
ICARO
Nuestro compromiso es con el futuro.

Introducción al Front End

Clase 3

¿Qué haremos hoy?

Hoy vamos a continuar viendo algunas etiquetas de **HTML** relacionadas con la inyección de contenido a nuestro sitio web y también con la interacción con el mismo. Esto nos preparará para poder reaccionar ante las acciones del usuario en próximas clases.

Otra cosa que vamos a aprender hoy es una de las herramientas más poderosas que nos ofrece **CSS**: **Flexbox**. Flexbox es un valor de la propiedad `display` que nos permite crear **contenedores flexibles**. Esto es de gran utilidad ya que nos permitirá crear **interfaces complejas** y **avanzadas** con gran facilidad. Veremos cómo aplicar **Flexbox** en nuestras páginas, qué propiedades tiene y cuándo nos conviene utilizarlo. Todo esto mientras vamos viendo poco a poco *cómo* ir implementando todas estas cuestiones.

Etiquetas de HTML

```
<iframe src="xxxx"></iframe>
```

Esta etiqueta, llamada Iframe o Marco, sirve para inyectar un HTML dentro de otro. Cuenta con una propiedad src o source, que es donde le indicaremos qué archivo HTML es el que debe inyectar o colocar como su contenido.

Esta etiqueta, además, permitía crear una navegación entre otras páginas de nuestro sitio. Aunque hoy por hoy ha caído en desuso no sería raro encontrar esta etiqueta en algún código.

Etiquetas de HTML

```
<button type="xxxx"> xxxx </button>
```

Esta etiqueta, llamada Button o Botón, sirve para representar un elemento con el que podemos interactuar a través del click de nuestro mouse. Cuenta con una propiedad type que es donde le indicaremos que tipo de comportamiento queremos que tenga este botón una vez que se le haga un click.

Esto es muy importante para los formularios (concepto que veremos más en profundidad en clases siguientes), elementos clave en cualquier sitio web ya que son la vía de entrada a casi la totalidad de la interacción entre el usuario y nuestro sitio.

Etiquetas de HTML

```
<video controls width="xxxx" height="xxxx">
    <source src="xxxx.mp4" type="video/mp4">
    <source src="xxxx.ogg" type="video/ogg">
</video>
```

Esta etiqueta, llamada Video, sirve para representar los videos. Cuenta con etiquetas interiores llamadas source que son las que contendrán las distintas fuentes de donde puede proveer nuestro video.

La etiqueta video lleva por lo general una propiedad controls que nos permitirá utilizar los controles del video cuando este se inyecte en el HTML, además de los atributos height y width para el tamaño.

Flexbox

Como vimos recién, **Flexbox** nos permite crear interfaces avanzadas gracias a que al definir un contenedor flexible, sus **elementos hijos** pueden ser ordenados por este contenedor.

Es fundamental que comprendamos que en **Flexbox** es el **elemento padre** quien ordenará a sus **elementos hijos** y que solamente lo hará con aquellos elementos que estén inmediatamente por dentro del mismo.

Flexbox

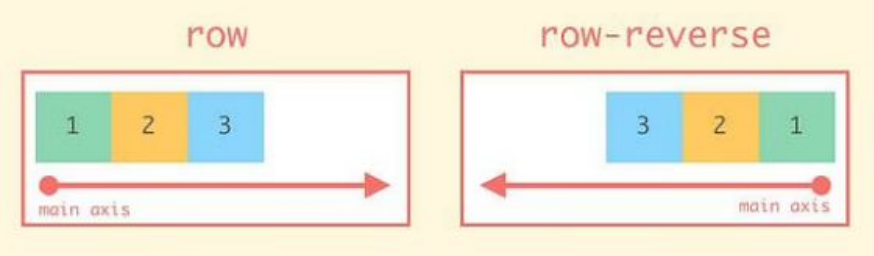
En **Flexbox** lo principal a definir es cómo el **elemento padre** ordena a sus **elementos hijos**.
Lo hace a través de dos ejes, uno **principal** y uno **secundario**.

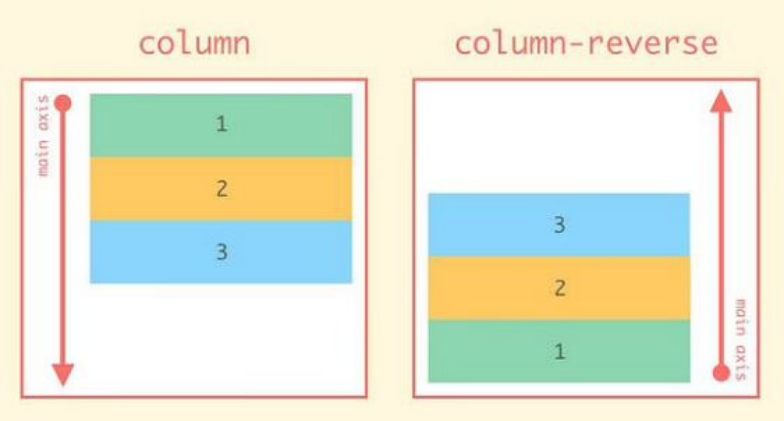
El **eje principal** o también llamado **main axis** es el más importante y es donde vamos a ver los cambios más grandes en cuanto a la ubicación de los elementos entre unos y otros.
El **eje secundario** o **cross axis**, es el que dependerá del principal, y nos permitirá alinear los elementos en conjunto.

Algo fundamental sobre este tipo de posicionamiento es que tanto el **main axis** como el **cross axis** no están definidos por el **plano horizontal** ni el **vertical**, sino que dependen de la configuración que nosotros les asignemos al momento de escribir el código.

flex-direction

Por defecto, el **main axis** es el eje horizontal y el cross axis el vertical, a menos que utilicemos, por ejemplo, la propiedad `flex-direction: column;` para invertir este posicionamiento (por defecto sería `flex-direction: row;`).
También podemos utilizar el valor `column-reverse` para crear una imagen en espejo de la columna (es decir que comience de abajo hacia arriba) o bien el valor `row-reverse` para una fila invertida (que comience desde la derecha hacia la izquierda).





justify-content

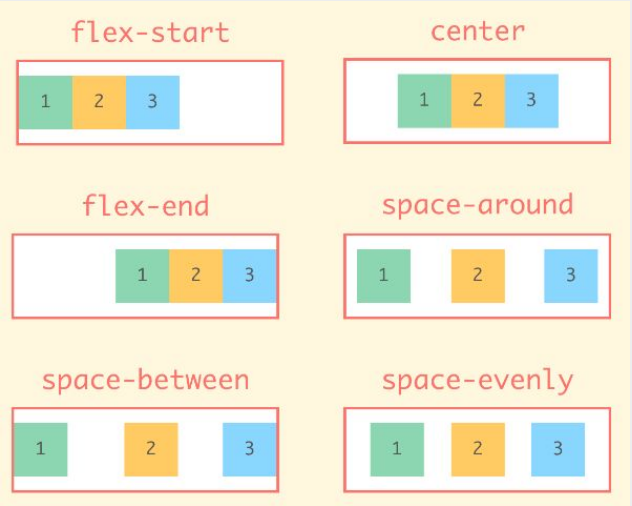
Habiendo ya definido que lo primordial es conocer cuál es el **main axis** y cuál el **cross axis**, ahora podemos comenzar a ver cómo podemos ordenar los elementos en este main axis utilizando **Flexbox**. En este eje utilizaremos la propiedad `justify-content` para indicarle a los elementos hijos cómo deben posicionarse entre sí.

Como valores de esta propiedad tenemos:

- `flex-start`: el valor por defecto, que alinea los elementos hacia el comienzo de la dirección (izquierda para `row`, derecha para `row-reverse`, arriba para `column` y abajo para `column-reverse`)
- `flex-end`: lo inverso a `flex-start`, por ejemplo, a la derecha para `row`, abajo para `column`, etc.
- `center`: alinea todos los elementos al centro

justify-content

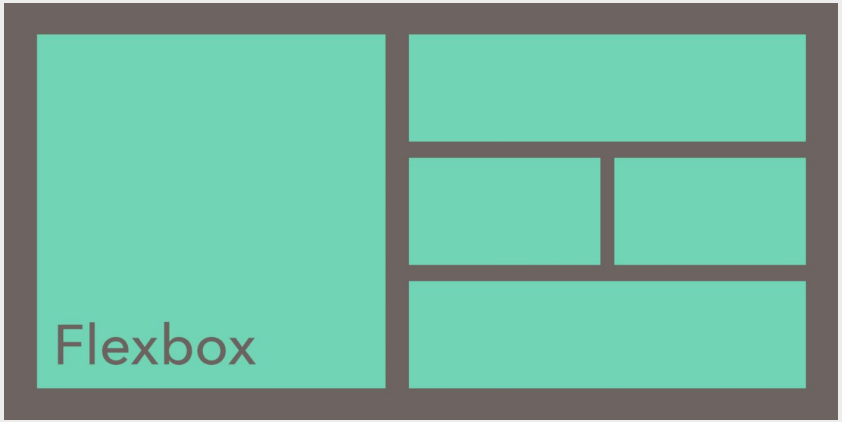
- `space-between`: separa los elementos lo más posible uno de otro, dejando el máximo espacio disponible entre ellos (el espacio entre todos los elementos es igual).
- `space-evenly`: muy similar a `space-between`, pero dejando también un espacio entre los elementos de los extremos y los bordes del contenedor (todos los espacios son iguales).
- `space-around`: parecido al anterior solamente que ahora cada elemento tiene los espacios hacia cada lado y éste mismo **NO** se solapa entre elementos, por ende el espacio que hay entre dos elementos es el doble que el que hay entre un elemento del extremo y el borde del contenedor.



justify-content

Por supuesto que estas propiedades, si tenemos un **elemento flexible** direccionado de manera vertical, tendrán **los mismos valores**, sólo que se aplicarán en la **dirección establecida**.

Esto nos permite comenzar a **ordenar** y **separar** nuestros elementos con mucha más precisión y empezar a crear interfaces **más complejas** y **cohesivas**.



¡Vamos al código!

align-items

Ahora que ya conocemos cómo se alinean los elementos en el **main axis**, podemos intentar ordenarlos en el **cross axis**.
Debemos tener en cuenta que en el **main axis** los elementos se separaban o se ordenaban respecto unos de otros; en el **cross axis** los elementos van a alinearse, en conjunto, en relación a este eje.

La propiedad a utilizar es `align-items` y sus valores son:

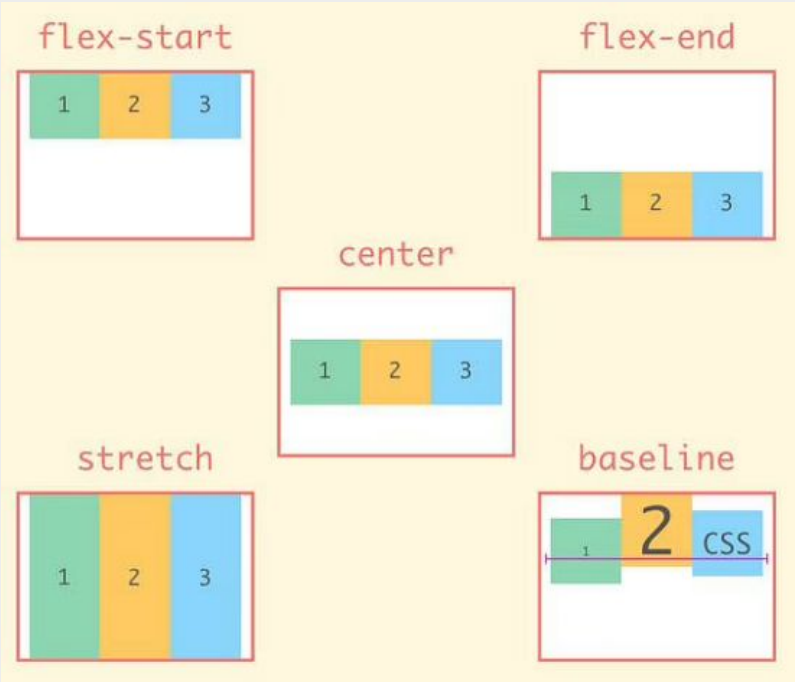
- `stretch`: el valor por defecto, que hace que los elementos ocupen todo el espacio disponible en este eje, ignorando su contenido y "expandiéndose" todo lo que puedan en el sentido que les indica el **cross axis** (de arriba a abajo para `column` por ejemplo).

- `flex-start`: al igual que en el `justify-content`, ordena a los elementos desde el comienzo del **cross axis** hacia el otro lado.

- `flex-end`: lo inverso a `flex-start`, desde el final del **cross axis** hacia el comienzo del mismo.

align-items

- `center`: alinea todos los elementos al centro.
- `baseline`: alinea todos los elementos en base a su contenido (como para que estén todos en el mismo renglón, por así decirlo) y no al tamaño de cada elemento (resulta muy útil cuando tenemos múltiples elementos con texto de tipo título en su interior, ya que todos van a comenzar a la misma altura o posición aunque tengan tamaños distintos en cuanto a su `height` o su `width`, según corresponda).



¡Vamos al código!

flex-wrap

Otra propiedad importante a la hora de crear interfaces dinámicas es la propiedad `flex-wrap`.
Si vemos un contenedor flexible con muchos elemento por dentro veremos que cuando ya no hay espacio disponible entre ellos, estos comienzan a achicarse en el sentido de su **main axis** haciendo que se vaya comprimiendo su contenido.

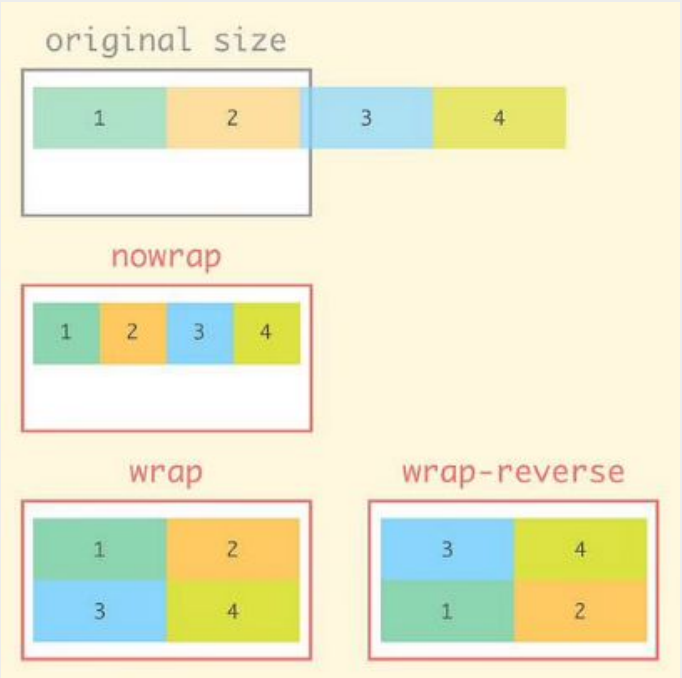
Dicho comportamiento es el que viene por defecto y es algo muy similar a lo que pasa con el `align-items: stretch;` pero para el **main axis**.

Este comportamiento es lo que nos permite modificar el `flex-wrap`, ya que esta propiedad hace que cuando un elemento ya no pueda ocupar su espacio mínimo, es decir, ya no entre en el flujo normal sin achicarse, crea una nueva línea y lo coloca allí.
Por esto mismo, nos puede resultar de gran utilidad a la hora de crear **grillas flexibles** o **contenedores fluidos**.

flex-wrap

Los valores más comunes son:

- `no-wrap`: es el comportamiento por defecto.
- `wrap`: es el que permite que se creen nuevas líneas y se vayan rellenando con el contenido.
- `wrap-reverse`: es igual al anterior pero de manera invertida, solo que aquí se invierte no como un espejo en el **main axis**, sino en el **cross axis**.



gap

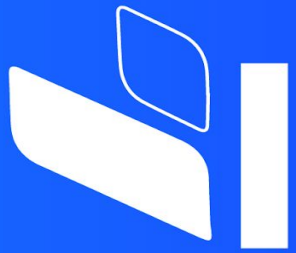
Esta propiedad nos permite definir qué espacio habrá entre los elementos dentro de un contenedor flexible.
De contar con una única línea o columna de contenido, la misma definirá simplemente el espacio que exista **entre elementos**.

En caso de utilizar líneas múltiples con propiedades como `flex-wrap`, la propiedad `gap` modificará tanto el espacio entre elementos como el espacio entre líneas. Es aquí donde podemos diferir esta propiedad en dos separadas: `column-gap` y `row-gap`.

Estas propiedades funcionan tal y como su nombre nos sugiere, el `row-gap` separaría las filas entre sí, mientras que el `column-gap` haría lo propio con las columnas.

Importante: Debemos tener en cuenta que utilizar gap al mismo tiempo que valores como `space-between` puede dar resultados inesperados si no tenemos en cuenta los tamaños que estamos manejando en el elemento contenedor.

Muchas gracias!



ICARO Asociación Civil
CUIT 30716564815
info@icaro.org.ar
www.icaro.org.ar