CSS. FLEXBOX

Ya hemos visto en el ejemplo del apartado 6 de HTML como poder componer elementos en la misma fila, vamos a ver un poco más de la funcionalidad que tiene FlexBox para poder realizar distintas composiciones de elementos.

Ejemplo:

En el siguiente ejemplo tenemos un contenedor principal con un identificador llamado *cajaflex* que a su vez tiene en su interior otros contenedores secundarios con identificadores *caja1*, *caja2* y *caja3*. Mediante CSS hemos creado cuatro reglas que aplican sobre cada identificador indicando que el fondo de cada contenedor sea de un color diferente y que el contenedor principal sea un cuadrado de 400px mientras que los secundarios sean cuadrados de 100px.

HTML

```html
<div id="cajaflex">
    <div id="caja1">
    </div>
    <div id="caja2">
    </div>
    <div id="caja3">
    </div>
</div>
```

CSS

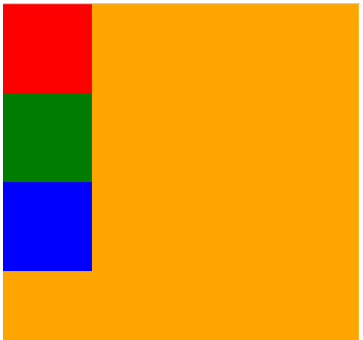
```css
#cajaflex
{
    background-color: orange;
    width:400px;
    height:400px;
}

#caja1
{
    background-color: red;
    width:100px;
    height:100px;
}

#caja2
{
    background-color: green;
    width:100px;
    height:100px;
}

#caja3
{
    background-color: blue;
    width:100px;
    height:100px;
}
```

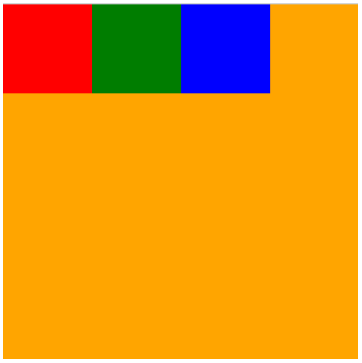
Si visualizamos en el navegador el documento HTML se vería de momento así:



Esto pasa porque los contenedores son elementos de bloque y si recuerdas empiezan siempre en una nueva línea y no permiten que otros elementos se pongan a su lado. Vamos a emplear FlexBox para conseguir que las cajas queden todas en la misma fila. Para poder aplicar FlexBox el primer paso es añadir en la regla del contenedor principal el atributo *display* con valor *flex*.

```
#cajaflex
{
    background-color: orange;
    width:400px;
    height:400px;

    display:flex;
}
```

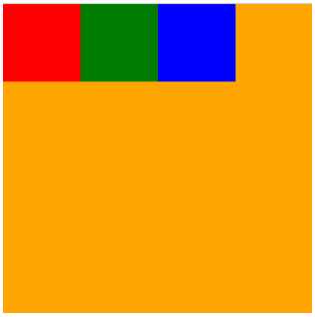


Si compruebas en el navegador las cajas ya saldrán en la misma fila.

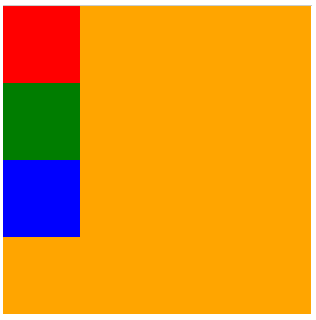
Hay más atributos de FlexBox que permiten configurar la visualización de los elementos de distinta forma. A continuación veremos algunos de ellos:

flex-direction: Nos permite controlar si la disposición de los elementos es en fila (valor por defecto) o en columna. Sus posibles valores son *row* (fila) o *column* (columna).

```
flex-direction: row;
```

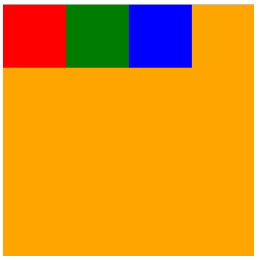
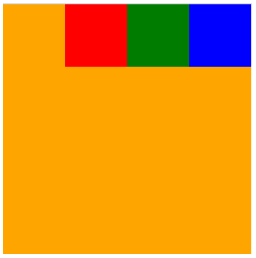
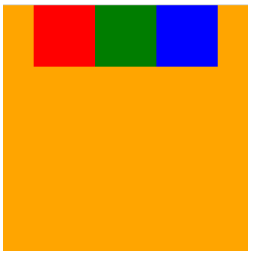
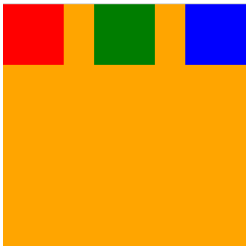


```
flex-direction: column;
```

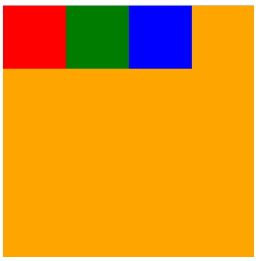
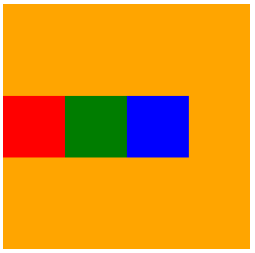


Si los elementos se han dispuesto en fila se dice que el eje principal es el horizontal y si se han dispuesto en columna se dice que el eje principal es el vertical.

justify-content: Nos permite distribuir el espacio entre y alrededor de los elementos a lo largo del eje principal (recuerda que puede ser horizontal o vertical). Algunos de sus posibles valores son *flex-start*, *flex-end*, *center*, *space-between* y *space-around*.

`justify-content: flex-start;` Los elementos se colocan desde el inicio del eje principal.	
`justify-content: flex-end;` Los elementos se colocan desde el final del eje principal.	
`justify-content: center;` Los elementos se colocan en el centro del eje principal.	
`justify-content: space-around;` Los elementos se distribuyen uniformemente sobre el eje principal dejando el mismo espacio por la parte izquierda y derecha de cada elemento.	
`justify-content: space-between;` Los elementos se distribuyen uniformemente sobre el eje principal ocupando todo el ancho.	

align-items: Nos permite alinear los elementos a lo largo del eje perpendicular al principal. Algunos de sus posibles valores son *flex-start*, *flex-end*, y *center*.

`align-items: flex-start;` Los elementos se colocan al inicio.	
`align-items: flex-end;` Los elementos se colocan al final.	
`align-items: center;` Los elementos se colocan en el centro.	

flex-grow: En el caso de que exista espacio sobrante en el eje principal esta propiedad nos permite repartirlo entre los elementos del contenedor. Su valor por defecto es 0, es decir, no se reparte. El valor que se le asigna indica la parte del espacio sobrante que se le dará a ese elemento.

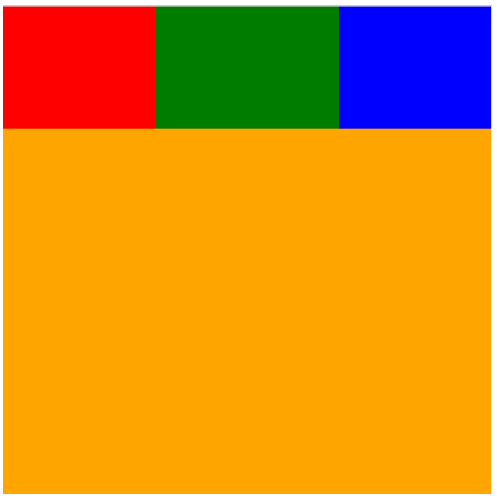
Volviendo a nuestro ejemplo, los tres cuadrados secundarios son de 100px y el contenedor principal tiene un ancho de 400px, eso significa que hay un espacio sobrante de 100px que se puede repartir, veamos dos ejemplos:

Ejemplo 1:

```css
#caja1
{
    background-color: red;
    width:100px;
    height:100px;
    flex-grow: 1;
}

#caja2
{
    background-color: green;
    width:100px;
    height:100px;
    flex-grow: 2;
}

#caja3
{
    background-color: blue;
    width:100px;
    height:100px;
    flex-grow: 1;
}
```



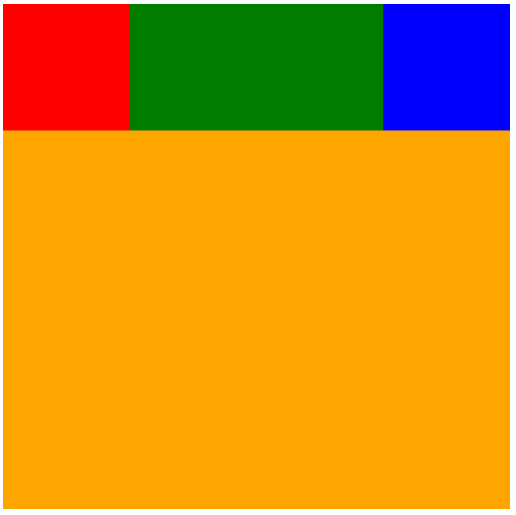
Del espacio sobrante (100px) la caja1 se lleva 1 parte, la caja2 se lleva 2 partes y la caja3 se lleva 1 parte. Es decir la caja1 se lleva 25px, la caja2 se lleva 50px y la caja3 se lleva 25px.

Ejemplo2:

```css
#caja1
{
    background-color: red;
    width:100px;
    height:100px;
    flex-grow: 0;
}

#caja2
{
    background-color: green;
    width:100px;
    height:100px;
    flex-grow: 1;
}

#caja3
{
    background-color: blue;
    width:100px;
    height:100px;
    flex-grow: 0;
}
```



Del espacio sobrante (100px) la caja1 se lleva 0 partes, la caja2 se lleva 1 parte y la caja3 se lleva 0 partes. Es decir la caja2 se lleva 100px.