UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA EN COMPUTACIÓN

INGENIERA:

Guillermo de Jesús Calderón Hernández

Tema:

Tarea Ex Aula 01

ASIGNATURA:

Lenguajes Interpretados en el Cliente G01L

FECHA DE ELABORACION: 09/09/2023

CICLO 01 / 2023

No.	INTEGRANTES:	CARNET
1	Castellanos Romero, Daniel Alexander	CR221376
2	Menjivar Lemus, José Daniel	ML210413
3	Ramirez Garcia, Francisco Samuel	RG221033

Nota: _____

Flexbox

Flexbox es un módulo de diseño CSS que proporciona una forma más eficiente de diseñar, alinear y distribuir el espacio entre los elementos de un contenedor, incluso cuando su tamaño es desconocido y/o dinámico. Flexbox es un método de diseño unidimensional para organizar elementos en filas o columnas, y permite que los elementos se flexionen (expandan) para llenar espacio adicional o se encojan para caber en espacios más pequeños.[13].

Algunos de los conceptos clave en Flexbox incluyen el eje principal y el eje transversal. El eje principal es el eje principal a lo largo del cual se colocan los elementos flexibles y está determinado por la propiedad de dirección de flexión. El eje transversal es perpendicular al eje principal y se utiliza para la alineación y distribución de elementos.[12]

Los dos ejes de flexbox

Cuando trabajamos con flexbox necesitamos pensar en términos de dos ejes — el eje principal y el eje cruzado. El eje principal está definido por la propiedad flex-direction, y el eje cruzado es perpendicular a este. Todo lo que hacemos con flexbox está referido a estos dos ejes, por lo que vale la pena entender cómo trabajan desde el principio.

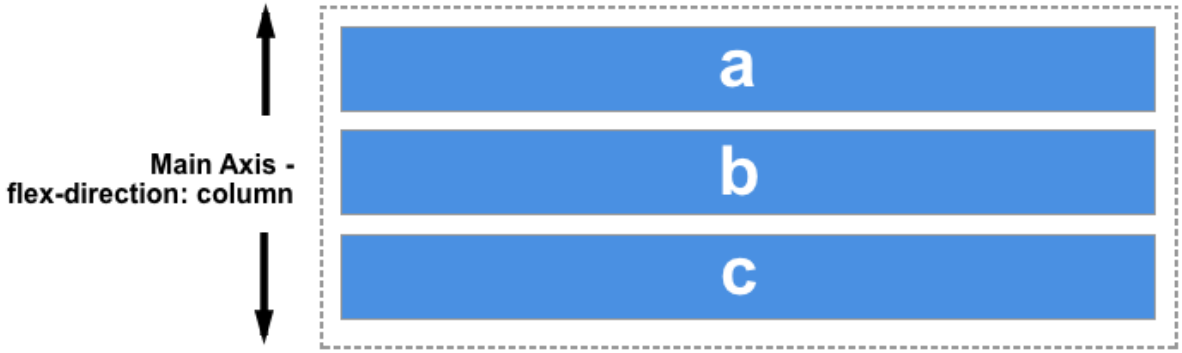
El eje principal

El eje principal está definido por flex-direction, que posee cuatro posibles valores:

- row
- row-reverse
- column
- column-reverse

Si elegimos row o row-reverse, el eje principal correrá a lo largo de la fila según la **dirección de la línea**.

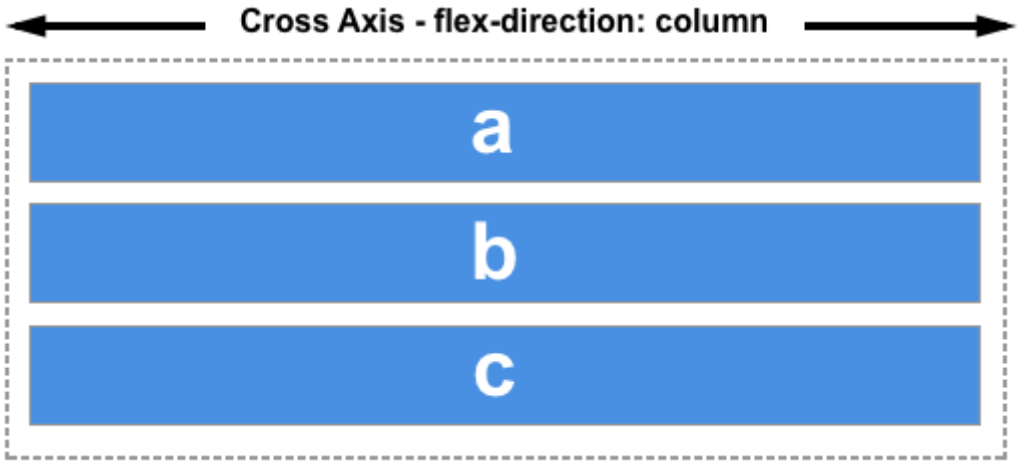
Al elegir column o column-reverse el eje principal correrá desde el borde superior de la página hasta el final — según la **dirección del bloque**.



El eje cruzado

El eje cruzado va perpendicular al eje principal, y por lo tanto si flex-direction (del eje principal) es row o row-reverse el eje cruzado irá por las columnas.

Si el eje principal es column o column-reverse entonces el eje cruzado corre a lo largo de las filas.



Entender cuál eje es cuál es importante cuando empezamos a mirar la alineación y justificación flexible de los ítems; flexbox posee propiedades que permiten alinear y justificar el contenido sobre un eje o el otro.

Líneas de inicio y de fin

Otra área vital de comprensión es cómo flexbox no hace suposiciones sobre la manera de escribir el documento. En el pasado, CSS estaba muy inclinado hacia el modo de escritura horizontal y de izquierda a derecha. Los métodos modernos de diseño acogen la totalidad de modos de escritura así que no es necesario asumir que una línea de texto empezará arriba del documento y correrá de izquierda a derecha, con nuevas líneas dispuestas una abajo de la otra.

Puede leer más acerca de la relación que hay entre flexbox y la especificación de los Modos de Escritura en un artículo posterior, sin embargo la siguiente descripción debería ayudar para explicar porqué no se habla de izquierda y derecha ni de arriba o abajo a la hora de Describa la dirección en la que fluyen los artículos flex.

Si flex-directiones rowy estoy trabajando en español, entonces el margen inicial del eje principal quedará a la izquierda, y el margen final a la derecha.

Si fuera a trabajar en árabe, entonces el margen inicial de mi eje principal quedaría a la derecha y el margen final a la izquierda.

En ambos casos el margen inicial del eje cruzado estará en el extremo superior del contenedor flex y el margen final en el extremo inferior, ya que ambos idiomas tienen un modo de escritura horizontal.

Después de un tiempo, pensar en inicial y final en vez de izquierda y derecha se hará natural, y será útil cuando interactúe con otros métodos de diseño tales como el CSS Grid Layout que sigue los mismos patrones.

El contenedor flexible

Un área del documento que contiene un flexbox es llamada **contendiente flex**. Para crear un contenedor flex, establecemos la propiedad del área del contenedor displaycomo flexo inline-flex. Tan pronto como hacemos esto, los hijos directos de este contenedor se vuelven **ítems flex** . Como con todas las propiedades de CSS, se definen algunos valores iniciales, así que cuando creemos un contenedor flex todos los ítems flex contenidos se comportarán de la siguiente manera.

- Los artículos se despliegan sobre una fila (la propiedad flex-directionpor defecto es row).
- Los artículos empiezan desde el margen inicial sobre el eje principal.
- Los artículos no se ajustan en la dimensión principal, pero se pueden contraer.
- Los artículos se ajustarán para llenar el tamaño del eje cruzado.
- La propiedad flex-basisestá definida como auto.
- La propiedad flex-wrapestá definida como nowrap.

El resultado es que todos los artículos se alinearán en una sola fila, usando el tamaño del contenedor como su tamaño en el eje principal. Si hay más elementos de los que caben en el contenedor, estos no pasarán más abajo si no que sobrepasarán el margen. Si hay elementos más altos que otros, todos los elementos serán ajustados en el eje cruzado para alcanzar al mayor.

Se puede ver en el ejercicio en vivo de abajo cómo luce. Intente editar el artículo o agregar artículos adicionales para así probar el comportamiento inicial de flexbox.

Cambiar dirección flexible

Al agregar la propiedad flex-directionen el contenedor flex nos permite cambiar la dirección de cómo los artículos son desplegados. Colocando flex-direction: row-reversese mantendrá el despliegue a lo largo de la fila, sin embargo el inicio y final quedarán al revés del original.

Si cambiamos flex-directional columneje principal se cambiará y los elementos aparecerán en una columna. Colocando column-reverselas líneas de inicio y fin serán nuevamente puestas al revés.

El ejemplo en vivo de abajo tiene flex-directionpuesto como row-reverse. Pruebe los otros valores — row, columny column-reverse— para ver qué sucede con el contenido.

Contenedores flex Multi-línea con flex-wrap

Si bien flexbox es un modelo unidimensional, es posible lograr que nuestros artículos flex sean repartidos en varías líneas. Haciendo esto, se deberá considerar cada línea como un nuevo contenedor flex. Cualquier distribución del espacio solo sucederá dentro de esa línea, sin referenciar las líneas colaterales.

Para lograr repartir en varias líneas añada la propiedad flex-wrapcon el valor wrap. Cuando los elementos sean demasiados para desplegarlos en una línea, serán repartidos en la línea siguiente. El ejemplo en vivo de abajo contiene ítems que se les ha asignado un ancho, donde el ancho total de los ítems excede al del contenedor flex. Cuando flex-wrapse coloca como wrap,

los artículos se repartirán. Al colocarlo como nowrap, el cual es el valor inicial, estos se contraerán para calzar con el contenedor ya que usan los valores iniciales de flexbox que permiten que los artículos se contraigan. Al usar nowraplos ítems podrían salirse del margen si estos no pudieran contraerse, o no contraerse lo suficiente para ser calzados.

La abreviatura flex-flow

Se pueden combinar las propiedades flex-directiony flex-wrapen la abreviatura flex-flow. El primer valor especificado es flex-directiony el segundo valor es flex-wrap.

En el ejemplo en vivo de abajo intente cambiar el primer valor por uno de los valores permitidos para flex-direction- row, row-reverse, columno column-reverse, y cambie también el segundo valor por wrapy nowrap.

Propiedades aplicadas a los artículos flex

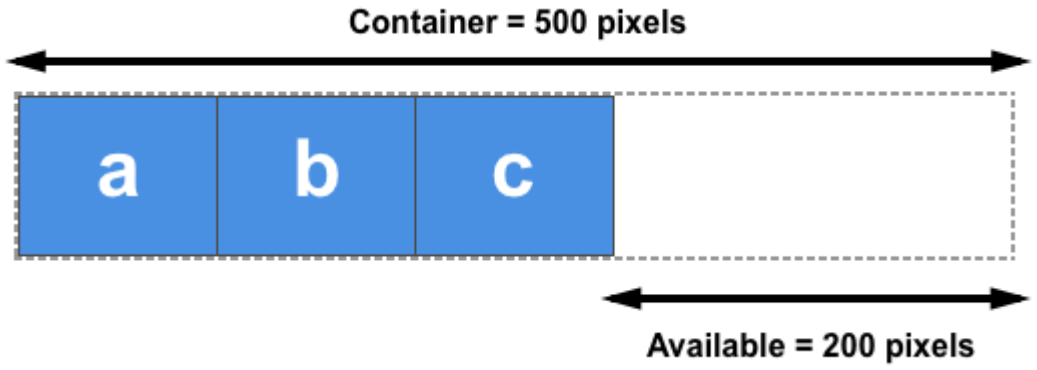
Para obtener más control sobre los elementos flex podemos apuntarlos directamente. Hacemos esto a través de tres propiedades:

- flex-grow
- flex-shrink
- flex-basis

Daremos un breve vistazo a estas propiedades en este resumen, y en un próximo artículo ahondaremos sobre su comportamiento.

Antes de darle sentido a estas propiedades debemos considerar el concepto de **espacio disponible** . Lo que hacemos cuando cambiamos el valor de alguna de estas propiedades es cambiar la forma que se distribuye el espacio disponible entre nuestros artículos. Este concepto de espacio disponible también es importante cuando veamos la alineación de elementos.

Si tenemos tres artículos con un ancho de 100 píxeles en un contenedor de 500 píxeles de ancho, entonces el espacio que se necesita para colocar nuestros artículos es de 300 píxeles. Esto deja 200 píxeles de espacio disponible. Si no cambiamos los valores iniciales entonces flexbox colocará ese espacio después del último artículo.



Si en cambio quisiéramos que los ítems crecieran para llenar ese espacio, entonces necesitaremos un método para distribuir el espacio sobrante entre los ítems. Es justo lo que hará las propiedades flexque aplicaremos a dichos artículos.

la propiedadflex-basis

Con flex-basisse define el tamaño de un elemento en términos del espacio que deja como espacio disponible. El valor inicial de esta propiedad es auto— en este caso el navegador revisa si los elementos definen un tamaño. En el ejemplo de arriba, todos los artículos tienen un ancho de 100 píxeles así que este se usa como flex-basis.

Si los artículos no tienen un tamaño, entonces el tamaño de su contenido se usa como base flexible. Y por eso, apenas declarado display: flexen el padre a fin de crear ítems flex, todos estos ítems se ubicaron en una sola fila y tomaron solo el espacio necesario para desplegar su contenido.

la propiedadflex-grow

Con la propiedad flex-growdefinida como un entero positivo, los artículos flexibles pueden crecer en el eje principal a partir de flex-basis. Esto hará que el artículo se ajuste y tome todo el espacio disponible del eje, o una proporción del espacio disponible si otro artículo también puede crecer.

Si le damos a todos los ítems del ejemplo anterior un valor flex-growde 1 entonces el espacio disponible en el contenedor flex será compartido igualitariamente entre estos ítems y se ajustarán para llenar el contenedor sobre el eje principal.

Podemos usar flex-grow apropiadamente para distribuir el espacio en proporciones. Si otorgamos al primer ítem un valor flex-growde 2 ya los otros un valor de 1, entonces 2 partes serán dadas al primer ítem (100px de 200px en el caso del ejemplo de arriba) y 1 parte para cada uno de los restantes (cada uno con 50px de los 200px en total).

la propiedadflex-shrink

Así como la propiedad flex-growse encarga de agregar espacio sobre el eje principal, la propiedad flex-shrinkcontrola como se contrae. Si no contamos con suficiente espacio en el contenedor para colocar los artículos y flex-shrinkposee un valor entero positivo, el artículo puede contraerse a partir de flex-basis. Así como podemos asignar diferentes valores de flex-growcon el fin que un ítem se expandirá más rápido que otros — un ítem con un valor más alto de flex-shrinkse contraerá más rápido que sus hermanos que poseen valores menores.

El tamaño mínimo del ítem tendrá que ser considerado cuando se determine un valor de contracción que pueda funcionar, esto significa que flex-shrink tiene el potencial de comportarse menos consistentemente que flex-grow. Por lo tanto, haremos una revisión más detallada de cómo funciona este algoritmo en el artículo Controlling Ratios de los ítems sobre el eje principal.

Grid

CSS Grid es un sistema de diseño bidimensional que permite a los desarrolladores crear diseños complejos y receptivos. Proporciona capacidades de diseño bidimensional, lo que permite organizar los elementos horizontal y verticalmente. Por otro lado, CSS Flexbox aporta flexibilidad, permitiendo colocar elementos en un solo eje, ya sea horizontal o vertical. CSS Grid está basado en contenedores y funciona a nivel de diseño, mientras que Flexbox está basado en contenido y se ajusta según el contenido. CSS Grid es más adecuado para diseños más complicados, mientras que Flexbox es ideal para diseños más simples.Los siguientes son los conceptos clave de CSS Grid:

- Contenedor de cuadrícula: el elemento principal que contiene elementos de la cuadrícula.
- Elemento de cuadrícula: el elemento secundario de un contenedor de cuadrícula.
- Línea de cuadrícula: La línea horizontal o vertical que separa las filas y columnas de la cuadrícula.
- Pista de cuadrícula: el espacio entre dos líneas de cuadrícula adyacentes.
- Área de cuadrícula: el espacio entre cuatro líneas de cuadrícula.

Las siguientes son las propiedades clave de CSS Grid:

- grid-template-columns: define el número y tamaño de las columnas de la cuadrícula.
- grid-template-rows: define el número y tamaño de las filas en la cuadrícula.
- grid-template-areas: define áreas de cuadrícula con nombre.
- grid-gap: define el tamaño del espacio entre las filas y columnas de la cuadrícula.
- columna de cuadrícula: especifica las posiciones inicial y final de un elemento de cuadrícula.
- fila de cuadrícula: especifica las posiciones inicial y final de un elemento de cuadrícula.
- justify-items: alinea los elementos de la cuadrícula a lo largo del eje horizontal.
- align-items: Alinea los elementos de la cuadrícula a lo largo del eje vertical.
- colocar elementos: alinea los elementos de la cuadrícula a lo largo de los ejes horizontal y vertical.
- justify-content: Alinea la cuadrícula a lo largo del eje horizontal.
- align-content: Alinea la cuadrícula a lo largo del eje vertical.
- place-content: Alinea la cuadrícula a lo largo de los ejes horizontal y vertical.