# Flexbox
# Flexible box layout CSS3

**Elioth Monroy Martos** **Miguel Ángel Chaves**

Tecnologías para la Web

# Antecedentes

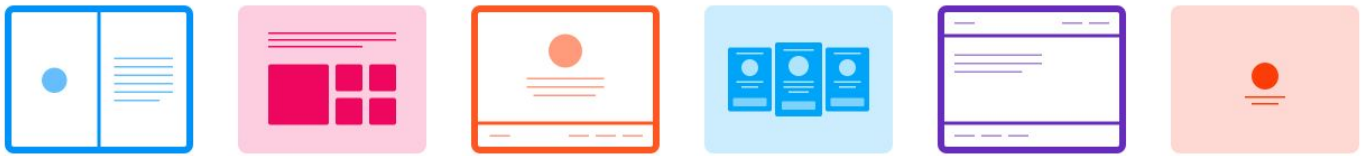
Tradicionalmente, en CSS se ha utilizado el posicionamiento (static, relative, absolute...), los elementos en línea o en bloque (y derivados) o los float.

Lo que a grandes rasgos no dejaba de ser un sistema de creación de diseños bastante tradicional que no encaja con los retos que se tienen hoy en día (sistemas de escritorio, dispositivos móviles, múltiples resoluciones, etc...).

# ¿Qué es Flexbox?

Es un modo de diseño de CSS3 que permite colocar los elementos de una página para que se acomoden de forma predecible cuando el diseño de la misma debe acomodarse a diferentes tamaños de pantalla y diferentes dispositivos.

Flexbox es sencillo de utilizar, ya que los elementos hijos de una “caja flexible” pueden colocarse en cualquier dirección y pueden tener dimensiones flexibles para adaptarse al espacio visible.



# Compatibilidad

Actualmente todos los navegadores modernos soportan esta funcionalidad como parte del estándar CSS3.

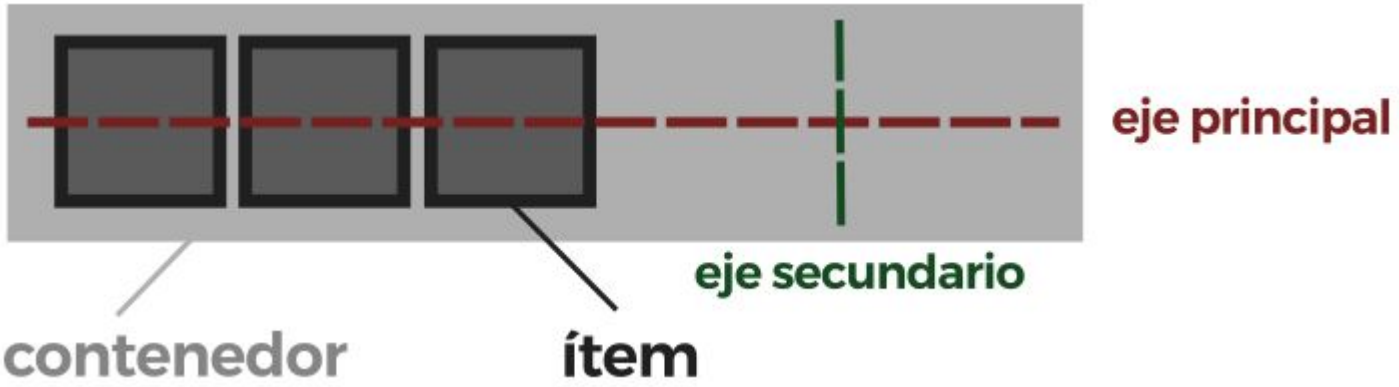
# Concepto de Flexbox

Lo que caracteriza un diseño flexible es su habilidad para alterar el ancho y el alto de sus elementos, para ajustarse lo mejor posible al espacio disponible en cualquier dispositivo.

Un contenedor flexible expande sus elementos para rellenar el espacio libre, o los comprime para evitar que rebasen el área prevista.

Flexbox *no parte de ninguna dirección predeterminada*, a diferencia de como ocurre con el modelo de “bloque”, que asume una disposición vertical de los elementos, o lo que pasa con el modelo “en línea”, que asume una disposición horizontal.

# Estructura de un flexbox



# Contenedor flexible

Es el elemento padre que contiene elementos flexibles, se define usando los valores *flex (bloque)* o *inline-flex (en línea)* en la propiedad **display**.

**HTML**

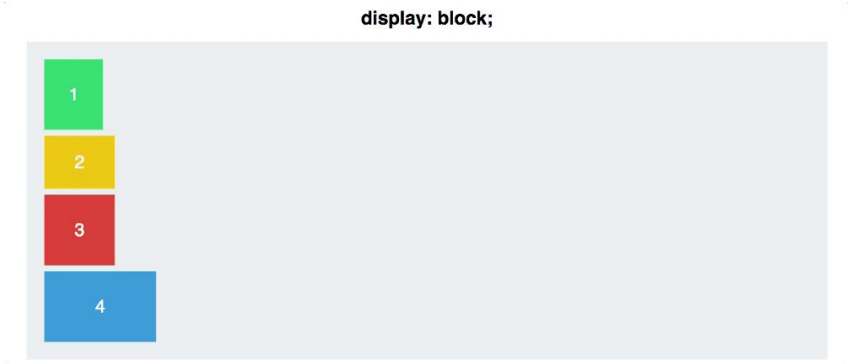
```
<div class="flex-container"></div>
```

**CSS**

```
.flex-container {
    display: flex;
}
```

# Elemento flexible

Cada hijo de un contenedor flexible se convierte en un elemento flexible.

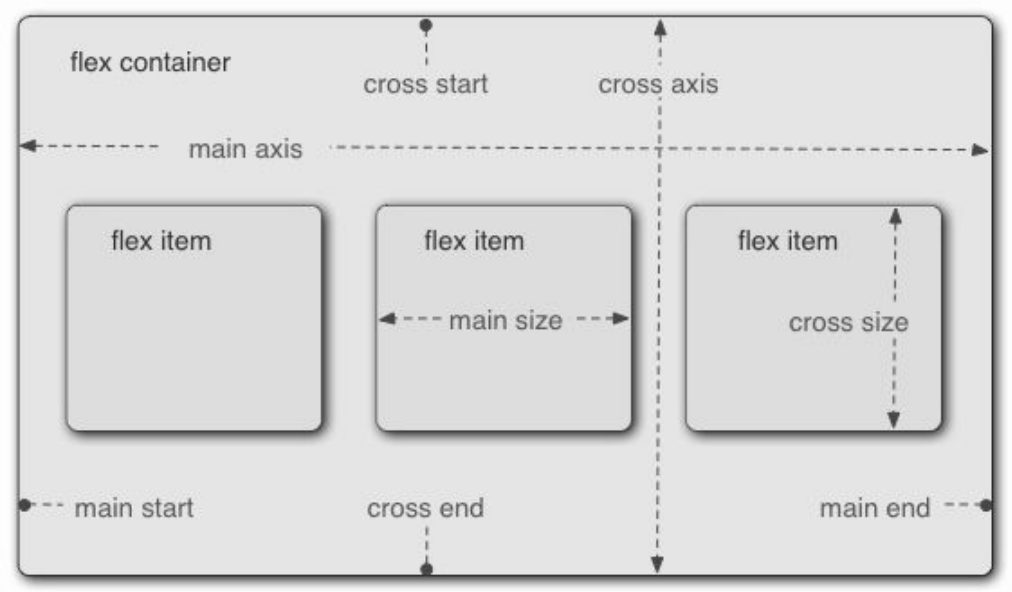



**HTML**

```
<div class="flex-container">
    <div>1</div>
    <div>2</div>
    <div>3</div>
    <div>4</div>
    <div>5</div>
    <div>6</div>
    .
    .
    .
    <div>n</div>
</div>
```

# Ejes

En Flexbox, se siguen dos ejes, el principal (main axis) es el eje a lo largo del cual los elementos flexibles se suceden unos a otros. El eje secundario (cross axis) es el eje perpendicular al eje principal.



# Consideraciones

- El texto que se encuentre dentro de un contenedor flexible, será automáticamente envuelto en un elemento flexible anónimo. Sin embargo, si un elemento flexible contiene solamente espacios en blanco no será mostrado, como si este tuviera la propiedada *display:none*.
- Los hijos de un contenedor flexible que tengan un posicionamiento absoluto, se situarán de manera que su posición estática se determine en referencia a la esquina del inicio principal de su contenedor flexible.

# Propiedades

1. Dirección de los ejes.
2. Alineación de ítems.
3. Propiedades de los elementos flexibles

# Dirección de los ejes

Existen dos propiedades principales para manipular la dirección y comportamiento de los ítems a lo largo del eje principal del contenedor.

**flex-direction.** Cambia la orientación del eje principal.

**flex-wrap.** Evita o permite el desbordamiento (multilínea).

**CSS**

```
.flex-container {
    display: flex;
    flex-direction:
        row - horizontal.
        row-reverse - horizontal r.
        column - vertical.
        column-reverse - vertical r.
    ;
    flex-wrap:
        nowrap - ítems en una sola
línea NO DESBORDAMIENTO.
        wrap - ítems en modo
multilínea DESBORDAMIENTO.
        wrap-reverse - ítems en modo
multilínea invertido DESBORDAMIENTO.
    ;
}
```

# flex-flow

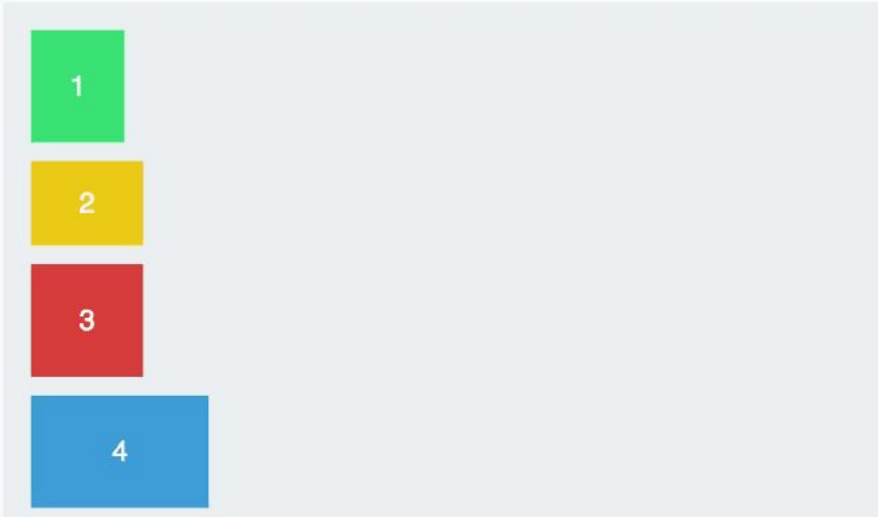
Permite especificar las dos propiedades anteriores.

- flex-direction
- flex-wrap

CSS

```
.flex-container {
    display: flex;
    flex-direction: row wrap;
}
```

flex-direction: column;



# Alineación de ítems

Ahora que tenemos un control básico del contenedor de estos ítems flexibles, necesitamos conocer las propiedades existentes dentro de flexbox para disponer los ítems dependiendo de nuestro objetivo.

**justify-content.** Se utiliza para alinear los ítems del eje principal.

**align-items.** Sirve para alinear los elementos flexibles verticalmente.

**align-content.** Alinea cada una de las líneas del contenedor multilínea.

**align-self.** Alinea cada una de las líneas del contenedor multilínea particular.

# justify-content

Sirve para colocar los ítems de un contenedor mediante una disposición concreta a lo largo del eje principal

**CSS**

```
.flex-container {
    display: flex;
    justify-content:
        center - centrado.
        flex-start - posc. inicio.
        flex-end - posc. final.
        space-between - espaciado.
        space-around - distribución.
    ;
}
```

**justify-content: flex-start;**

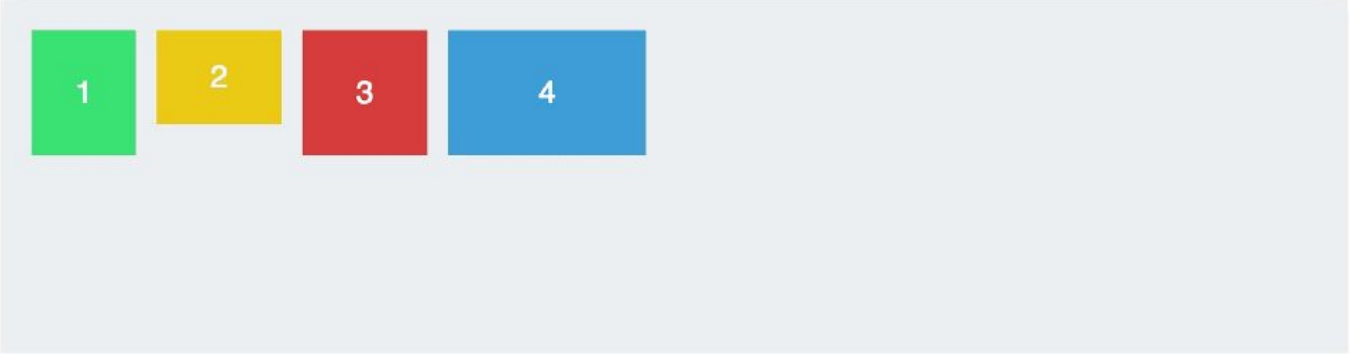
# align-items

Se encarga de alinear los ítems en el eje secundario del contenedor

**CSS**

```
.flex-container {
    display: flex;
    align-items:
        center - centrado.
        flex-start - posc. inicio.
        flex-end - posc. final.
        baseline - alineado debajo.
        stretch - todo el espacio.
    ;
}
```

**align-items: flex-start;**



# align-content

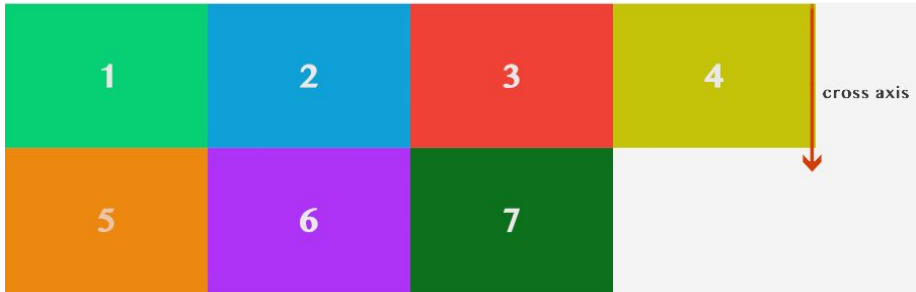
La propiedad align-content es un caso particular del anterior. Nos sirve cuando estemos tratando con un contenedor flex multilínea donde el eje principal se divide en múltiples líneas.

**CSS**

```
.flex-container {
    display: flex;
    align-content:
        center - centrado.
        flex-start - posc. inicio.
        flex-end - posc. final.
        stretch - todo el espacio.
        space-between - espaciado.
        space-around - distribución.
    ;
}
```

**ALIGN-CONTENT:STRECH;**

```
.container{ display:flex; min-height:100vh;border:5px solid #c2480c;flex-wrap:wrap; align-content:strech;}
.flex-item{  width:27.999%;}
```



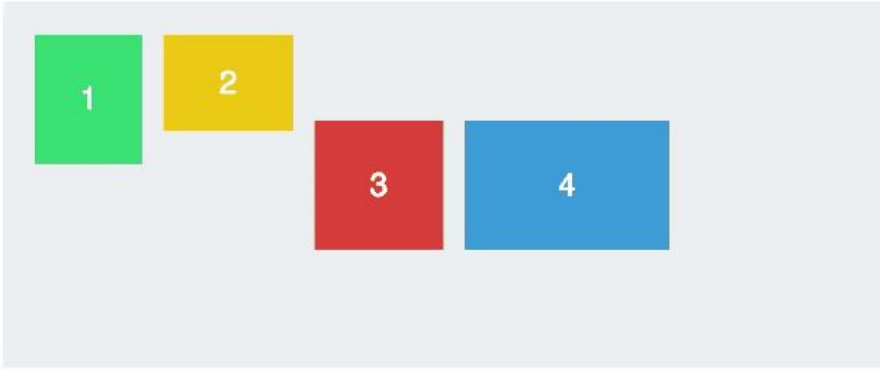
# align-self

La propiedad align-self actúa exactamente igual que align-items, sin embargo es la primera propiedad de flexbox que vemos que se utiliza sobre un ítem hijo específico y no sobre el elemento contenedor. Salvo por este detalle, funciona exactamente igual que align-items.

**CSS**

```
#container {
  align-items: center;
}
.square#one {
  align-self: baseline;
}
.square#two {
  align-self: baseline;
}
```

**align-self: flex-start;**



# **Demo #1**
## Centrar un elemento perfectamente

# Propiedades de los elementos flexibles

A excepción de la propiedad align-self, todas las propiedades que hemos visto hasta ahora se aplican sobre el elemento contenedor. Las siguientes propiedades, sin embargo, se aplican sobre los ítems hijos.

- ***order***
- ***flex-grow***
- ***flex-shrink***
- ***flex-basis***
- ***flex***

# flex-basis

Define el tamaño por defecto que tendrán los ítems antes de aplicarle una etiqueta flex. Se aplica por unidades de medida o por **content** que ajusta automáticamente el tamaño.

**CSS**

```
.square#one {
  flex-basis: 150px;
}
.square#two {
  flex-basis: 150px;
}
```

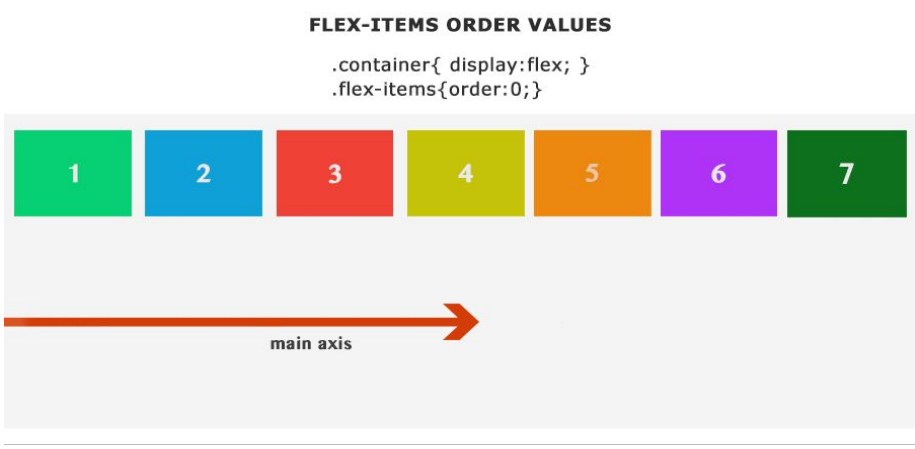
# order

La propiedad *order* modifica y establece el orden de los ítems según una secuencia numérica.

Por defecto, todos los ítems flex tienen un order: 0 implícito, si indicamos un order con un valor numérico, este irá colocando los ítems según su número.

**CSS**

```
#container {
  align-items: center;
}
.square#one {
  order: 4;
}
.square#two {
  order: -2;
}
```



# flex-grow

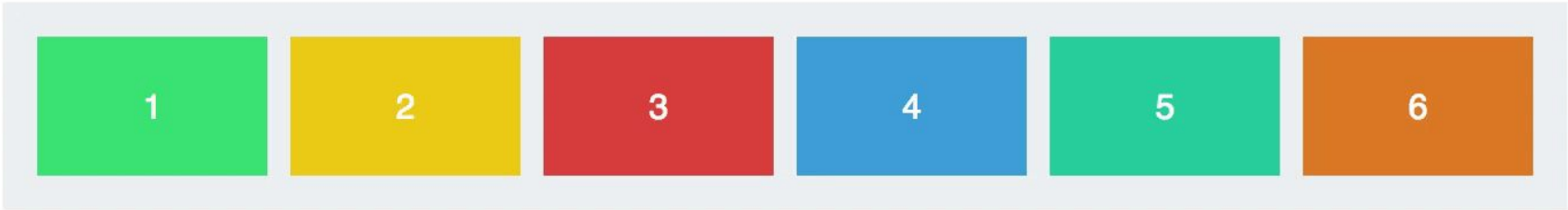
Permite especificar que tanto crecerá un elemento flexible en comparación con el resto.

CSS

```
.square#one {
  flex-grow: 1;
}
.square#two {
  flex-grow: 2;
}
```

.square { flex-grow: 1; }

.square#three { flex-grow: 1; }



# flex-shrink

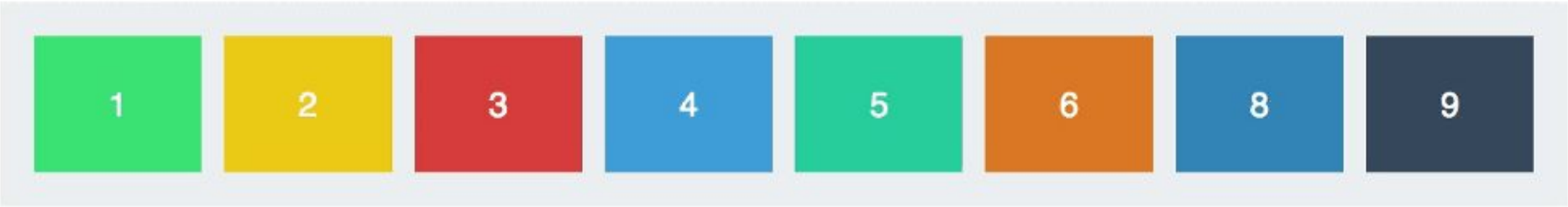
Permite especificar qué tan pequeño se volverá un elemento en comparación al resto

CSS

```
.square#one {
  flex-shrink: 1;
}
.square#two {
  flex-shrink: 5;
}
```

.square { flex-shrink: 1; }

.square#three { flex-shrink: 0; }



# flex

Nos permite definir las 3 propiedades anteriores. (flex-grow, flex-shrink, flex-basis)

CSS

```
.square#one {
  flex: 2 1 300px;
}
.square#two {
  flex: 1 2 300px;
}
```

.square#one { flex: 2 1 300px; }

.square#two { flex: 1 2 300px; }

# Demo #2
## Galería de imágenes responsivas

# Referencias


- https://developer.mozilla.org/es/docs/Web/CSS/CSS_Flexible_Box_Layout/Usando_las_cajas_flexibles_CSS
- https://www.w3schools.com/css/css3_flexbox.asp
- https://medium.freecodecamp.org/even-more-about-how-flexbox-works-explained-in-big-colorful-animated-gifs-a5a74812b053


Código fuente:

- https://github.com/EliothMonroy/Tecnologias-Web/tree/master/expo

# Gracias ;)

Link de la presentación:

goo.gl/9n3xTM