FINEST en Desarrollo Web

Tema 5 - Page Layout: Flexbox

Agenda de la clase

Agenda

- Repaso
- Flexbox
- Ejercicio

Repaso

REPASO

Elementos de tipo Block vs. Inline

Por defecto, todo elemento en HTML tiene un tipo de visualización (*display*) predeterminado. Los dos más comunes son:

Block: es un elemento que ocupa todo el ancho de su elemento padre (contenedor), generando un salto de línea antes y después de sí mismo.

Elementos *block* más usados: `<h1>`, `<h2>`, `<div>`, `<p>`, `<ul>`, `<ol>`, `<form>`.

```
Este es un elemento <div>.
```

Inline: es un elemento que ocupa sólo el ancho que precisa y no genera saltos de línea.

Elementos *inline* más usados: `<a>`, `<em>`, `<strong>`, `<button>`, `<span>`.

```
Este es un elemento <span> dentro de un párrafo.
```

CSS Box Model

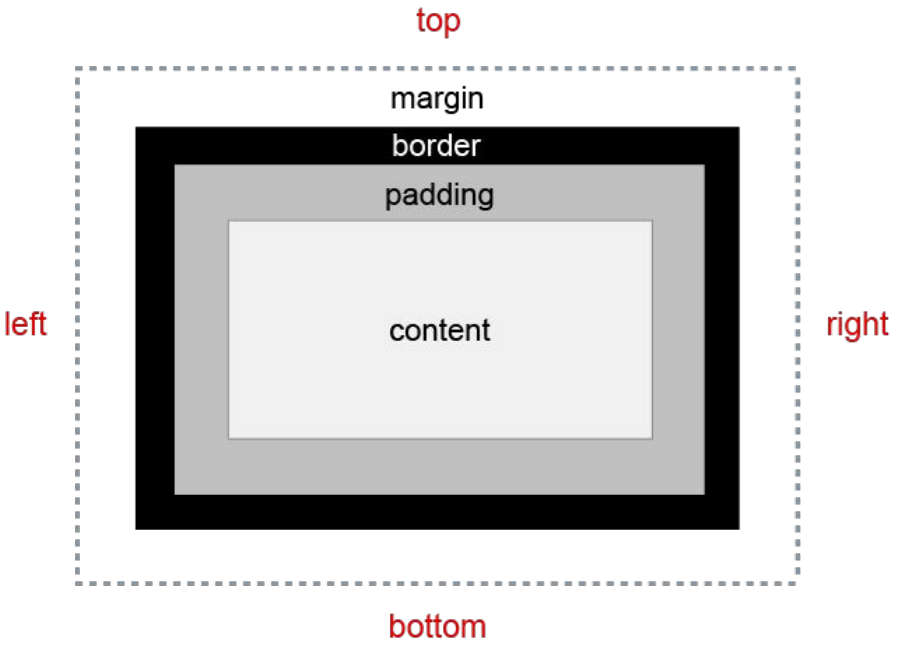
REPASO

Un `h1`, un `p`, un `div`, un `span`, etc., todos son tratados como boxes.

Todos los elementos HTML son tratados como *boxes* (cajas) por el navegador. Pueden ser `inline` o `block` boxes.

Un box consiste en:

- Content.
- Padding.
- Border.
- Margin.



Usar Chrome *Developer Tools* para inspeccionar *boxes*.

👉 Ver video explicativo.

REPASO

Page layout – Estructura de una página

Propiedad	¿Qué hace?	Posibles valores

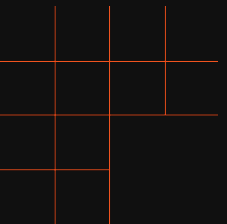
`position`	Establece el tipo de posicionamiento de un elemento.
	Más información: https://css-tricks.com/almanac/properties/p/position/. Se suele usar junto con las propiedades `top`, `bottom`, `left` y `right`.
		`static`
		`absolute`
		`relative`
		`fixed`

`display`	Establece si un elemento debe ser mostrado o no, y en caso afirmativo, cómo se debe mostrar.	`none`
		`block`
		`inline`
		`inline-block`
		`flex`
		`grid`

`float`	Establece si un elemento debe "flotar", y en caso afirmativo, hacia dónde debe flotar.
	Más información: https://css-tricks.com/all-about-floats/.
		`left`
		`right`
		`none`

Flexbox

Otra forma de armar layouts



Flexbox (1/3)

Flexbox no es una propiedad CSS, sino un conjunto de propiedades CSS, para poder armar el *layout* de una página. Es decir, Flexbox permite posicionar elementos en la página, de forma horizontal y vertical.

En general, lo que nos propone Flexbox es controlar la ubicación de los elementos (*flex items*) a través de propiedades que le pasaremos a su contenedor padre (*flex container*).

Flexbox (2/3)

Flexbox es muy poderoso, pero también es bastante complejo.

Tiene tantas opciones de configuración, que se podría hacer un curso sólo de Flexbox. De hecho, hay uno muy bueno y gratuito hecho por Wes Bos: https://flexbox.io. Nosotros nos limitaremos a hacer una breve introducción al tema.

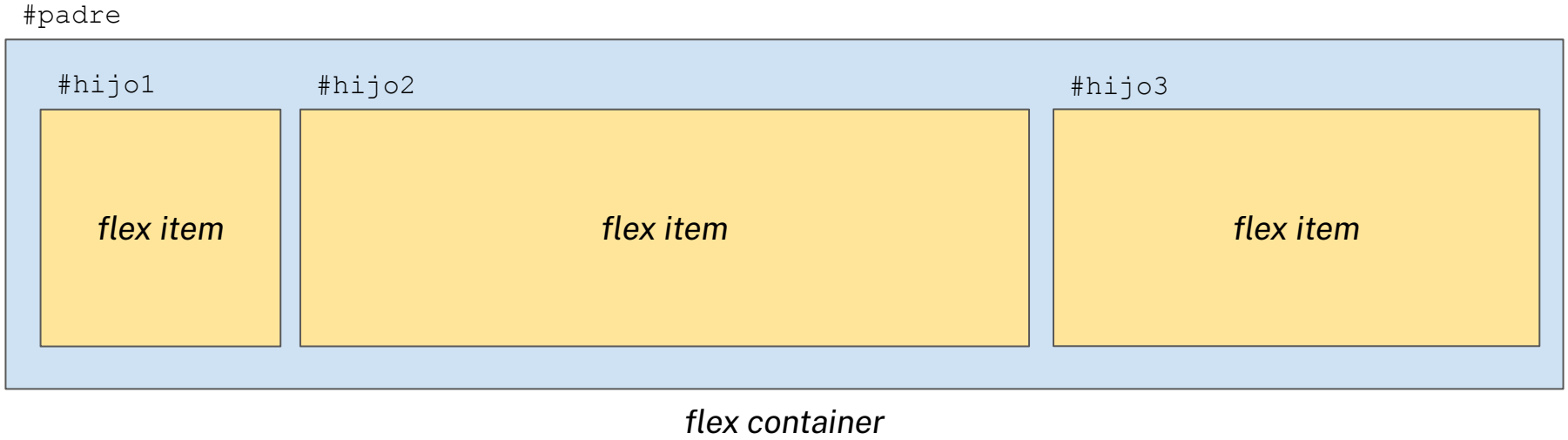
Documentación: MDN / CSS Tricks.

👉 Podemos hacer los primeros niveles entre todos, en clase.

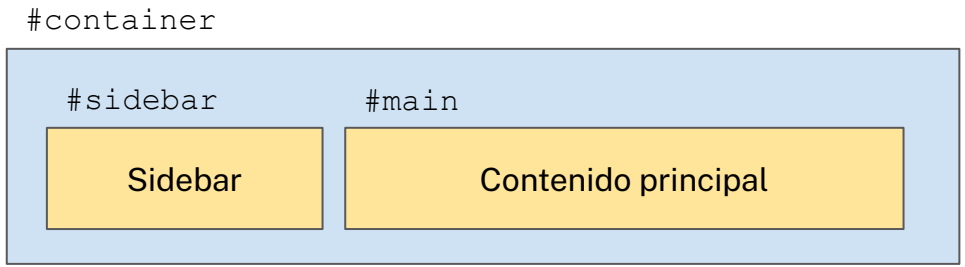
👉 Juego muy bueno para aprender sobre el tema: 🐸 FlexboxFroggy.

Flexbox (3/3)

Como señalamos, la idea principal de Flexbox es: dado un elemento *padre*, al cual se llamará *flex container*, ubicar sus elementos *hijos*, a los cuales se llamará *flex items*.



Flexbox – Ejemplo 1



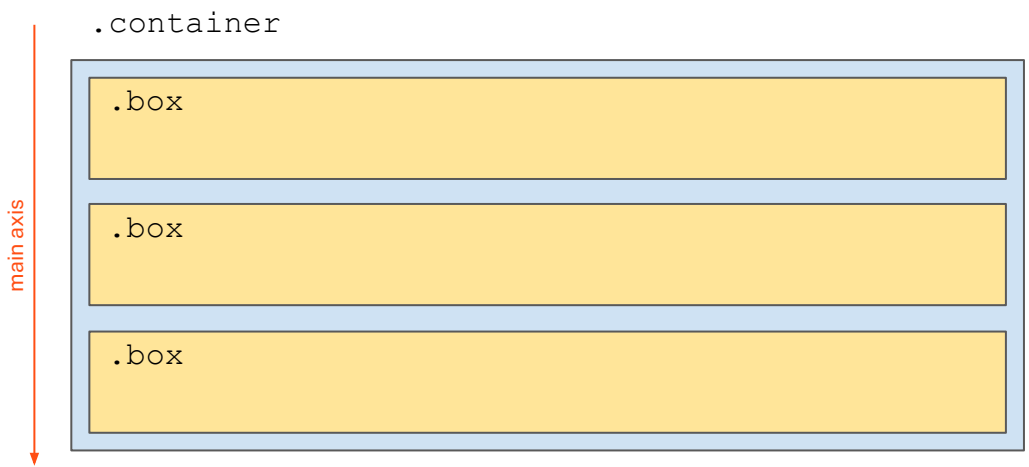
En este ejemplo, se establece que el *flex container* se comporte como una *row*, y por lo tanto, los *flex items* se comportan como columnas. La primera de ancho ⅓ y la segunda ⅔.

```css
#container {
  display: flex;
  flex-direction: row; /* Default */
}
#sidebar {
  flex: 1;
}
#main {
  flex: 2;
}
```

Flexbox – Flex Direction

Por defecto, los *flex items* se posicionan en fila. En este caso, se dice que el eje principal (*main axis*) es el horizontal. Ver el ejemplo de la diapositiva anterior.

Sin embargo, Flexbox permite cambiar el eje principal (hacerlo vertical) y posicionar los *flex items* en columna, usando la propiedad `flex-direction`.

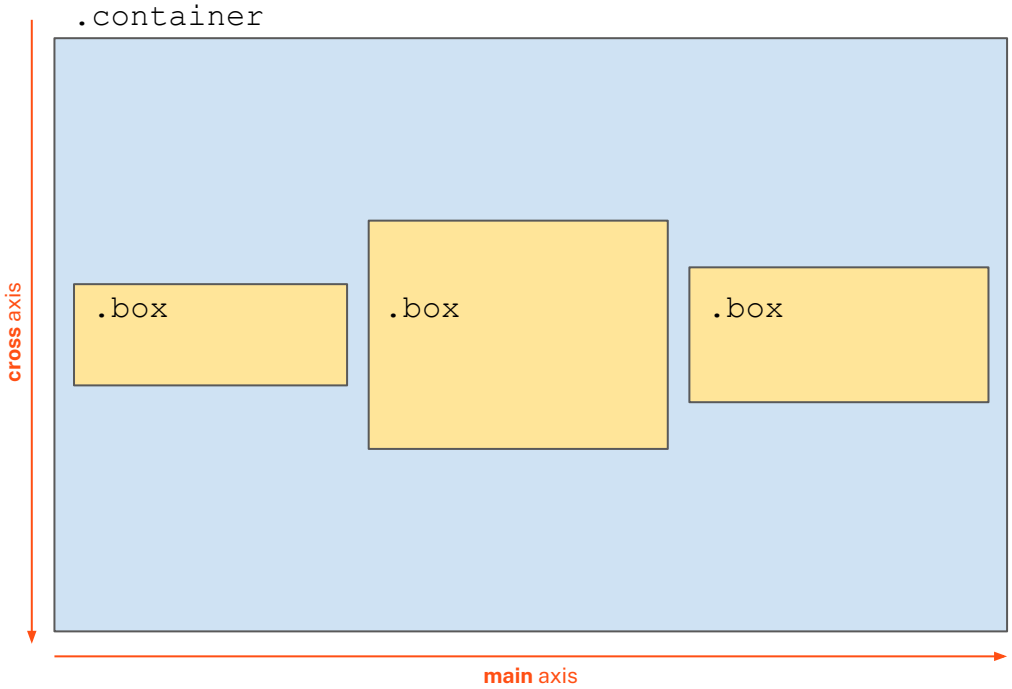



```css
.container {
  display: flex;
  flex-direction: column;
}
```

Flexbox – Alineación vertical (1/2)

👉 Ver Demo

Alinear elementos de forma vertical siempre fue algo complicado en CSS, sobre todo si las alturas varían. Gracias a Flexbox, este problema se resuelve muy fácilmente.



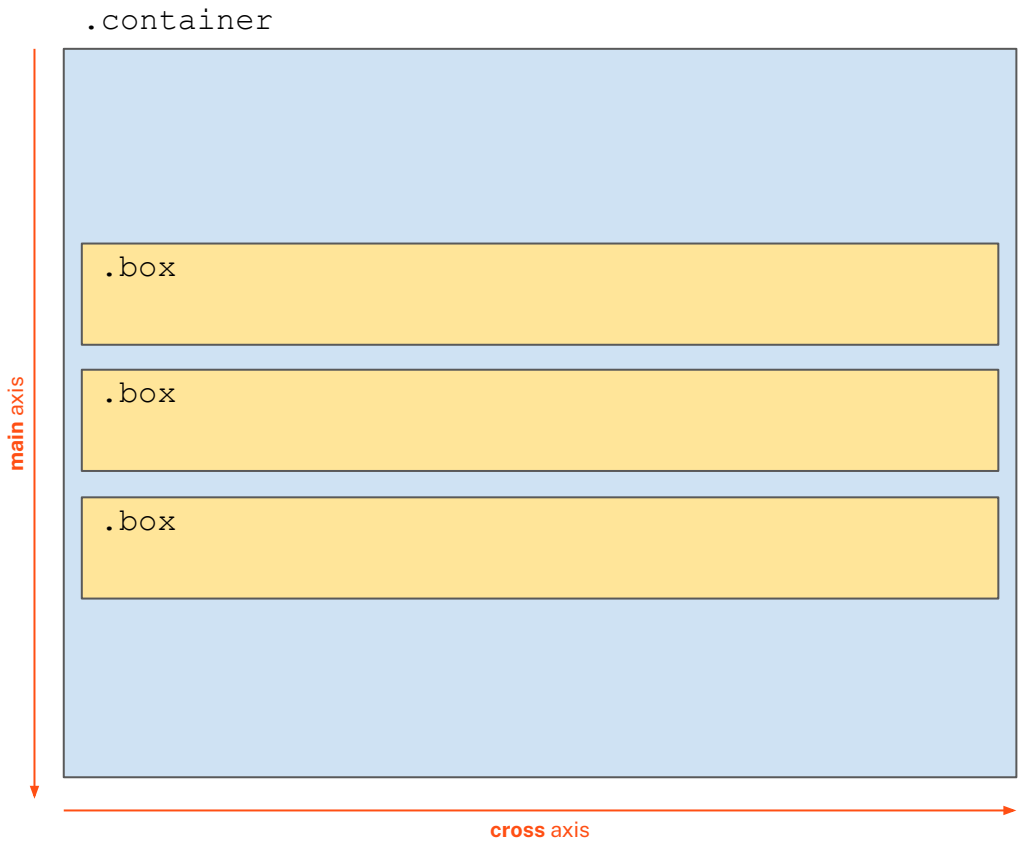

```css
.container {
  height: 800px;
  display: flex;
  flex-direction: row; /* Default */
  align-items: center;
}
```

Nota 1: La propiedad `flex-direction` en este caso no es necesaria, ya que `row` es el valor por defecto.

Nota 2: La propiedad `align-items` se usa para alinear los elementos en el *cross axis*, que podría ser el eje vertical u horizontal, dependiendo de cómo esté configurado el `flex-direction`

Flexbox – Alineación vertical (2/2)

👉 Ver Demo

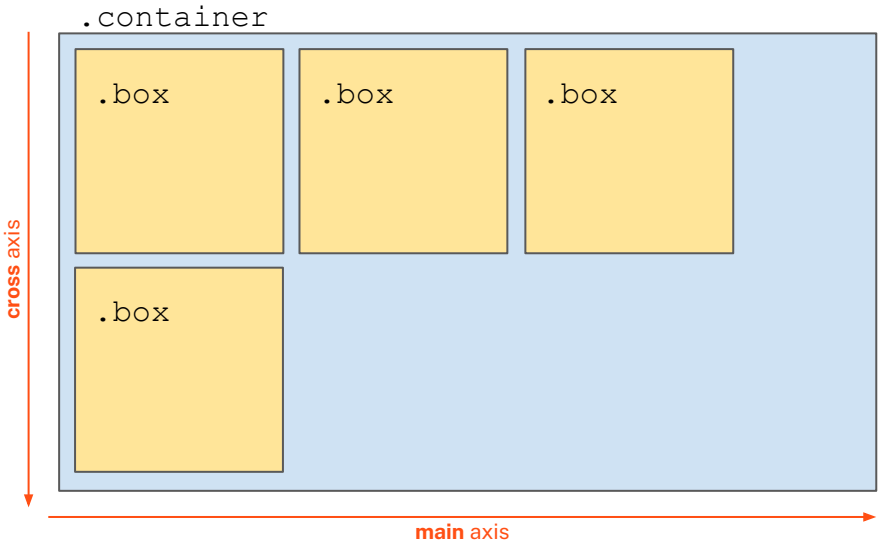



```css
.container {
  height: 100vh;
  display: flex;
  flex-direction: column;
  justify-content: center;
}
```

Flexbox – Flex Wrap

Por defecto, los *flex items* se colocan en una sola línea, es decir, no bajan de fila, y todos se alinean en la misma fila o columna dependiendo de la dirección del contenedor.

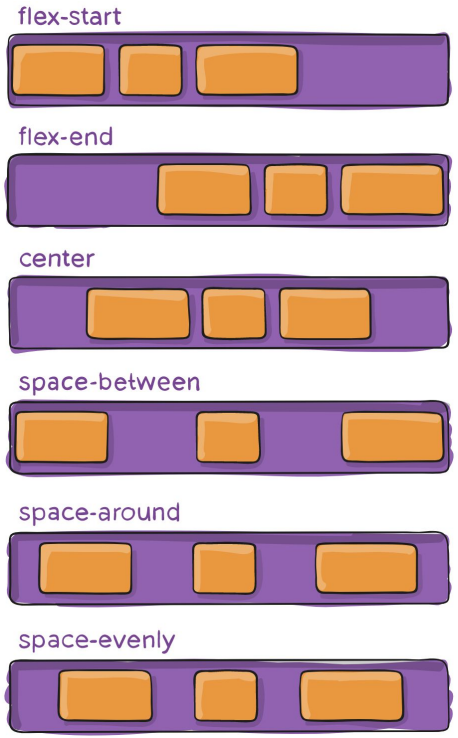
Flexbox permite cambiar este comportamiento utilizando la propiedad `flex-wrap`, que permite que los flex items se ubiquen en una nueva línea (o columna) cuando no caben en la principal.



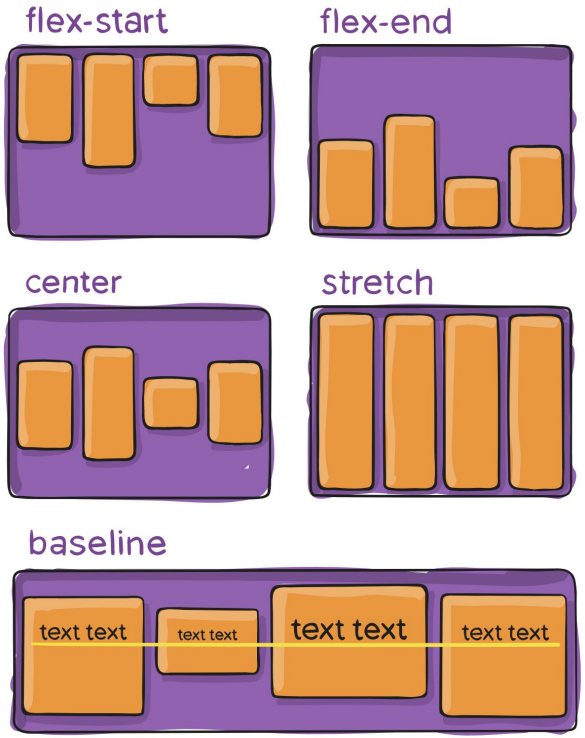

```css
.container {
  display: flex;
  flex-wrap: wrap;
  width: 500px;
}
.box {
  width: 140px;
}
```

Flexbox – Justify Content & Align Items

Valores para `justify-content`:



Valores para `align-items`:



Fuente

Ejercicio

Ejercicio

1. Crear una carpeta en el Escritorio (o donde prefieran) con el nombre `Clase05_Ejercicio`.
2. Abrir dicha carpeta en Visual Studio Code.
   Esto se puede hacer yendo al menú: `File > Open Folder` en Windows o `File > Open` en Mac.
3. Desde VSC, crear un archivo llamado `index.html` dentro de la carpeta.
4. Desde VSC, crear una carpeta `css` y dentro de la misma el archivo `styles.css`.
5. Escribir el HTML y CSS necesario para lograr un resultado similar al siguiente diagrama.

Ejercicio (cont)