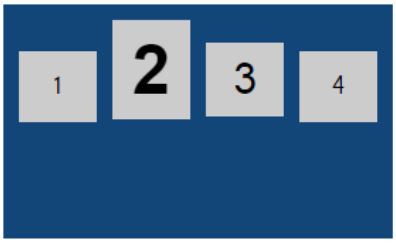
```
.flex-container.stretch{
        -webkit-align-items: stretch;
        -ms-flex-align: stretch;
        align-items: stretch;
}
```



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{
        -webkit-align-items: baseline;
        -ms-flex-align: baseline;
        align-items: baseline;
}
```



4. 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; }
```

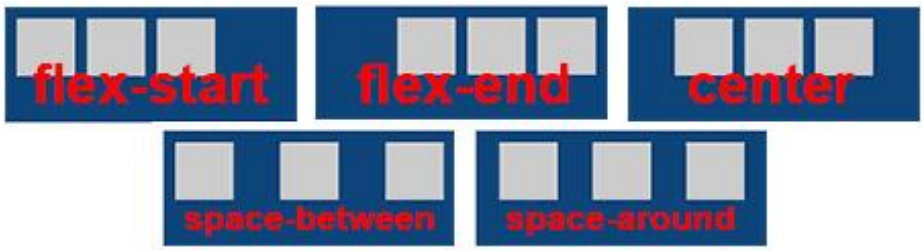
`flex-start` (el valor por defecto): los elementos aparecen agrupados al principio (*start*) del eje principal.

`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.



Para los usuarios de IE10

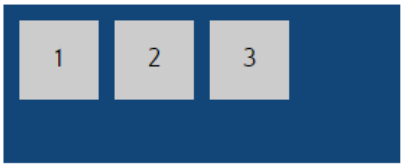
Para que los usuarios de IE10 tengan una experiencia similar utilizamos la propiedad `-ms-flex-line-pack` cuyos posibles valores son:

`start` (en lugar de `flex-start`)
`end` (en lugar de `flex-end`)
`center`
`stretch`
`justify` (en lugar de `space-between` y `space-around`)

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;`

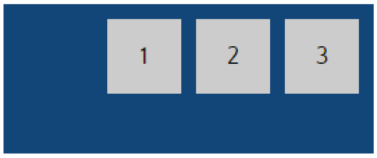
```
.flex-container{
        width: 250px;
        height:100px;
        padding:5px;
        margin: 10px auto;
        background-color:#124678;
        display:-webkit-flex;
        display: -ms-flexbox;
        display: flex;
}
.flex-item{
        display: inherit;
        width:50px;
        height:50px;
        background-color:#ccc;
        margin:5px;
}
.flex-item p{
        width:100%;
        text-align:center;
        -webkit-align-self: center;
        -ms-flex-item-align: center;
        align-self: center;
        margin:0; }

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

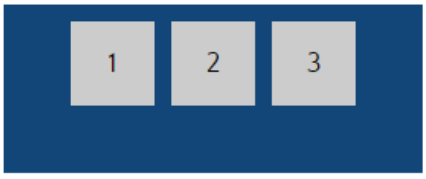
```
.flex-container.flex-end{
        -webkit-justify-content: flex-end;
        -ms-flex-pack: end;
        justify-content: flex-end;
}
```



justify-content: center

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

```
.flex-container.center{
        -webkit-justify-content: center;
        -ms-flex-pack: 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{
        -webkit-justify-content: space-between;
        -ms-flex-pack: justify;
        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{
        -webkit-justify-content: space-around;
        -ms-flex-pack: justify;
        justify-content: space-around;
}
```

5. 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.

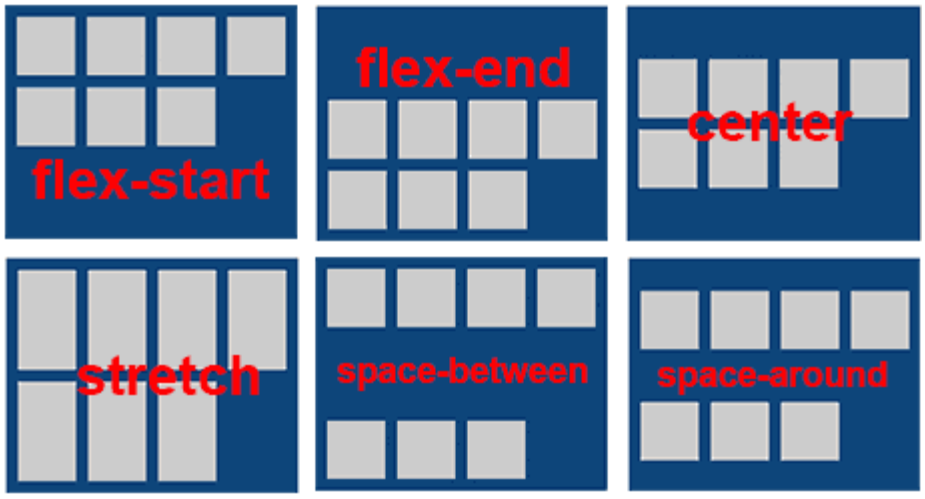
`flex-end`: los elementos aparecen agrupados al final (*end*) del eje transversal.

`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.



Para los usuarios de IE10

Para que los usuarios de IE10 tengan una experiencia similar utilizamos la propiedad `-ms-flex-line-pack` cuyos posibles valores son:

`start` (en lugar de `flex-start`)
`end` (en lugar de `flex-end`)
`center`
`stretch`
`justify` (en lugar de `space-between` y `space-around`)

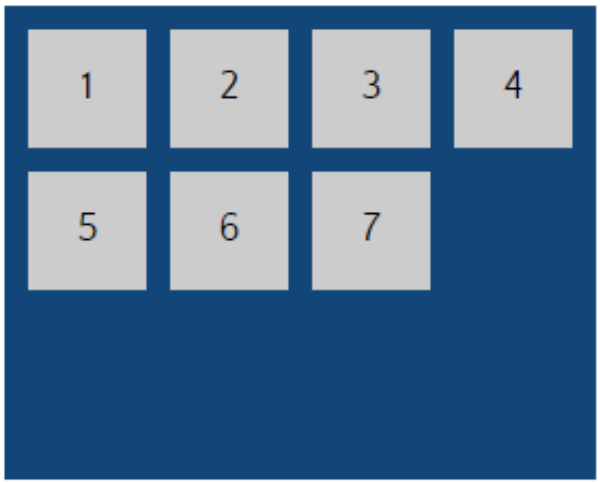
align-content: flex-start

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{
        -webkit-align-content: flex-start;
        -ms-flex-line-pack: start;
        align-content: flex-start;
}
```

```
.flex-container{
        width: 250px;
        height:200px;
        padding:5px;
        margin: 10px auto;
        background-color:#124678;
        display:-webkit-flex;
        display: -ms-flexbox;
        display: flex;
        -webkit-flex-wrap: wrap;
        -ms-flex-wrap: wrap;
        flex-wrap: wrap;
}
.flex-item{
        display: inherit;
        width:50px;
        min-height:50px;
        background-color:#ccc;
        margin:5px;
}
.flex-item p{
        width:100%;
```

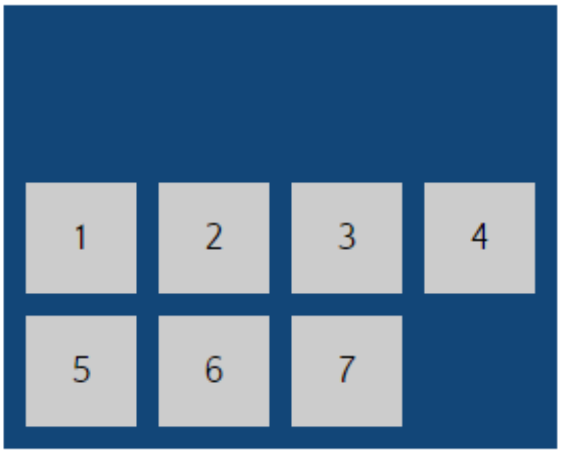
```
        text-align:center;
        -webkit-align-self: center;
        -ms-flex-item-align: center;
        align-self: center;
        margin:0;
        }

.flex-container.flex-start{
        -webkit-align-content: flex-start;
        -ms-flex-line-pack: start;
        align-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 class="flex-item"><p>4</p></div>
   <div class="flex-item"><p>5</p></div>
   <div class="flex-item"><p>6</p></div>
   <div class="flex-item"><p>7</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;`

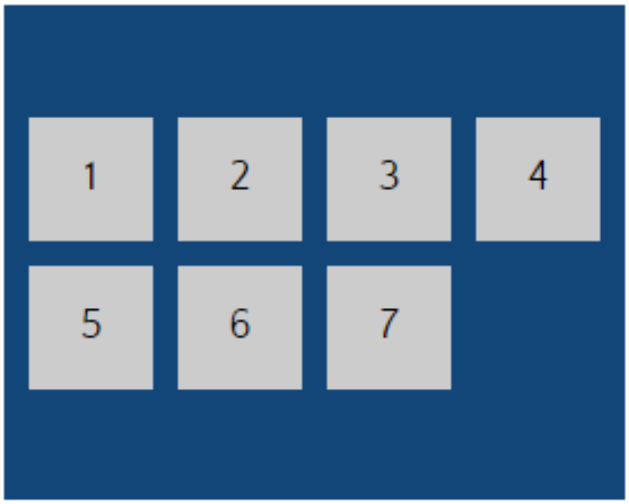
```
.


.flex-container.flex-end{
        -webkit-align-content: flex-end;
        -ms-flex-line-pack: 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;`

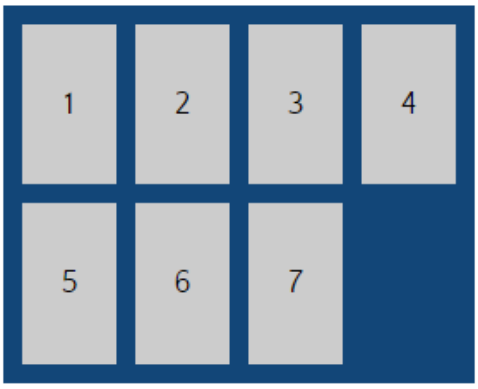
```
.flex-container.center{
        -webkit-align-content: center;
        -ms-flex-line-pack: center;
        align-content: center;
}
```



align-content: stretch

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

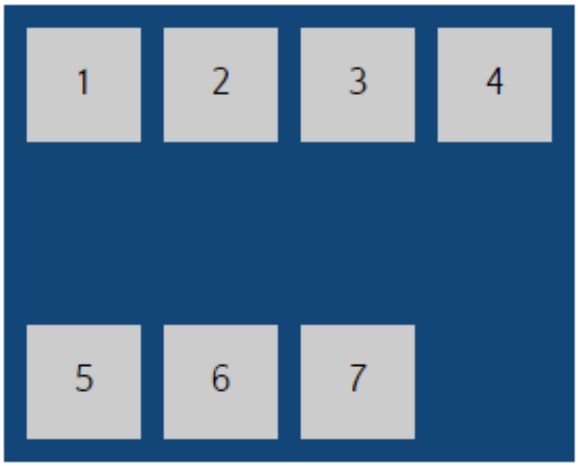
```
.flex-container.stretch{
        -webkit-align-content: stretch;
        -ms-flex-line-pack: stretch;
        align-content: stretch;
}
```



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.

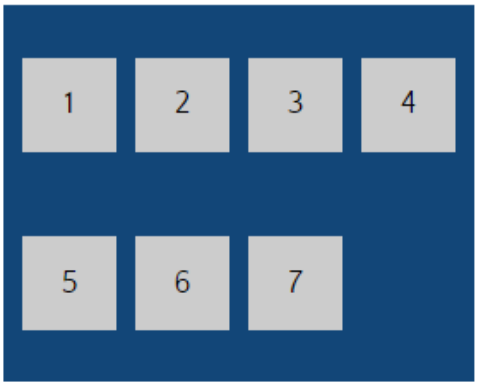
```
.flex-container.space-between{
        -webkit-align-content: space-between;
        -ms-flex-line-pack: justify;
        align-content: space-between;
}
```



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{
        -webkit-align-content: space-around;
        -ms-flex-line-pack: justify;
        align-content: space-around;
}
```



6. 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; }
```

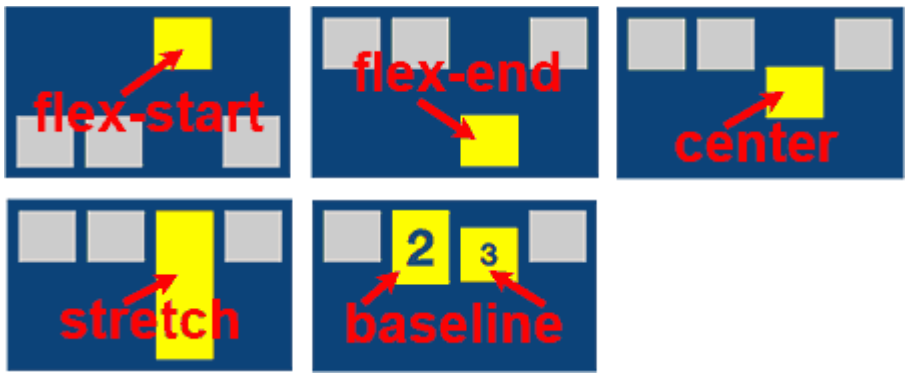
`flex-start`: el elemento aparece al principio (*start*) del eje transversal.

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

`center`: el elemento reposicionado 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*).



Para los usuarios de IE10

Para que los usuarios de IE10 tengan una experiencia similar utilizamos la propiedad `-ms-flex-align` cuyos posibles valores son:

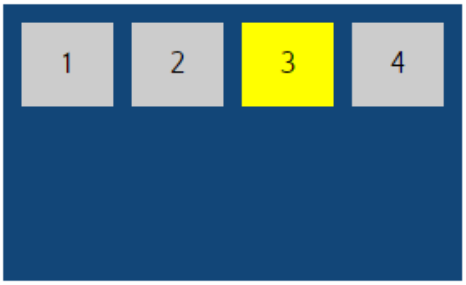
`start` (en lugar de `flex-start`)
`end` (en lugar de `flex-end`)
`center`
`stretch`
`baseline`

align-items: 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{
        width: 250px;
        height:150px;
        padding:5px;
        margin: 10px auto;
        background-color:#124678;
        display:-webkit-flex;
        display: -ms-flexbox;
        display: flex;
        -webkit-align-items: flex-start;
        -ms-flex-align: start;
        align-items: flex-start;
}
.flex-item{
        display: inherit;
        width:50px;
        background-color:#ccc;
        margin:5px;
}
.flex-item:nth-of-type(3){background-color:yellow;}

.flex-container .flex-item *{
        width:100%;
        text-align:center;
        -webkit-align-self: center;
```

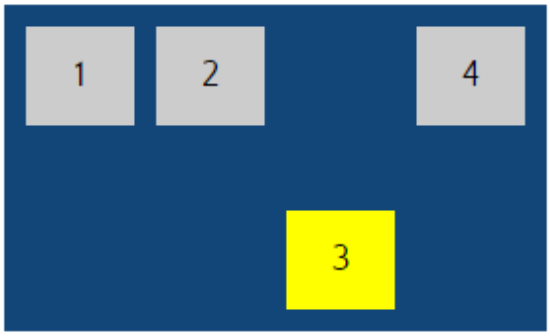
```
        -ms-flex-item-align: center;
        align-self: center;
        margin:10px;}

.flex-container .flex-start{
        -webkit-align-self: flex-start;
        -ms-flex-align: 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-items: flex-start

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

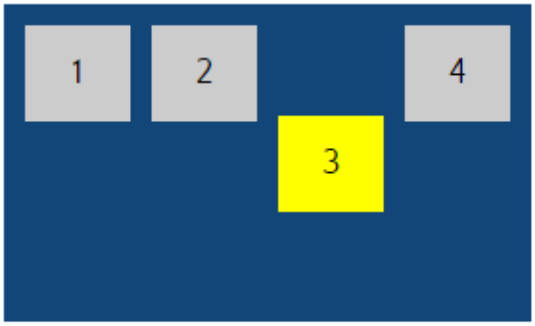
```
.flex-container .flex-end{
        -webkit-align-self: flex-end;
        -ms-flex-align: end;
        align-self: flex-end;
}
```



align-items: center

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

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

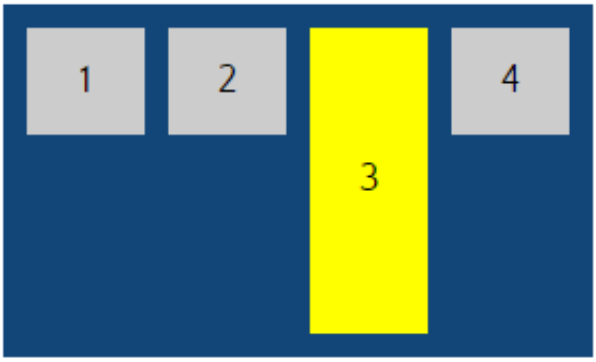


align-items: stretch

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

```
.flex-container .stretch{
        -webkit-align-self: stretch;
        -ms-flex-align: stretch;
        align-self: stretch;
```

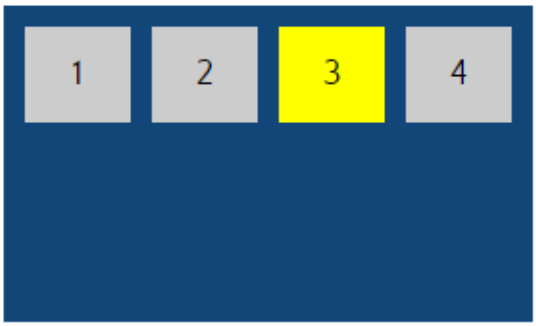
```
}
```



align-items: baseline

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

```
.flex-container .baseline{
        -webkit-align-self: baseline;
        -ms-flex-align: baseline;
        align-self: baseline;
        background-color:yellow;
}
```



7. 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 {
    -webkit-flex: 0 1 auto;
    -ms-flex: 0 1 auto;
    flex: 0 1 auto;
}
```

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 cuanto 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{
    -webkit-flex-grow: 0;
    -ms-flex-grow:0;
    flex-grow:0;
}
```

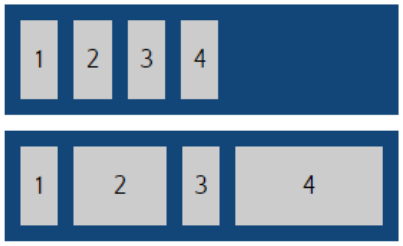
Veamos un ejemplo: los contenedores `.flex-container` tienen una anchura `height: 250px`. Anidados dentro de cada contenedor hay 4 ítems (elementos flex) cuya anchura declarada es de 10%. Esto genera un espacio restante de 60%.

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:-webkit-flex;
        display: -ms-flexbox;
        display: flex;
}
[class^="flex-item"] p{
        width:100%;
        text-align:center;
        -webkit-align-self: center;
        -ms-flex-item-align: center;
        align-self: center;
        margin:0; }
.flex-item{
        display: inherit;
        width:10%;
        height:50px;
        background-color:#ccc;
        margin:5px;
}

.flex-item.flex-grow1{
        -webkit-flex-grow: 1;
    -ms-flex-grow:1;
        flex-grow:1;
}
.flex-item.flex-grow2{
        -webkit-flex-grow: 2;
    -ms-flex-grow:2;
        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 cuanto 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.

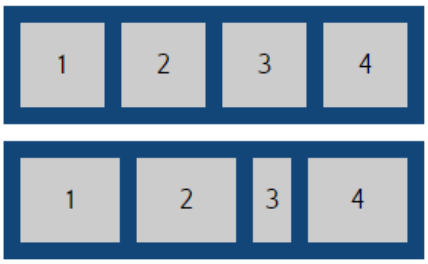
```
.flex-item{
    -webkit-flex-shrink: 1;
    -ms-flex-shrink:1;
    flex-shrink:1;
}
```

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{
        -webkit-flex-shrink: 2;
    -ms-flex-shrink:2;
        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>
```



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`.

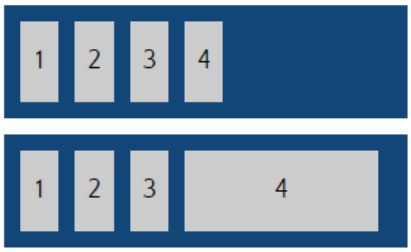
Recuerde que: el tamaño principal de un elemento flex es la anchura en contenedores horizontales - donde `flex-direction: row;` o la altura en contenedores verticales - donde `flex-direction: column`.

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 {
   -webkit-flex-basis: auto;
   -ms-flex-basis: auto;
   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-basis50 {
   -webkit-flex-basis: 50%;
   -ms-flex-basis: 50%;
   flex-basis: 50%;
}
<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"><p>2</p></div>
   <div class="flex-item"><p>3</p></div
```



8. 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; }

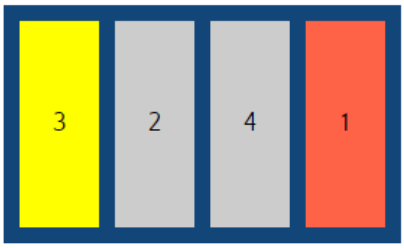
El valor por defecto de `order` es 0. Todos los elementos flex con el mismo valor, aparecen en el mismo orden que en el código. En el siguiente ejemplo el tercer elemento `.flex-item` tiene el orden `order:-1`, lo que hace que se desplace delante de los demás elementos `.flex-item` que tienen el orden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

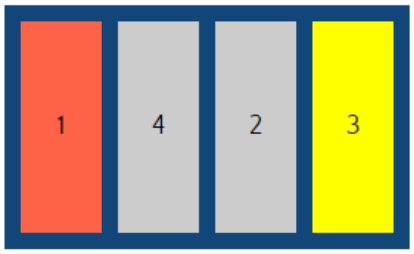
```
.flex-container{
        width: 250px;
        height:150px;
        padding:5px;
        margin: 10px auto;
        background-color:#124678;
        display:-webkit-flex;
        display: -ms-flexbox;
        display: flex;
}
```

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

.flex-container .flex-item *{
        width:100%;
        text-align:center;
        -webkit-align-self: center;
        -ms-flex-item-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 le dirección de los elementos determinada por la propiedad `flex-direction`.

```
.flex-container.row-reverse{
            -webkit-flex-direction: row-reverse;
                -ms-flex-direction: row-reverse;
                flex-direction: row-reverse;
        }
<div class="flex-container row-reverse">
    <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>
```



9. Cómo utilizar Modernizr

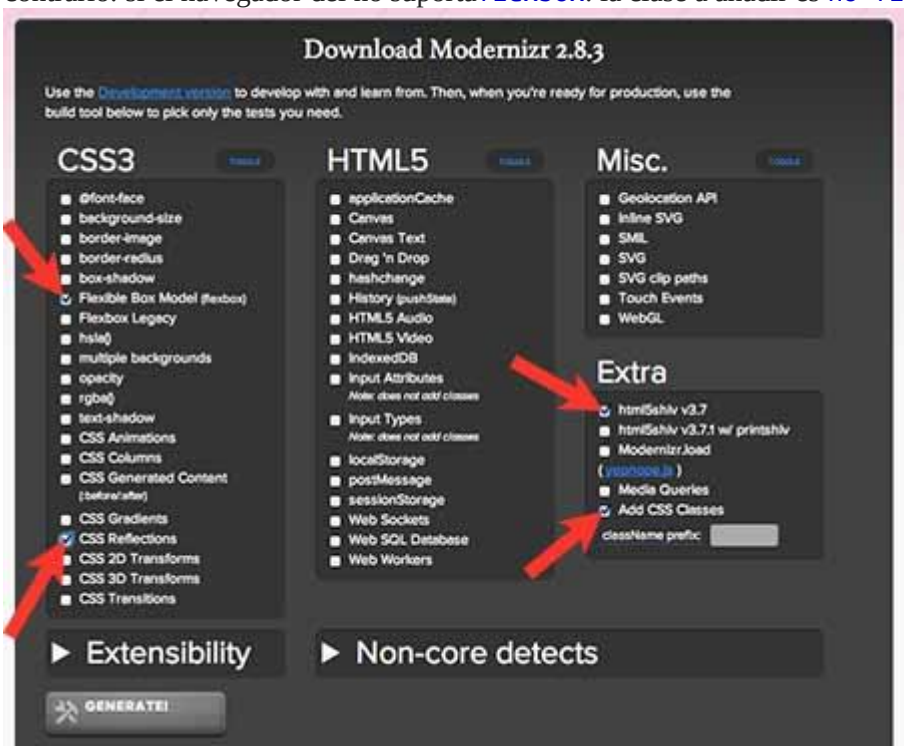
Modernizr es una pequeña librería JavaScript que detecta si el navegador del usuario soporta o no las nuevas funcionalidades del HTML5 y CSS3.
Con Modernizr podemos utilizar las nuevas características del HTML5 y CSS3 y obtener resultados coherentes en los diferentes navegadores.

A continuación veremos como nos puede ayudar Modernizr en un caso concreto: flexbox.

Descargar modernizr

1. Primero marcamos solo lo que nos interesa (descargar modernizr) :

- **Flexible Box Model (flexbox):** en el fieldset que recoge las nuevas características de CSS3 marcamos las casilla Flexible Box Model (flexbox). Esto nos ayuda a detectar si el navegador que utiliza el usuario soporta esta característica o no.
- **CSS Reflections:** en este momento Firefox no soporta los reflejos CSS. Aunque no tenemos intención de utilizar reflejos esto nos viene de maravilla: utilizaremos Firefox para crear y verificar el CSS de substitución: el que utiliza posicionamiento flotante en lugar de flexbox.
- **html5shiv:** una solución JavaScript que hace que los nuevos elementos del HTML5 (`<header>`, `<section>`,`<article>` etc...) sean correctamente interpretados por InternetExplorer 7 y 8.
- **Add CSS Classes**: es una funcionlidad de modernizr que añade clases al `<html>`. En el caso de `flexbox`: si el navegador del usuario suporta `flexbox`: la clase a añadir es `flexbox`. Por lo contrario: si el navegador del no soporta`flexbox`: la clase a añadir es `no-flexbox.`



Después de decidir qué es lo que necesita, pulse el botón **GENERATE!** para generar el código comprimido de modernizr.js. Para un formato sin comprimir marque la casilla **Don't Minify Source**. Podemos copiar el código, o podemos descargarlo.

Como funciona

Después de descargar Modernizr y guardarlo, añadimos un enlace hacia `modernizr.js` o como queremos llamarlo. Lo ponemos en el `<head>`.

```
<script src="js/modernizr.js"></script>
```

Probablemente algunos se preguntan ¿por qué en el `<head>` si sabemos que hay que insertar el JavaScript al final del `<body>`, ya que los scripts bloquean el renderizado mientras se están descargando y ejecutando, y si los ponemos en el `<head>` ralentizan la página.
El problema es que Modernizr tiene el html5shiv incorporado, y para que este funcione tiene que estar en el`<head>`.

El segundo paso es asignar una clase al `<html>`, así:

```
<html class="no-js">
```

Si JavaScript no funciona, tampoco Modernizr, y ni otras funcionalidades JavaScript. Podemos utilizar la clase `.no-js` para dar formato en este caso.

Por de otra parte si JavaScript funciona, una vez que la página se haya cargado, Modermizr elimina la clase `.no-js` y la reemplaza (*dinámicamente*) por por algo asi:

En Chrome 37

```
<html class=" js flexbox flexboxlegacy canvas canvastext webgl no-touch
geolocation postmessage websqldatabase indexeddb hashchange history
draganddrop websockets rgba hsla multiplebgs backgroundsize borderimage
borderradius boxshadow textshadow opacity cssanimations csscolumns
cssgradients cssreflections csstransforms csstransforms3d csstransitions
fontface generatedcontent video audio localstorage sessionstorage
webworkers applicationcache svg inlinesvg smil svgclippaths">
```

En FireFox 32

```
<html class=" js flexbox canvas canvastext webgl no-touch geolocation
postmessage no-websqldatabase indexeddb hashchange history draganddrop
websockets rgba hsla multiplebgs backgroundsize borderimage borderradius
boxshadow textshadow opacity cssanimations csscolumns cssgradients no-
cssreflections csstransforms csstransforms3d csstransitions fontface
generatedcontent video audio localstorage sessionstorage webworkers
applicationcache svg inlinesvg smil svgclippaths">
```

En IE 9

```
<html class=" js no-flexbox canvas canvastext no-webgl no-touch
geolocation postmessage no-websqldatabase no-indexeddb hashchange no-
history draganddrop no-websockets rgba hsla multiplebgs backgroundsize
no-borderimage borderradius boxshadow no-textshadow opacity no-
cssanimations no-csscolumns no-cssgradients no-cssreflections
csstransforms no-csstransforms3d no-csstransitions fontface
generatedcontent video audio localstorage sessionstorage no-webworkers
no-applicationcache svg inlinesvg smil svgclippaths">
```

De hecho en nuestro caso (al generar el código de Modernizr hemos escogido solo dos características del CSS) el código tendrá este aspecto:

<html class=" js flexbox cssreflections"> <!--para Chrome 37 -->
<html class=" js flexbox no-cssreflections"> <!-- para FF 32 -->
<html class=" js no-flexbox no-cssreflections"> <!-- para IE 9 -->

¿Que quiere decir esto?

El Chrome 37 soporta tanto `flexbox` como los reflejos de CSS (`box-reflect`). Por lo tanto Modernizr da al `<html>` una clase `js` (porque el JavaScript funciona), otra clase `flexbox` (porque Chrome soporta las cajas flexibles) y una clase `cssreflections` (porque Chrome también soporta los reflejos de CSS).
Por de otra parte FireFox 32 soporta `flexbox`, pero **no** soporta los reflejos de CSS. Por lo tanto recibirá las clases `js`,`flexbox` y `no-cssreflections`.
Finalmente InternetExplorer 9 que no soporta ni `flexbox` ni reflejos recibirá como clases `js`, `no-flexbox` y `no-cssreflections`.

Es obvio que podemos utilizar estas clases en el CSS para dar formato en función de si el navegador del usuario soporta las nuevas funcionalidades de CSS3 o no.

10. Maquetación flex

En el CSS

En el siguiente ejemplo los nombres de las clases CSS sugieren la funcionalidad flex.
Por ejemplo: la clase `.flex-container` transforma una caja cualquiera en un contenedor flex (`display: flex;`). Los contenedores flex verticales llevan la clase `.column` ya que `flex-direction: column`. Por de otra parte los contenedores flex horizontales (`.row`) tienen asignada también la clase `.justify-space-between` donde:

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

Hemos dividido el código CSS en 3 partes

- **general**: incluye colores, tipo y tamaño de fuente, tipo de caja, márgenes, rellenos etc. . .
- **.flexbox**: para los navegadores que soportan flexbox.
  Incluye los estilos que transforman las cajas de nuestra página, en contenedores o ítems flex. Todos estos estilos empiezan con `.flexbox`, la clase que modernizr da al `<html>` en navegadores que soportan flexbox.
- **.no-flexbox**: para navegadores antiguos que **NO** soportan flexbox.
  Incluye los estilos que transforman las cajas de nuestra página, en cajas flotantes. Todos estos estilos empiezan con `.no-flexbox`, la clase que modernizr da al `<html>` en navegadores que `NO` soportan flexbox.

Planificar la maqueta web

En el `<body>` de la página creamos un `#contenedor` para la maqueta. Le damos una clase de `.flex-container` y otra de `.column` ya que queremos transformarlo en un contenedor flex vertical.

Dentro del contenedor flex colocamos tres elementos, tres ítems flex: un `#encabezado <header>`, una `<section>`, y un `#pie <footer>`.

```
<div id="contenedor" class="flex-container column">
    <header id="encabezado"></header>
    <section></section>
    <footer id="pie"></footer>
</div>
```

A continuación transformamos la `<section>` en un contenedor flex horizontal. Para esto le damos una clase de `.flex-container` y otra de `.row`. Queremos que los elementos dentro de la `<section>` aparecan distribuidos uniformemente: al

principio, en el centro y al final del contenedor flex. Para esto le damos también la clase de `.justify-space-between`.
Dentro de `<section>` colocamos otros tres elementos, que se transforman automáticamente en ítems flex: una barra `#lateral <aside>`, una `<section> #central` y otra caja `<nav>` donde colocaremos la navegación o el menú.

```
<div id="contenedor" class="flex-container column">
<header id="encabezado"></header>
<section class="flex-container justify-space-between row">
    <aside id="lateral"></aside>
    <section id="central"></section>
    <nav id="nav"></nav>
</section>
<footer id="pie"></footer>
</div>
```

Podemos anidar contenedores flex

Es interesante ver que los ítems flex pueden ser a la vez contenedores flex. Esto nos permite anidar cajas flex dentro de otras cajas flex sin más complicaciones, ya que ninguna de las propiedades de este módulo es heredada, excepto si así especificado (`inherit`).

También transformamos la `<section> #central` en un contenedor flex horizontal. Para esto le damos una clase de `.flex-container` y otra de `.row`. Asimismo le damos la clase `.wrap` ya que queremos que los ítems flex de la sección `#central` aparezcan colocados en varias líneas.

Vea la estructura general:

```
<div id="contenedor" class="flex-container column">
<header id="encabezado"></header>
<section class="flex-container justify-space-between row">
    <aside id="lateral" class="flex-container column"></aside>
        <section id="central">
        <div id="escaparate" class="flex-container justify-
space-between wrap row">
        </div>
        </section>
    <nav id="nav">
    <ul class="flex-container column"></ul>
    </nav>
    </section>
<footer id="pie"></footer>
</div>
```

Sugerencia: podemos trabajar utilizando la clase `.no-cssreflections`. Firefox no soporta los reflejos CSS, y podemos verificar fácilmente la viabilidad del código en Firefox. Cuando todo esté como queremos podemos cambiar la clase `.no-cssreflections` por `.no-flexbox`.

Como verificar los resultados

En el internet podemos encontrar varias páginas web que nos ayudan a verificar nuestra "creación" en varios navegadores. He aquí algunas de ellas:

- Browser Screenshots for Quick Testing
- Cross Platform Browser Test - Browsershots
- IE NetRenderer - Browser Compatibility Check