

# Flexbox

Módulo 03

# Flexbox

Una opción muy interesante al momento de encolumnar elementos de nuestra estructura es hacerlo a **través de flexbox**.

Es interesante para comprender esto mejor separar en dos partes las propiedades que veremos en la imagen de la derecha,

Con esto le damos al  
contenedor que será flex.

```
<style>
#contenedor {
  display: flex;
  width: 900px;
  padding: 2%;
  margin: auto;
  background-color: grey;
}

.items {
  background-color: pink;
  width: 50%;
  height: 100px;
  margin: 2%;
  padding: 1%;
  box-sizing: border-box;
}
</style>
```

# Flexbox

A su vez el contenedor puede tener una dirección esta tiene los siguientes valores posibles y se llama a saber,

**#ROW**, valor predeterminado, los ítems van de izquierda a derecha →

**#ROW-REVERSE**, de derecha a izquierda ←

**#COLUMN**, desde arriba hacia abajo ↓

**#COLUMN-REVERSE**, de abajo hacia arriba ↑

*por defecto es column y row.*

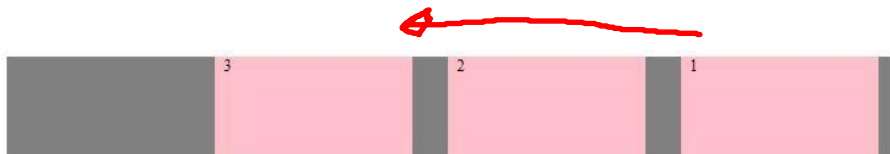
```
<style>
#contenedor {
  display: flex;
  width: 900px;
  padding: 2%;
  margin: auto;
  background-color: grey;
  flex-direction: row-reverse
}

.items {
  background-color: pink;
  width: 50%;
  height: 100px;
  margin: 2%;
  padding: 1%;
  box-sizing: border-box;
}
</style>
```

# Flexbox

*flexbox - direction : row-reverse.*

El ejemplo anterior, donde utilizamos la dirección de derecha a izquierda se verá de la siguiente manera,



Si hubiésemos utilizado column, se vería de la siguiente forma,



# Flexbox

Lo interesante es que los elementos contenidos se van a adaptar al contenedor, es decir que por más que el contenedor tuviese una medida tal que los elementos planteados así se desbordan estos acomodan su alto cambiándolo automáticamente para acomodarse al contenedor.

Al elemento contenedor **a su vez del contenedor que tiene display: flex** a su vez se le puede agregar la propiedad direction.

A menos que centramos el contenedor que lo haremos como en el ejemplo anterior con margin: auto , podemos indicar que el contenedor que tiene a su vez dentro los elementos encolumnados **estará hacia la derecha o hacia la izquierda**.

# Flexbox

Por otro lado nosotros hemos predispuesto **nuestro diseño de tal forma que los elementos** están distribuidos de manera proporcional.

En realidad, podría no pasar esto, por lo tanto también hay una propiedad para determinar cómo estarán los elementos dentro de nuestro contenedor.

*justify-content :*  
*da la posición en la que van a aparecer los elementos horizontalmente,*

```
<style>

#contenedor {
  display: flex;
  width: 900px;
  padding: 2%;
  margin: auto;
  background-color: grey;
  flex-direction: row-reverse
}

.items {
  background-color: pink;
  width: 50%;
  height: 100px;
  margin: 2%;
  padding: 1%;
  box-sizing: border-box;
}

</style>
```

# Flexbox

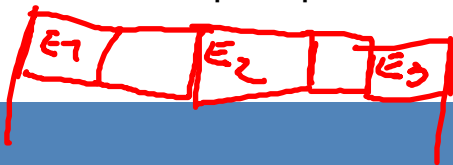
justify-content, es otra propiedad complementaria de Flexbox para determinar cómo estarán los elementos dentro de nuestro contenedor con los siguientes valores,

#FLEX-START, valor predeterminado, están al principio del contenedor

#FLEX-END, al contrario del valor anterior, están al final del contenedor

#CENTER, los elementos están centrados en el contenedor

#SPACE-BETWEEN, los elementos desde el principio del contenedor tendrán espacio



```
<style>
#contenedor {
  display: flex;
  width: 900px;
  height: 200px;
  margin: auto;
  background-color: grey;
  flex-direction: row-reverse;
  justify-content: space-around;
}

.items {
  background-color: pink;
  width: 200px;
  height: 100px;
  padding: 1%;
  box-sizing: border-box;
}
</style>
```

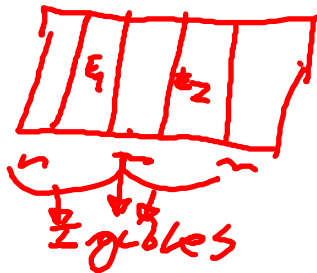
# justify-content



#SPACE-AROUND, parecido al anterior pero tendrán espacio no sólo entre los elementos sino también antes y después de cada uno

#SPACE-EVENLY, en el caso de space-around el espacio del segundo elemento estaba precedido por su propio espacio y su vez el espacio posterior del elemento anterior, pero en este caso, los espacios entre todos los elementos son iguales

En el caso de space-around el espacio del segundo elemento estaba precedido por su propio espacio y su vez el espacio posterior del elemento anterior, pero en este caso, los espacios entre todos los elementos son iguales





# justify-content

Es importante aclarar que todo el código anterior se basará en la siguiente estructura de **HTML**,

```
<div id="contenedor">  
  <div class="items">1</div>  
  <div class="items">2</div>  
  <div class="items">3</div>  
  </div>
```

# align-items

*justify-content: → Horizontal.*

*align-items: ↑ Vertical.*

Para poder acomodar los elementos verticalmente,

## #STRETCH

Valor predeterminado, los ítems están pegados al contenedor (en tanto y en cuanto estos elementos contenidos no tengan height sino el valor predeterminado es flex-start).

*↳ o sea height: auto.*

## #FLEX-START

Están en la parte superior del contenedor

## #FLEX-END

Al contrario del valor anterior, están al final del contenedor

## #CENTER

Los elementos están centrados en el contenedor

# align-items

Un ejemplo en nuestro código de la propiedad anterior será como el que se puede ver en la imagen derecha, otro ejemplo interesante lo podremos ver en la imagen de abajo,

```
<style>

nav { display: flex; }

a { color: red; width: 100px; height: 30px; background: black;}

</style>

</head>

<body>
<nav><a href="#"> inicio </a> <a href="#"> contacto </a> </nav>
<body>
```

```
<style>
#contenedor {
    display: flex;
    width: 900px;
    height: 200px;
    margin: auto;
    background-color: grey;
    align-items:center;
}

.items {
    background-color: pink;
    width: 200px;
    height: 100px;
    padding: 1%;
    box-sizing: border-box;
}

</style>
```

# Revisión

- Repase la propiedad **float**
- Trabaje con todas las opciones de la propiedad **float**
- Aplique todas las propiedades en el **proyecto integrador**
- Combine las propiedades con **margin y padding**
- Realice las preguntas necesarias al/la docente antes de continuar



# ¡Muchas gracias!

¡Sigamos trabajando!