

Investigación grid y flexbox

**Nombre: Eduardo José Collado Barrera**

**Carné: 1590-19-23490**

**Carrera: Ingeniería en Sistemas**

**Sección: “A”**

**Curso: Desarrollo Web**

**Fecha: 12/08/2022**

**FLEXBOX**

Dado que tradicionalmente en Css se ha usado el posicionamiento: static, relative, absolute... los elementos en línea (o en bloque *y derivados*) o los float, y que a grandes rasgos no dejaba de ser un sistema de creación de diseños bastante tradicional que no encaja con los retos que tenemos hoy en día: sistemas de escritorio, dispositivos móviles, múltiples resoluciones.

Flexbox surgió como un sistema de elementos flexibles que llega con la idea de olvidar estos mecanismos y acostumbrarnos a una mecánica más limpia, potente y personalizable; en la que los elementos HTML se adaptan y colocan automáticamente, y es fácil personalizar los diseños. Especialmente en estructuras de una sola dimensión.

Este modelo es empleado para evitar el uso de float o posicionamiento.

Al momento de hacer uso de Flexbox, lo primero que debemos hacer es conocer algunos de los elementos básicos de este, los cuales son:

* **Contenedor**: Este es el que tendrá cada uno de los ítems flexibles, por norma general, en este se establecen las propiedades de cada elemento de flexbox
* **Eje Principal**: Este es horizontal por defecto
* **Eje Secundario**: Este por defecto es vertical
* **Ítem**: Es cada uno de los hijos que tendrá el contenedor en su interior

**Una estructura sobre el uso de flex**

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

Para activar el modo flexbox, se debe hacer uso de la propiedad **Display** y especificar el valor, ya sea: **flex o inline-flex** (esto va dependiendo de como queramos que se comporte el contenedor)

**Módulos de diseño de Css flexbox:**

* **Block**: Este es empleado para secciones de una página web
* **Inline**: Generalmente usado para texto.
* **Table**: Para datos de tablas bidimensionales.
* **Positioned**: Para la posición implícita de un elemento.

**Para dirección de los ejes**

Flexbox hace uso de propiedades como:

Flex-direction: La cual cambia la orientación del eje principal, en donde se hace uso de:

* Row: Este establece la dirección del eje principal en horizontal
* Row-reverse: Este establece la dirección del eje principal en horizontal invertido
* Column: Este establece la dirección del eje principal en vertical
* Column-reverse: Este establece la dirección del eje principal en vertical invertido

**Para contenido multimedia**

Se hace uso de Flex-wrap; el cual evita el desbordamiento multilínea, y lleva como atributos:

* Nowrap: Establece los ítems en una sola línea (no permite que se desborde el contenedor)
* Wrap: Establece los ítems en modo multilínea (permite que se desborde el contenedor)
* Wrap-reverse: Establece los ítems en modo multilínea, pero en dirección inversa.

**Para alineación**

justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly. Eje: 1️

align-content: flex-start | flex-end | center | space-between | space-around | space-evenly | stretch.Eje: 2️

align-items: flex-start | flex-end | center | stretch | baseline. Eje: 2️

align-self: auto | flex-start | flex-end | center | stretch | baseline. Eje: 2️

**Para eje Principal**

flex-start: Agrupa los ítems al principio del eje principal.

flex-end: Agrupa los ítems al final del eje principal.

Center: Agrupa los ítems al centro del eje principal.

space-between: Distribuye los ítems dejando el máximo espacio para separarlos.

space-around: Distribuye los ítems dejando el mismo espacio alrededor de ellos (izq/dcha).

space-evenly: Distribuye los ítems dejando el mismo espacio (solapado) a izquierda y derecha.

**Para eje Secundario**

flex-start: Alinea los ítems al principio del eje secundario.

flex-end: Alinea los ítems al final del eje secundario.

Center: Alinea los ítems al centro del eje secundario.

Stretch: Alinea los ítems estirándolos de modo que cubran desde el inicio hasta el final del contenedor.

Baseline: Alinea los ítems en el contenedor según la base del contenido de los ítems del contenedor.

**GRID**

Dado que el posicionamiento es uno de los factores más problemáticos de CSS, especialmente para novatos o principiantes; CSS Grid recoge las ventajas del sistema que permite crear rápidamente cuadrículas sencillas y potentes de forma prácticamente instantánea.

Como comentario, destacamos que el sistema FLEXBOX es una de las bases para que naciera CSS grid.

**Grid está compuesta por:**

* **Contenedor**: En este se definirá la rejilla o cuadrícula que contendrá todos los ítems
* **Ítem**: Es cada uno de los hijos que contiende la cuadrícula (contenedor)
* **Celda (grid cell)**: Cada uno de los cuadritos de la cuadrícula
* **Area (grid area)**: Región o conjunto de celdas de la cuadrícula
* **Banda (grid track)**: Banda horizontal o vertical de las celdas de la cuadrícula
* **Línea (grid line)**: Separador horizontal o vertical de las celdas de la cuadrícula

**Para activar grid**

Es necesario que ese utilice en el elemento contenedor la propiedad display y definir:

inline-grid: Establece una cuadrícula con ítems en línea, de forma equivalente a inline-block.

grid: Establece una cuadrícula con ítems en bloque, de forma equivalente a block.

**Grid con columnas y filas**

grid-template-columns: [col1] [col2] ... Establece el de cada columna (col 1, col 2...).

grid-template-rows: [fila1] [fila2] ... Establece el de cada fila (fila 1, fila 2...).