

**TECNOLÓGICO NACIONAL DE MÉXICO**

**INSTITUTO TECNOLOGICO DE MERIDA**

**INGENIERÍA EN SISTEMAS COMPUTACIONALES**



**DESARROLLO FRONTEND**

**PROFESOR: Héctor Jesús Cetina Cordero**

**Investigacion DISPLAY:FLEX y sus propiedades en la maquetacion**

**Unidad 2**

**Grupo: 9SA**

**ALUMNO(A)**

* **Alcala Morales Aldo Gaspar**

**Fecha: 16/10/2019**

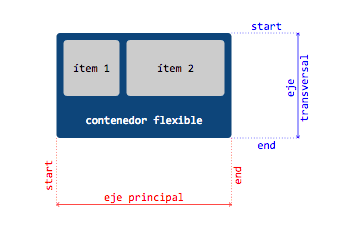
Realiza la investigación de cada una de las propiedades y valores que tiene el DISPLAY:FLEX y describe para que sirve cada una de ellas.

**Flex-direction**

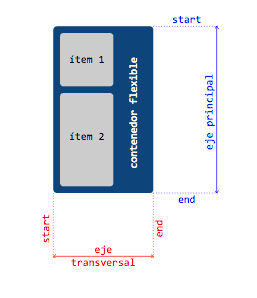
La propiedad flex-direction es una propiedad del contenedor Flex.

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:

SINTAXIS

.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 ).

Se aceptan los siguientes valores:

**row**

El eje principal del contenedor flexible está definido para ser el mismo que la dirección del texto. Los **puntos principales de inicio y final** son los mismos que la dirección del contenido.

**row-reverse**

Se comporta igual que row pero los **puntos principales de inicio** y **final** son intercambiados.

**column**

El eje principal del contenedor flexible es el mismo que el eje del bloque. Los **puntos principales de inicio y final** son los mismos que los **puntos de antes y después** del modo escritura.

**column-reverse**

Se comporta igual que row pero los **puntos principales de inicio** y **final** son intercambiados.

Sobre Accesibilidad

Si utilizas flex-direction con un valor de row-reverse o column-reverse en elementos que necesitan foco (como botones) el orden de visualización será distinto al orden el DOM, por lo que los usuarios de lectores de pantalla no verán reflejado el mismo orden de los elementos que un usuario vidente.

**FLEX-WRAP**

.flex-container {

flex-wrap: wrap || nowrap || wrap-reverse;

}

Se usa para indicar cómo se distribuyen los elementos en la fila y cómo afecta esto a su tamaño.

Como ejemplo, vamos a aumentar los elementos dentro del contenedor:

<div class="flex-container">

<div class="flex-item">1</div>

<div class="flex-item">2</div>

<div class="flex-item">3</div>

<div class="flex-item">4</div>

<div class="flex-item">5</div>

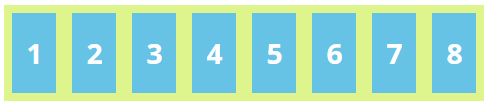
<div class="flex-item">6</div>

<div class="flex-item">7</div>

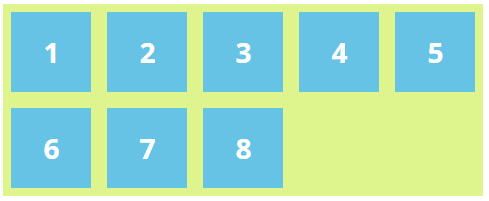
<div class="flex-item">8</div>

</div>

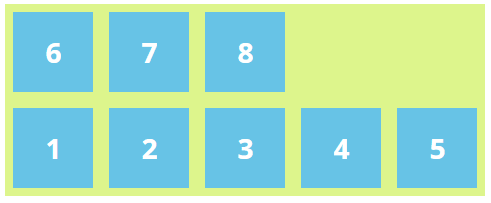
* **Nowrap**: Valor por defecto. Los elementos siempre se muestran en la misma línea, el eje principal. Por lo tanto, tengamos los elementos que tengamos, el tamaño del contenedor no aumentará e intentará acomodarlos a todo el eje principal.



* **Wrap**: No ajusta el tamaño del elemento y permite que puedan ocupar varias líneas. Si tenemos muchos elementos, se colocarán a lo largo del eje y, cuando no quede más espacio, pasarán a la siguiente fila.



* **Wrap-reverse**: Similar a wrap pero, como su nombre indica, en orden inverso. Se colocan todos los elementos en fila, de forma inversa, y cuando no queda espacio, pasan a la siguiente línea, sin alterar su tamaño.



Se aceptan los siguientes valores:

**nowrap**

Los elementos flex son distribuidos en una sola línea, lo cual puede llevar a que se desborde el contenedor flex. El valor **cross-start** es equivalente a **start** o **before** según el valor de [flex-direction](https://developer.mozilla.org/es/docs/Web/CSS/flex-direction).

**wrap**

Los elementos flex son colocados en varias líneas. El valor **cross-start** equivale a **start** o **before** dependiendo del valor flex-direction y **cross-end** implicaría lo opuesto a lo especificado por**cross-start**.

**wrap-verse**

Actúa como wrap pero **cross-start** y **cross-end** están intercambiados.

**Justify-content**

Posiciona los elementos, y los valores que puede tener son:

.flex-container {

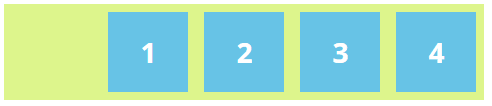
justify-content: flex-start || flex-end || center || space-between || space-around

}

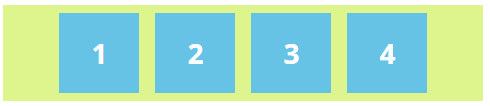
* **Flex-start**: Valor por defecto. Todos los elementos comienzan en el main-start (inicio principal).



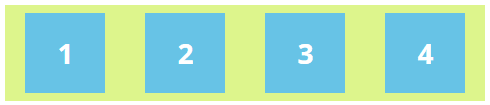
* **Flex-end**: Posicionan los elementos desde el main-end (el otro extremo, el final).



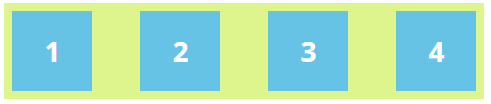
* **Center**: Los posiciona en el centro.



* **Space-around**: Posiciona los elementos con un espacio alrededor del mismo tamaño.



* **Space-between**: Los posiciona con el mismo tamaño entre cada elemento.



**Align-items**

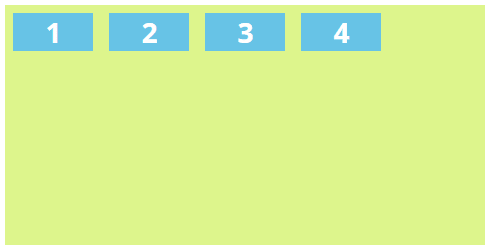
Esta propiedad es similar a justify-content, pero la diferencia es el eje a lo largo del cual funciona. Para explicar esto, deberemos quitar la propiedad que define la altura de los elementos hijos, y aumentar la del elemento padre:

.flex-container {

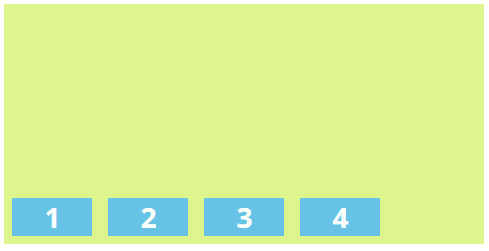
align-items: flex-start || flex-end || center || stretch || baseline;

}

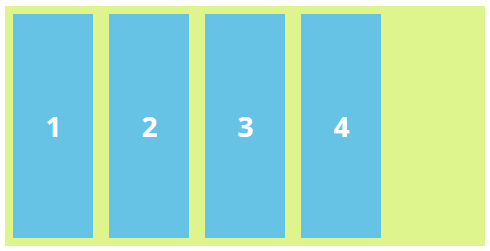
* Flex-start: Posiciona los elementos desde el inicio del contenedor, es decir, a lo largo del eje transversal.



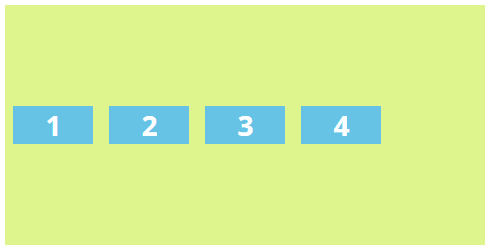
* Flex-end: Los posiciona al final del contenedor, a lo largo del eje transversal.



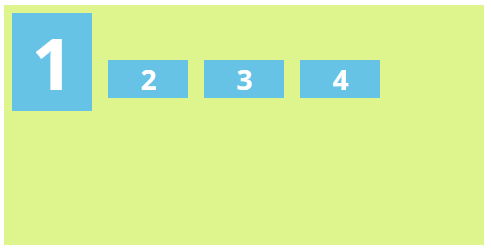
* Stretch: Valor por defecto. Extiende los elementos desde el principio hasta el final a lo largo del eje transversal.



* Center: Centra los elementos a lo largo del eje.



* Baseline: Aparentemente, la baseline se parece a flex-start, pero con la diferencia de que la línea que posiciona los elementos es la primera línea de texto del primer elemento. Para que se note la diferencia, hemos aumentado el tamaño de la fuente del primer elemento.



**Align-content**

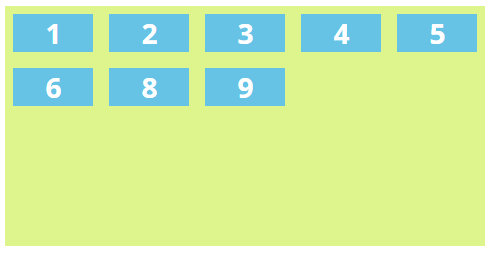
Gestiona cómo se alinean los elementos dentro del contenedor, con múltiples líneas. La diferencia con align-items es que el primero alinea elementos, y align-content las filas. Además, align-self, alinea los elementos de forma independiente.

.flex-container {

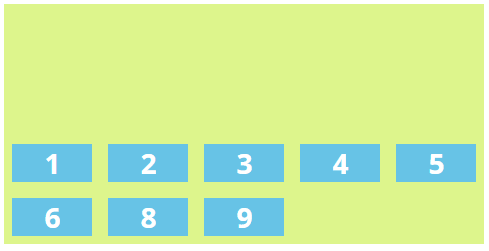
align-content : flex-start || flex-end || center || stretch;

}

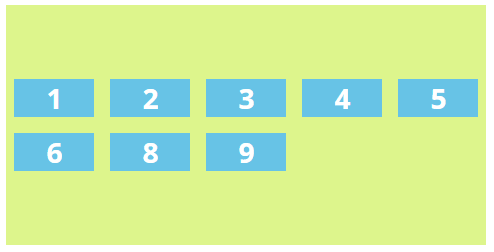
* **Flex-start**: Aliena los elementos a lo largo del inicio del eje transversal.



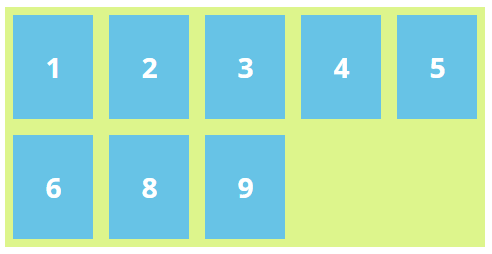
* **Flex-end**: Alinea los elementos a lo largo del extremo del eje transversal.



* **Center**: Los alinea en el centro del eje.



* **Stretch**: Valor por defecto. Estira los elementos a lo largo del eje transversal de principio a fin.



REFERENCIAS

<http://w3.unpocodetodo.info/css3/flex-direction.php>

<https://developer.mozilla.org/es/docs/Web/CSS/flex-direction>

<https://webappdesign.es/guia-flexbox/>

<https://developer.mozilla.org/es/docs/Web/CSS/flex-wrap>