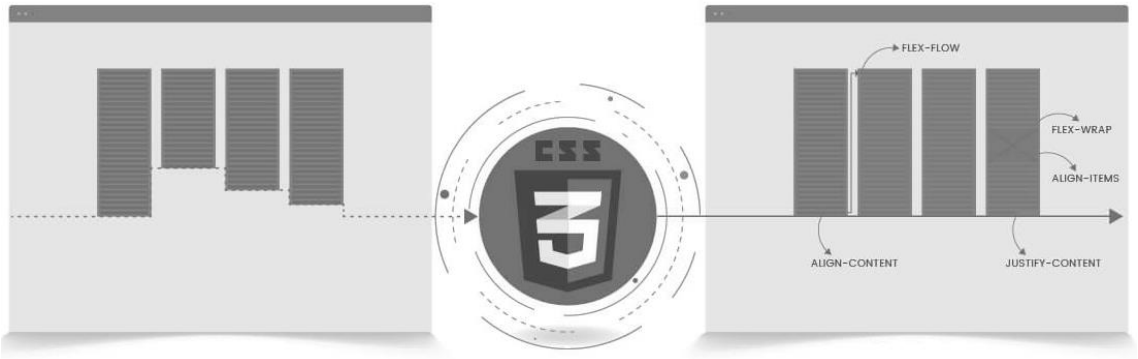
FLEXBOX

"Un mar calmo no hace buenos marineros."

(Proverbio inglés)

INTRODUCCION A FLEXBOX CSS

No hace mucho tiempo, el diseño de todas las páginas HTML se hacía a través de tablas HTML o el uso de las propiedades float y clear de CSS que no eran adecuadas para diseñar páginas web complejas. El nuevo Modelo de Caja Flexible, comúnmente llamado flexbox, es un mecanismo de CSS3 que nos permite ubicar elementos de nuestro sitio para que se acomoden lo mejor posible al espacio disponible en cualquier dispositivo. Gracias a FlexBox nuestro layout será más simple y usará menos código.
Antes de comenzar es importante tener en cuenta que un flexbox no es una propiedad de CSS sino un conjunto de propiedades y valores que debemos aplicar a los elementos de nuestro layout de una manera concreta.



¿CÓMO CREAR UNA CAJA FLEX?

Flexbox representa un modelo 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 reorganizarse dentro de la caja flex con facilidad. También tienen la capacidad de alinearse tanto horizontalmente como verticalmente y esto puede ser muy interesante a la hora de diseñar páginas web adaptativas.

La propiedad display usada en CSS3 cuando adquiere la capacidad de transformar una caja cualquiera en un contenedor flex. Para esto tiene que tomar uno de estos valores: flex o flex-inline. Por ejemplo:

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

Los elementos hijos de ese contenedor obtienen las propiedades de Flexbox donde el primer comportamiento que notará es que estos elementos se convierten en columnas ocupando el ancho de acuerdo con su contenido y que además ocuparan todo el alto disponible de su contenedor.

Nota

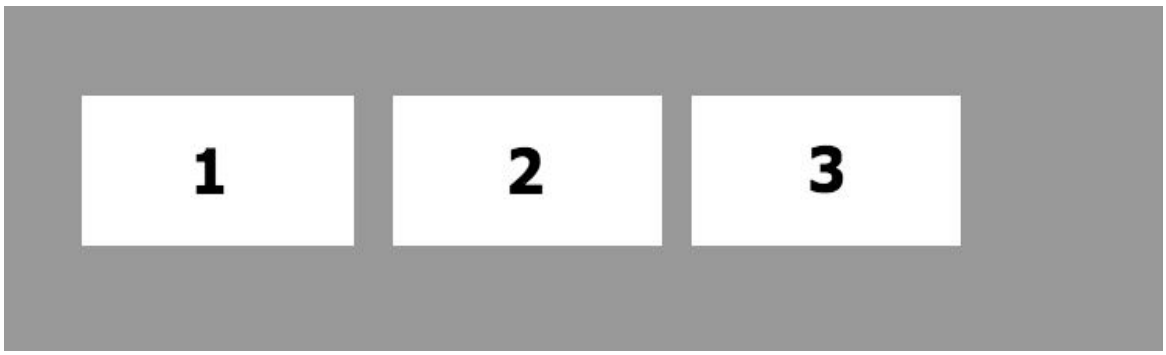
Otra de las innovaciones que se introdujeron con CSS3 se denomina CSS Grid, una tecnología que ofrece a los diseñadores web otras formas de distribuir elementos en la interfaz.

FILAS O COLUMNAS

Anteriormente vimos que por defecto los elementos dentro de nuestro contenedor flex se acomodaban de manera horizontal uno detrás del otro. Cada flexbox sigue dos ejes. El eje principal (main-axis) es el eje a lo largo del cual los elementos flexibles se suceden unos a otros. El eje secundario es el eje perpendicular al eje principal. También podemos usar la propiedad flex-direction para modificar fácilmente la dirección de los elementos. Ejemplo:

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

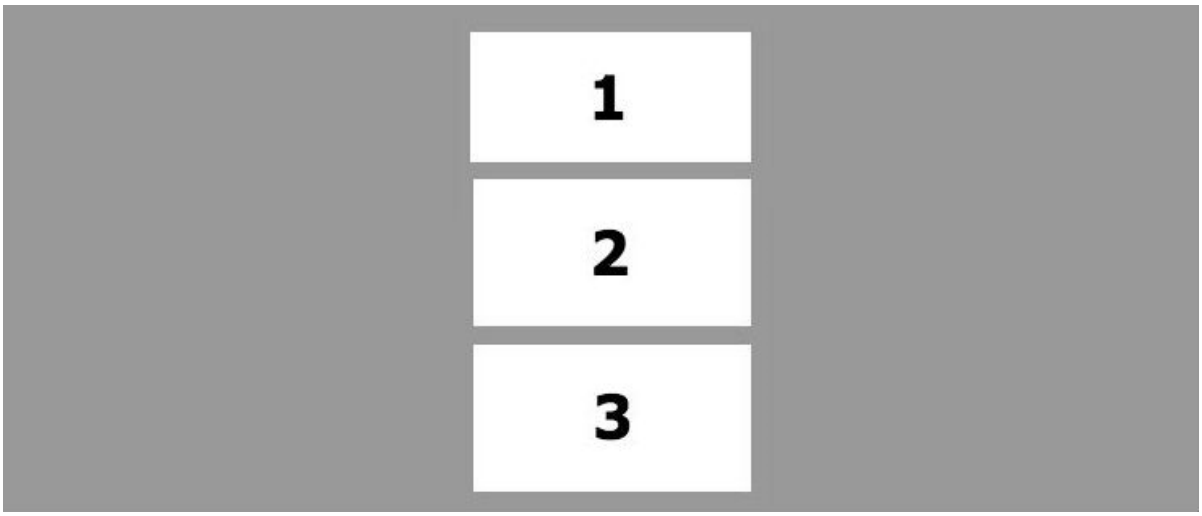
El contenedor se vería de la siguiente manera:



Como podemos ver en la sintaxis, su valor está establecido en row (que es su valor por defecto), pero también podemos cambiarlo a column.

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

Ahora el contenedor se vería de la siguiente manera:



ORDEN DE LOS ELEMENTOS

Con Flexbox podemos establecer el orden de los elementos de un contenedor flex. Para ello utilizamos la propiedad order en cada uno de los elementos. Por ejemplo:

```
.container-item {
  order: 1;
}
```

Por defecto, todos los ítems flex tienen un order: 0 implícito, aunque no se especifique. Si indicamos un order con un valor numérico, irá colocando los ítems según su número, colocando antes los ítems con número más pequeño y después los ítems con números más altos.

PROPIEDAD FLEX-WRAP

La propiedad flex-wrap de CSS especifica si los elementos hijos son obligados a permanecer en una misma línea o pueden fluir en varias líneas. La propiedad flex-wrap puede tomar uno de estos valores:

- **nowrap** (el valor por defecto): los elementos flex aparecen en una sola línea.

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

Figura 13.1 A la izquierda valor nowrap, a la derecha valor wrap.

Por otro lado, tenemos la propiedad flex-flow que combina las propiedades flex-direction y flex-wrap. El valor por defecto es row no-wrap.

```
flex-flow: row no-wrap;
```

Ahora realizaremos un pequeño ejercicio para aplicar lo que recién vimos:

```
<section class="flexContainer">
        <div class="item1">
                <h2>Lorem ipsum.</h2>
                <p>Lorem ipsum dolor sit amet, consectetur adipisicing.</p>
        </div>
        <div class="item2">
                <h2>Lorem ipsum dolor.</h2>
                <p>Lorem ipsum dolor sit amet, consectetur.</p>
        </div>
         <div class="item3">
                <h2>Lorem ipsum dolor sit.</h2>
                <p>Lorem ipsum dolor sit amet, consectetur adipisicing
                   elit.</p>
        </div>
</section>
```

Esta pequeña estructura HTML permitirá armar un section contenedor con tres ítems en su interior. Luego aplicaremos las CSS:

```
.flexContainer {
  display: flex;
  flex-flow: row wrap;
}
.item1 {
  width: 400px;
  background:#6ABD27;
  order: 2;
}
.item2 {
  width: 400px;
  background:#00A6EE;
  order: 3;
}
.item3 {
  width: 400px;
  background:#6ABD27;
  order: 1;
}
```

En este ejemplo el contenedor .flexContainer usa la propiedad flex-flow para indicar que en la orientación principal los elementos se acomodaran uno al lado del otro de izquierda a derecha (valor row), también con el valor wrap indicamos que cuando el tamaño del contenedor flexible se reduce, los elementos flexibles se mostrarán en filas diferentes. También usamos la propiedad order, para que los elementos flexibles no se muestran en el mismo orden que se especifica en el documento de origen.
Ahora modificaremos la apariencia controlando el alineamiento de los elementos a lo largo de su eje principal con la propiedad justify-content. Si colocamos el valor center: los elementos aparecen centrados.

```
.flexContainer {
  display: flex;
  flex-flow: row wrap;
  justify-content: center;
}
```

Si ponemos el valor flex-start los elementos se alinearán a la izquierda del contenedor, si colocamos el valor flex-end los elementos se alinearán al lado derecho del contenedor. Con el valor space-between los elementos tienen la misma distancia entre ellos, pero el primer y el último elemento están alineados con los bordes del contenedor. Finalmente, con el valor space-around, los elementos tienen la misma distancia entre ellos, incluso el primero y el último elemento.
Por otro lado, tenemos la propiedad align-items, a diferencia de la anterior este modifica la alineación sobre el eje secundario. El valor por defecto de esta propiedad es stretch, que hace que los flex-items ocupen todo el espacio disponible sobre el eje secundario. Si deseamos que centren sobre dicho eje deberemos usar el valor center. En el caso de que tomen el valor flex-start se apilan al inicio del eje secundario y si toman el valor flex-end los elementos se apilan al final del eje secundario.

TAMAÑO BASE PARA LOS ITEMS

Con la propiedad flex-basis (tamaño base) podemos establecer el ancho inicial de los elementos de un contenedor antes de que el espacio se reparta entre los elementos (o sea no es el ancho definitivo). Se puede usar porcentaje, pixeles o cualquier otra unidad. Si el main axis es horizontal (predeterminado), flex-basis será equivalente a width; y si el main axis es vertical, flex-basis será equivalente a height.
Hay que tener en cuenta que el tamaño definido por flex-basis es, como su nombre lo dice, el tamaño base. Es decir, que podrá variar (crecer o encogerse).

Nota

Si no se define el valor de flex-basis o se establece en auto, se tomará en cuenta el valor de width o height según sea el caso.

PROPIEDAD FLEX-GROW

Con esta propiedad, podemos controlar cómo deberían crecer los elementos cuando hay espacio adicional disponible. El valor predeterminado, si no se especifica lo contrario, es 0. Eso significa que los elementos no crecerán para llenar el espacio adicional en el eje principal de forma predeterminada. Sin embargo, esto se puede cambiar asignando cualquier valor numérico mayor que 0 (No se permiten valores negativos). Digamos que le asignamos el valor 1 a todos los elementos. Esto significa que el espacio adicional se asignará proporcionalmente a cada elemento de manera que cada elemento obtenga la misma cantidad de espacio adicional. Esto se debe a que los valores son los mismos.

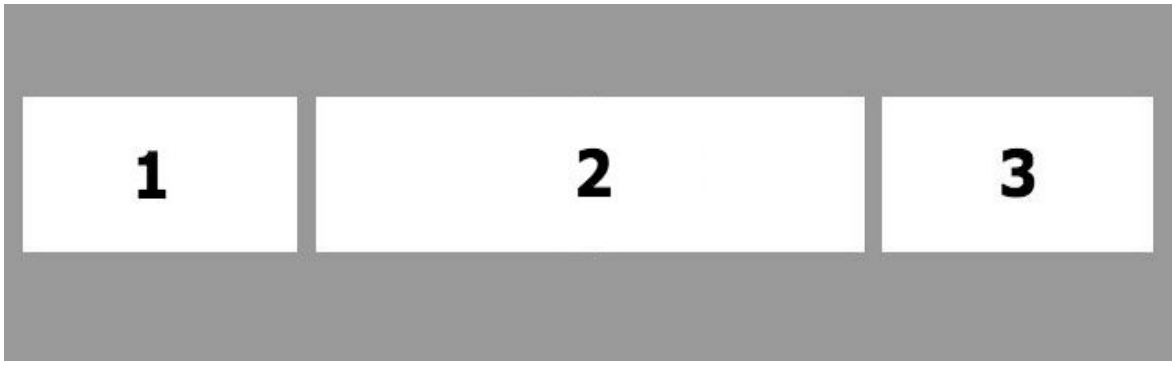
Ejemplo:

```
.container {
    display: flex;
}

.item1,.item3 {
    flex-grow: 1;
}

.item2 {
    flex-grow: 2;
}
```

Este es el resultado del ejercicio:



GAPS

Existen dos propiedades de flexbox que han surgido recientemente: row-gap y column-gap. Dichas propiedades, permiten establecer el tamaño de huecos (gaps) entre ítems que debe ser definido desde el elemento padre contenedor, y sin necesidad de estar utilizando padding o margin en los elementos hijos. Hay que tener en cuenta que sólo una de las dos propiedades tendrá efecto, dependiendo de si la propiedad flex-direction está establecida en column o en row. Por ello row-gap define el espacio solo si el flex-direction es column, column-gap define el espacio entre columnas solo si el flex-direction es row.

CENTRADO ABSOLUTO CON FLEXBOX

Una de las mayores dificultades del CSS tradicional ha sido el centrado absoluto es decir centrar vertical y horizontalmente. Con flexbox una gran cantidad de código ha desaparecido ya que solo basta aplicar las siguientes propiedades con estos valores:

```
.flexContainer {
  display: flex;
  justify-content: center;
  align-items: center;
}
```