



Facultad Regional Rosario

Universidad Tecnológica Nacional

Clase 06. DESARROLLO WEB

FLEXBOX



OBJETIVOS DE LA CLASE

- Conocer nuevos elementos de flexbox.
- Aplicar sus diferentes variantes.

GLOSARIO:

Clase 5

Box model: ese concepto de que “todo es una caja”, da lugar a algo llamado en web como box model. Sin importar si son de línea o de bloque (pero tienen su incidencia en lo que sean), todas las etiquetas tienen propiedades en común.

Width: es la propiedad CSS que controla la anchura de la caja de los elementos.

Height: es la propiedad CSS que controla la altura de la caja de los elementos.

Overflow: es una propiedad que tiene 4 valores posibles. Ellos son: *visible*, *hidden*, *scroll* y *auto*.

Espacio exterior

Margin (márgenes): las propiedades *margin-top*, *margin-right*, *margin-bottom* y *margin-left*, se utilizan para definir los márgenes de cada uno de los lados del elemento por separado.

Espacio exterior

Padding (relleno): las propiedades *padding-top*, *padding-right*, *padding-bottom* y *padding-left*, se utilizan para definir los espacios internos de cada uno de los lados del elemento, por separado.

Border: las propiedades *border-top*, *border-right*, *border-bottom*, y *border-left*, se utilizan para definir los bordes de cada lado del elemento por separado.

GLOSARIO:

Clase 5

Display: se encarga de definir cómo se ve un elemento HTML.

Inline-block: esta propiedad permite tomar lo mejor de ambos grupos. Brinda la posibilidad tener “padding” y “margin” hacia arriba y abajo.

Float: la flotación consiste en mover un elemento hacia la derecha o izquierda de su línea, y todo lo que viene después se acomodará en el “hueco” que queda vacío.

Clear: esta propiedad permite modificar el comportamiento por defecto del posicionamiento flotante, para forzar a un elemento a mostrarse debajo de cualquier caja flotante.

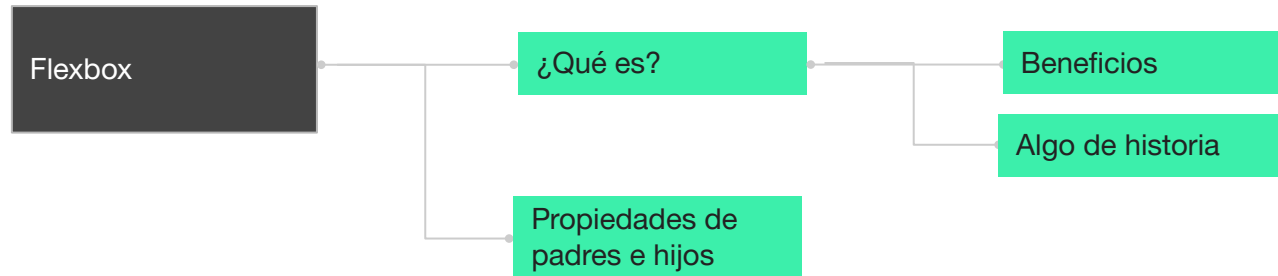
Position: una propiedad CSS pensada para ubicar un elemento, con una libertad muy flexible.

Propiedad z-index: entra en juego cuando dos elementos que tienen position se superponen. Acepta como valor un número (sin ninguna unidad, ni px, ni cm, ni nada); a valor más alto, se mostrará por encima de los demás elementos.

MAPA DE CONCEPTOS

MAPA DE CONCEPTOS CLASE 6

¡Para
recordar!



FLEXBOX



Facultad Regional Rosario
Universidad Tecnológica Nacional

¿QUÉ ES FLEXBOX?

Flexbox es un modo de diseño que nos permite crear estructuras para sitios web de una forma más fácil. Si ya sabes de HTML y CSS, probablemente alguna vez habrás visto que los sitios web se realizan utilizando la propiedad float, para desplazar contenedores.

Con Flexbox ya no necesitarás usar float para posicionar tus elementos. Al contrario, con Flexbox podrás posicionar y distribuir los elementos como tú quieras.

¿QUÉ ES FLEXBOX?

Entonces, no se trata de una propiedad de CSS, sino de un conjunto de ellas. Se basa sobre un contenedor (**padre**) para ordenar a sus ítems (**hijos**).

No sólo puedes posicionar elementos vertical y horizontalmente, sino que **puedes establecer cómo se distribuirán, el orden que tendrán e incluso el tamaño que tendrán en proporción a otros elementos**. Esto es perfecto para crear diseños adaptables a dispositivos móviles (Responsive Design).

BENEFICIOS

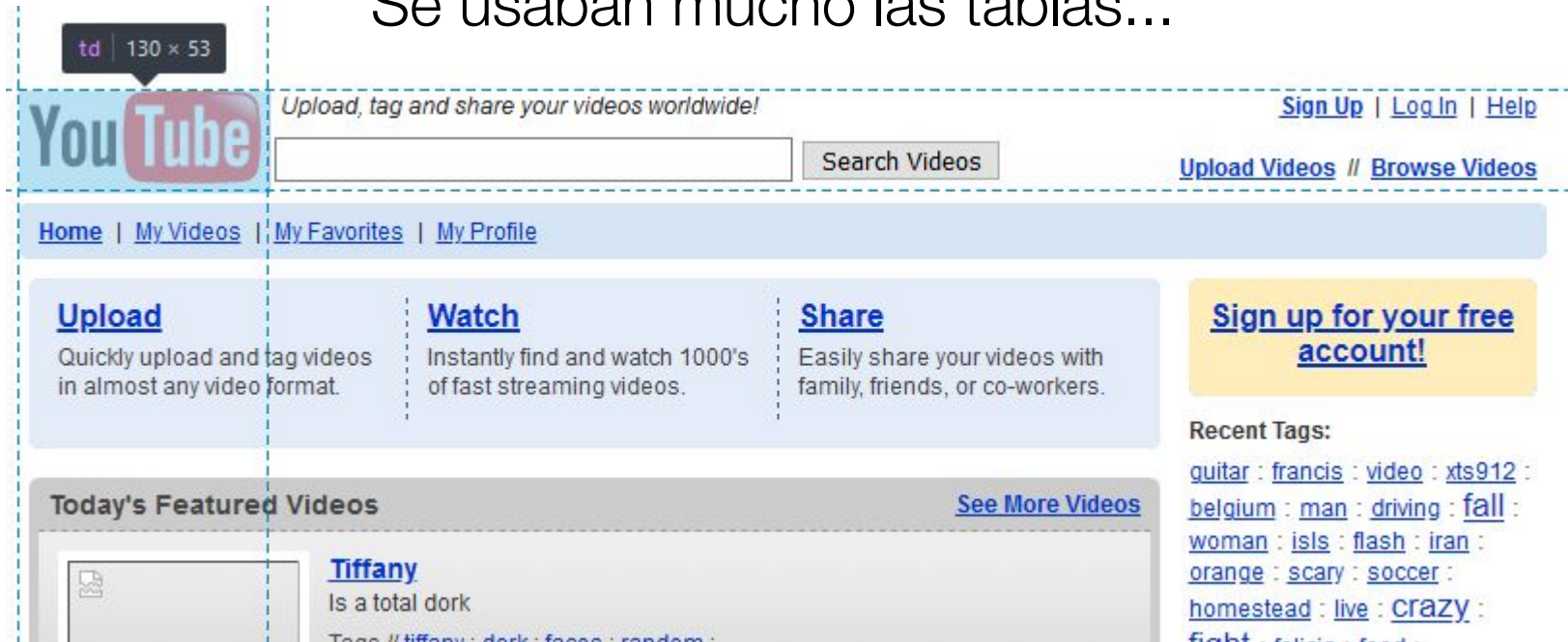
- Distribuir los elementos en sentido vertical y horizontal.
- Facilita la adaptación del contenido en distintos dispositivos
- Redefinir el sentido del flujo de los elementos (hacia arriba, hacia abajo, hacia la izquierda o hacia la derecha).
- Alinear los elementos respecto al padre o respecto a sus hermanos.

ALGO DE HISTORIA

Veamos, brevemente, un poco de historia acerca del modo en que ha evolucionado la forma de crear layouts...

ANTES

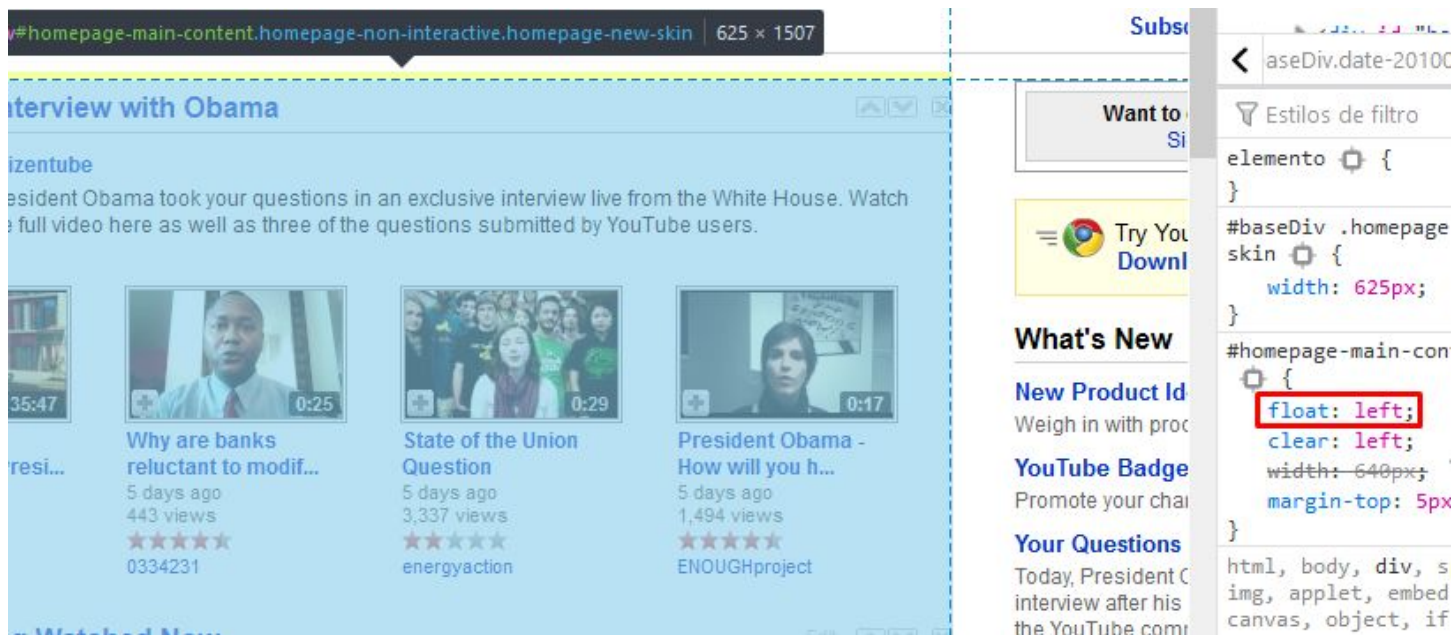
Se usaban mucho las tablas...



Fuente: <https://web.archive.org/web/20050810030357/http://www.youtube.com/>

ALGÚN TIEMPO ATRÁS (NO TANTO)...

Usando el elemento Float:



The image shows a screenshot of a YouTube homepage from 2010, with a CSS developer tool open on the right. The browser's address bar shows the URL `#homepage-main-content.homepage-non-interactive.homepage-new-skin` and the dimensions `625 x 1507`. The page content includes a header for "Interview with Obama" and a grid of video thumbnails. The developer tool on the right shows the CSS for the `#homepage-main-content` element, with the `float: left;` property highlighted in a red box.

Interview with Obama

President Obama took your questions in an exclusive interview live from the White House. Watch the full video here as well as three of the questions submitted by YouTube users.

Why are banks reluctant to modify...

State of the Union Question

President Obama - How will you handle...

What's New

New Product Ideas

YouTube Badge

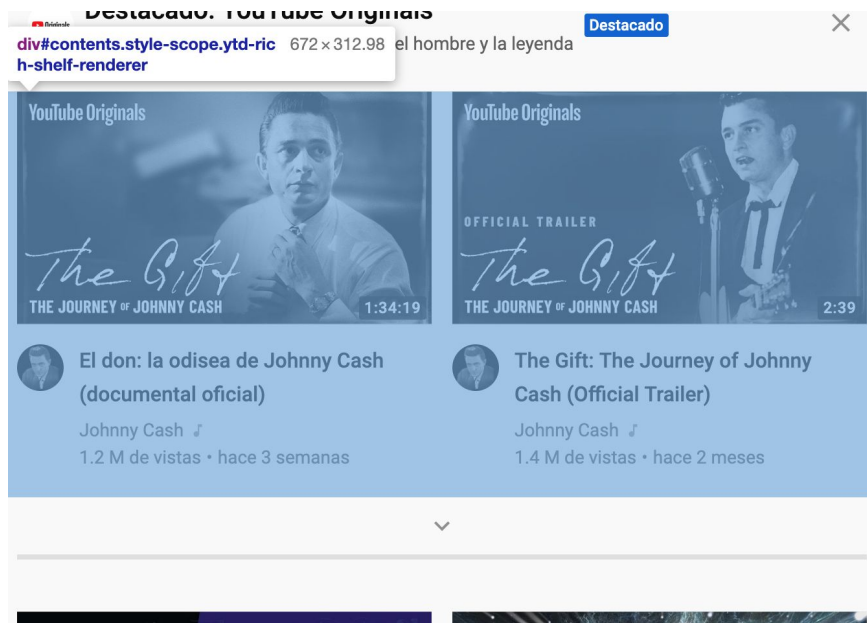
Your Questions

```
aseDiv.date-20100
Estilos de filtro
elemento {
}
#baseDiv .homepage
skin {
  width: 625px;
}
#homepage-main-con
{
  float: left;
  clear: left;
  width: 640px;
  margin-top: 5px;
}
html, body, div, s
img, applet, embed
canvas, object, if
```

Fuente: <https://web.archive.org/web/20100202095628/http://www.youtube.com/>

PRESENTE

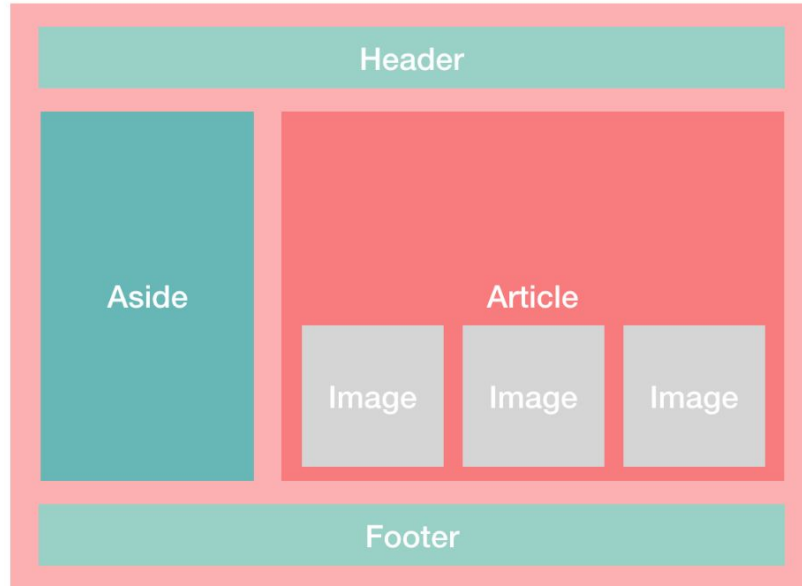
Aplicando Flexbox...



```
... renderer">...</div>
... ><div id="contents"
... class="style-scope yt
... rich-shelf-renderer">
... </div> == $0
... ><div id="button-
... div#contents.style-scope.ytd-rich-shelf-renderer
Styles Computed Event Listeners >>
Filter :hov .cls
element.style {
}
#contents.ytd-rich-shelf-renderer {
  overflow-x: auto;
  margin-left: calc(-1 * 16px / 2);
  margin-right: calc(-1 * 16px / 2);
  display: flex;
  -ms-flex-direction: row;
  -webkit-flex-direction: row;
  flex-direction: row;
  -ms-flex-wrap: wrap;
  -webkit-flex-wrap: wrap;
  flex-wrap: wrap;
}
```

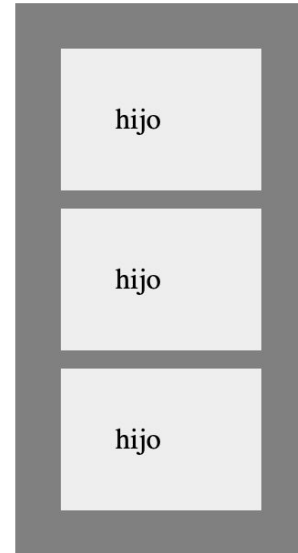
OBJETIVO

Vamos a crear una estructura así:

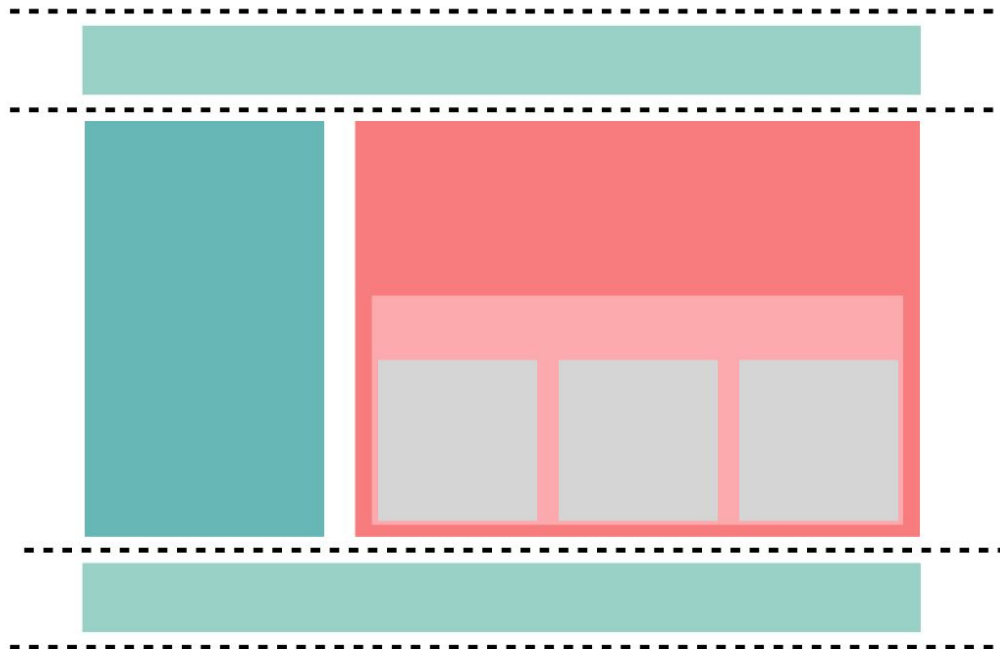


ENTONCES...

Para alcanzar este objetivo, vamos a necesitar seccionarlo en filas o columnas. En este caso, serán filas:



ENTONCES...



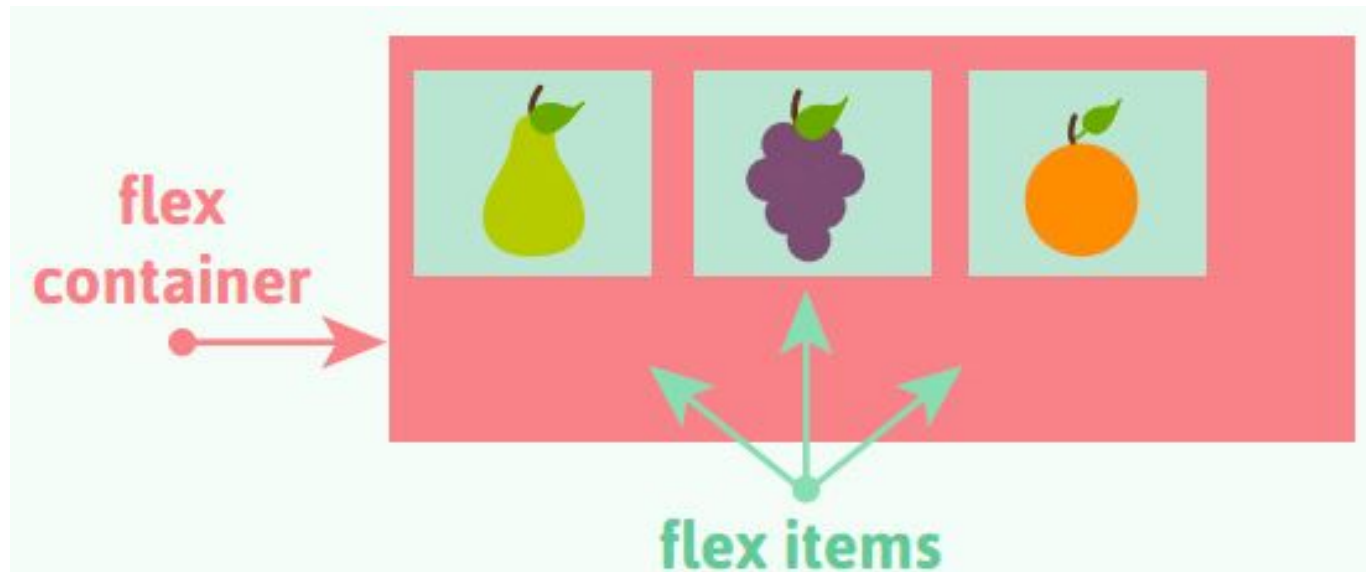
PROPIEDADES DE PADRES E HIJOS

FLEXBOX

Propiedades para aplicar en el contenedor flexible (el **padre**)

- display: *flex* (el que inicia): indicará que sus hijos serán “flexibles”.
- flex-direction: elegir dirección **vertical** u **horizontal**.
- flex-wrap: ¿se hará multilínea cuando llegue al límite?
- flex-flow: abreviación de propiedades
- justify-content: alinear horizontalmente a los hijos si el padre es “fila” o verticalmente si el padre es “columna”
- align-items: alinea verticalmente a los hijos (si el padre es “columna”)
- align-content: alinea verticalmente a los hijos cuando son **multilínea**

PROPIEDADES DEL PADRE



¿CÓMO EMPEZAMOS?

Lo primero que debemos hacer es establecer la propiedad display con el valor flex en el elemento padre.

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

FLEX-DIRECTION: ROW

Esta propiedad nos va a permitir especificar si queremos que los flex items se dispongan en filas o columnas.

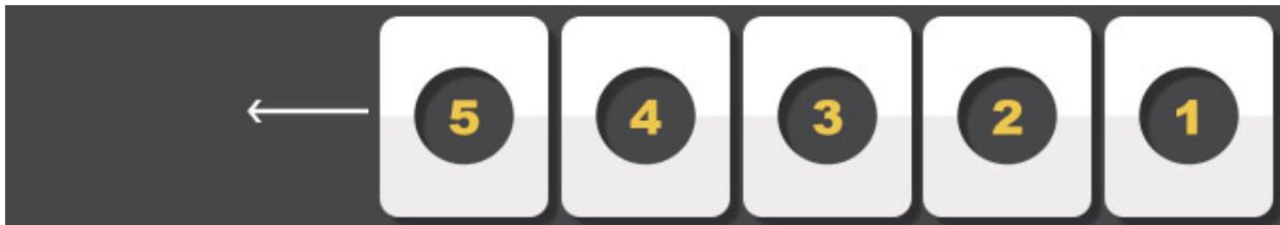
```
.padre-flex {  
  display: flex;  
  flex-direction: row; /* predeterminado */  
}
```



FLEX-DIRECTION: ROW-REVERSE

Con el valor **row-reverse** (fila inversa) los flex items se apilan en una fila de derecha a izquierda.

```
.padre-flex {  
  display: flex;  
  flex-direction: row-reverse;  
}
```



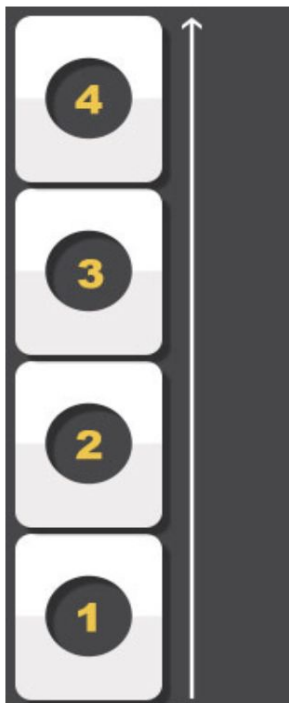
FLEX-DIRECTION: COLUMN



Con el valor **column**, los flex items se apilan en una columna de arriba hacia abajo.

```
.padre-flex {  
  display: flex;  
  flex-direction: column;  
}
```

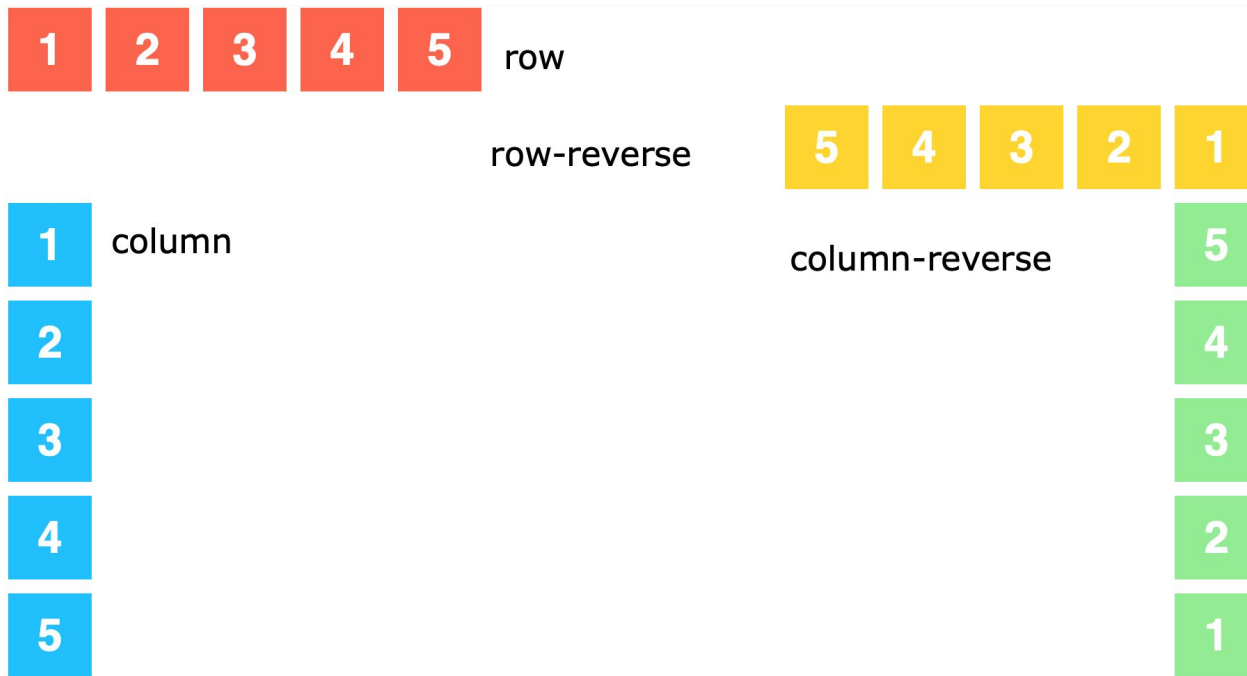

FLEX-DIRECTION: COLUMN-REVERSE



Con el valor `column-reverse` los flex items se apilan en una columna de abajo hacia arriba

```
.padre-flex {  
  display: flex;  
  flex-direction: column-reverse;  
}
```

FLEX-DIRECTION - RESUMEN



Ejemplo
en vivo



¡VAMOS A PRACTICAR LO VISTO!

FLEX-WRAP

El comportamiento inicial del contenedor flexible es poder mantener los flex items en su eje, sin importar que las dimensiones de los mismos cambien.

Con *flex-wrap* vamos a poder especificar si queremos que los ítems puedan saltar a una nueva línea, cuando el contenedor flexible se quede sin espacio.

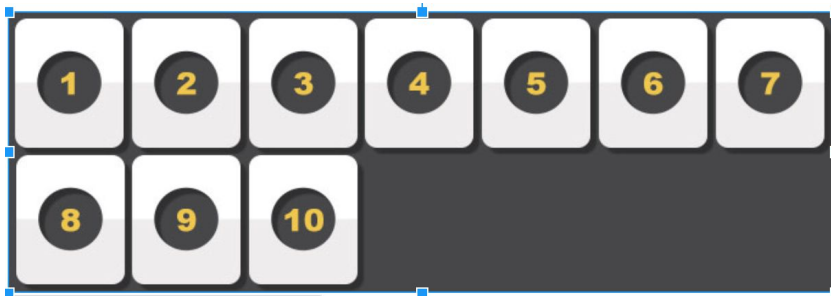
FLEX-WRAP: NOWRAP

```
.padre-flex {  
  display: flex;  
  flex-wrap: nowrap; /* predeterminado */  
}
```



FLEX-WRAP: WRAP

Los *flex items* (*hijos*) pueden romper la línea del eje horizontal, si les es necesario para conservar las características de sus dimensiones. Esto es de izquierda a derecha, y de arriba a abajo.



```
.padre-flex {  
  display: flex;  
  flex-wrap: wrap;  
}
```

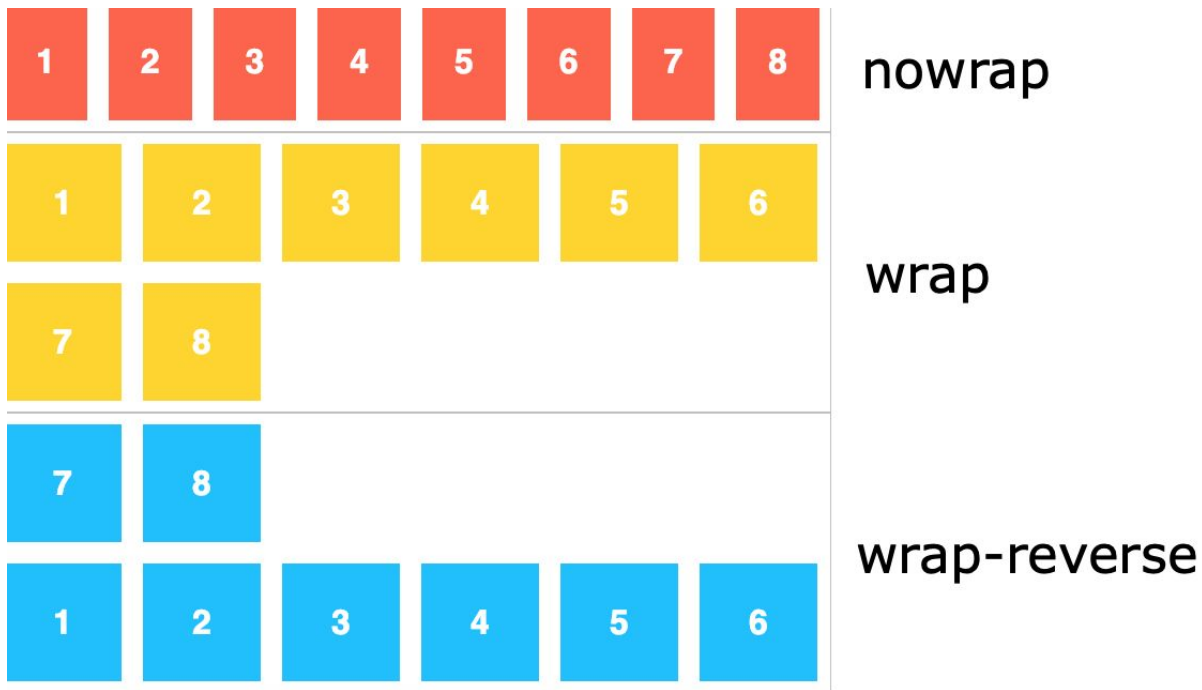
FLEX-WRAP: WRAP-REVERSE

Esta vez el orden es de izquierda a derecha, y de abajo a arriba.



```
.padre-flex {  
  display: flex;  
  flex-wrap: wrap-reverse;  
}
```

FLEX-WRAP - RESUMEN



Ejemplo
en vivo



¡VAMOS A PRACTICAR LO VISTO!

FLEX-FLOW

Es la forma abreviada (shorthand) o rápida para las propiedades:

- *flex-direction*
- *flex-wrap*

Se pone primero la propiedad de *flex-direction*, y luego la de *flex-wrap*.

```
.padre-flex {  
  display: flex;  
  flex-flow: row nowrap;  
}
```

JUSTIFY-CONTENT

- *Justify-content* nos va a permitir alinear los elementos.
- Esto puede ser de forma vertical u horizontal, según lo especifiquemos con *flex-direction*.
- Nos va a ayudar a distribuir los *flex items* (*hijos*) en el *contenedor* (*padre*), cuando los ítems no utilicen todo el espacio disponible en su eje principal actual.

JUSTIFY-CONTENT

Los siguientes ejemplos parten de la base del contenedor en “row”:

```
.padre-flex {  
  display: flex;  
  flex-direction: row;  
}
```

JUSTIFY-CONTENT: FLEX-START

Consiste en alinear los *flex items* (*hijos*) al lado izquierdo.

```
.padre-flex {  
  display: flex;    flex-direction: row;  
  justify-content: flex-start; /* predeterminado */  
}
```



JUSTIFY-CONTENT: FLEX-END

Consiste en alinear los *flex items* (hijos) al lado derecho.

```
.padre-flex {  
  display: flex;  flex-direction: row;  
  justify-content: flex-end;  
}
```



JUSTIFY-CONTENT: CENTER

Consiste en alinear los *flex items* (*hijos*) al centro.

```
.padre-flex {  
  display: flex;  flex-direction: row;  
  justify-content: center;  
}
```



JUSTIFY-CONTENT: SPACE-BETWEEN

Es hacer que los *flex items* (*hijos*) tomen la misma distancia o espaciado entre ellos dentro del contenedor flexible, quedando el primer y último elemento pegados con los bordes del contenedor en el eje principal.

```
.padre-flex {  
  display: flex; flex-direction: row;  
  justify-content: space-between;  
}
```



JUSTIFY-CONTENT: SPACE-AROUND

Muestra los *flex items* (hijos) con el mismo espacio de separación entre sí.
El espaciado entre los bordes lo toman del contenedor padre.

```
.padre-flex {  
  display: flex; flex-direction: row;  
  justify-content: space-around;  
}
```



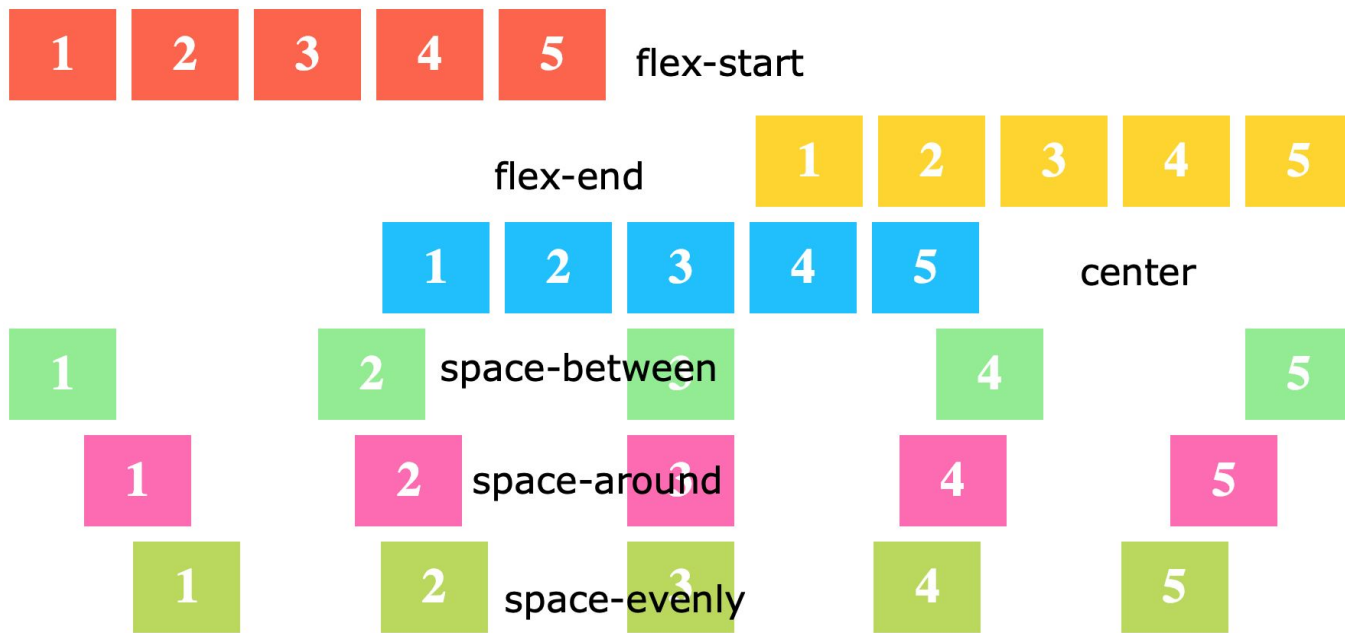
JUSTIFY-CONTENT: SPACE-EVENLY

Hace que el espacio entre los *flex items* (*hijos*) sea igual. **No es lo mismo que space-around.**

```
.padre-flex {  
  display: flex; flex-direction: row;  
  justify-content: space-evenly;  
}
```



JUSTIFY-CONTENT - RESUMEN



Ejemplo
en vivo



¡VAMOS A PRACTICAR LO VISTO!

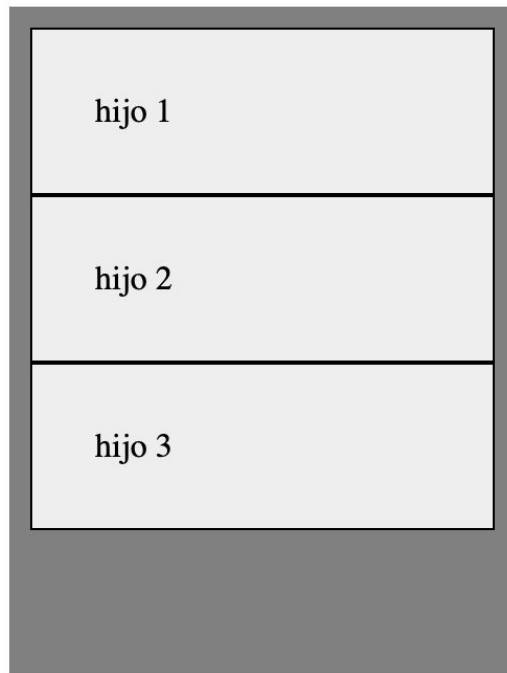
JUSTIFY-CONTENT

Los siguientes ejemplos parten de la base del contenedor en “column”, con altura definida:

```
.padre-flex {  
  display: flex;  
  flex-direction: column;  
}
```

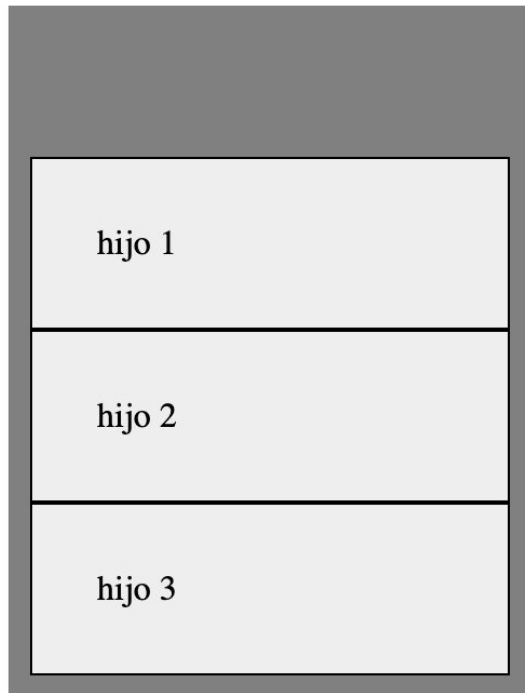
JUSTIFY-CONTENT: FLEX-START

```
.padre-flex {  
  display: flex;  
  flex-direction: column;  
  justify-content: flex-start;  
  height: 300px;  
}
```



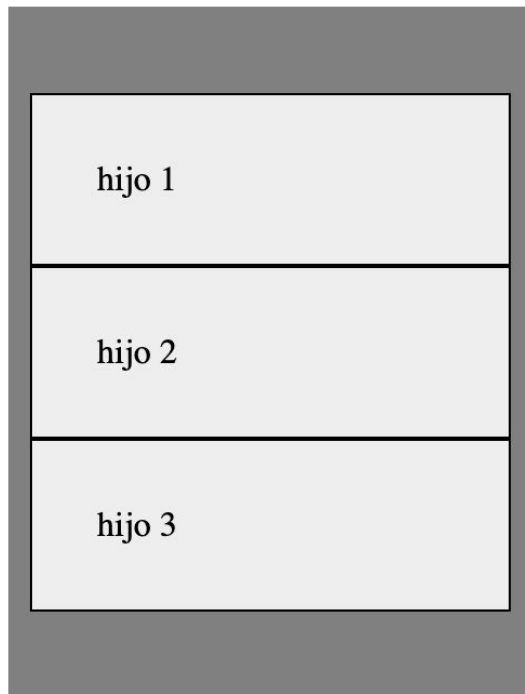
JUSTIFY-CONTENT: FLEX-END

```
.padre-flex {  
  display: flex;  
  flex-direction: column;  
  justify-content: flex-end;  
  height: 300px;  
}
```



JUSTIFY-CONTENT: CENTER

```
.padre-flex {  
  display: flex;  
  flex-direction: column;  
  justify-content: center;  
  height: 300px;  
}
```



JUSTIFY-CONTENT: SPACE-BETWEEN

```
.padre-flex {  
  display: flex;  
  flex-direction: column;  
  justify-content: space-between;  
  height: 300px;  
}
```



JUSTIFY-CONTENT: SPACE-AROUND

```
.padre-flex {  
  display: flex;  
  flex-direction: column;  
  justify-content: space-around;  
  height: 300px;  
}
```



JUSTIFY-CONTENT: SPACE-EVENLY

```
.padre-flex {  
  display: flex;  
  flex-direction: column;  
  justify-content: space-evenly;  
  height: 300px;  
}
```



Ejemplo
en vivo

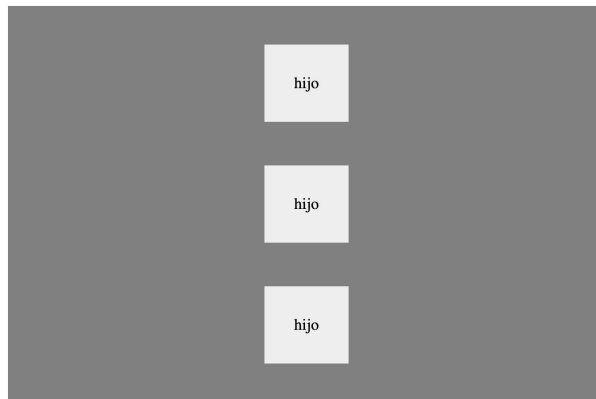


¡VAMOS A PRACTICAR LO VISTO!

ALIGN-ITEMS

- Alinear los elementos verticales de forma horizontal.

(flex-direction: column)



- Alinear los elementos horizontales de forma vertical.

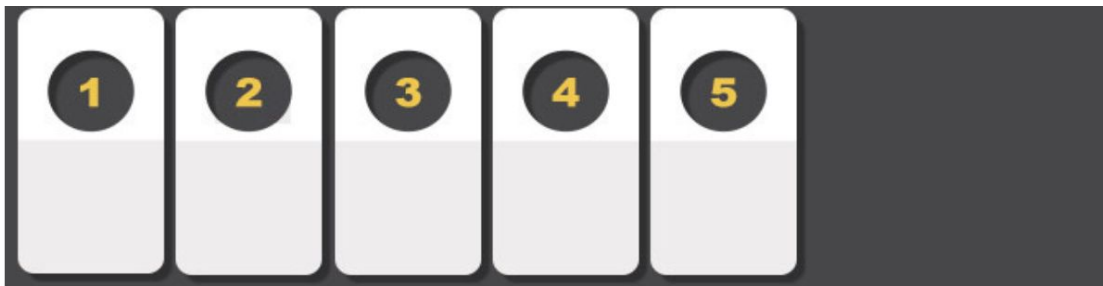
(flex-direction: row)



ALIGN-ITEMS: STRETCH

Tratará de llenar toda la altura (o anchura) del contenedor, siempre y cuando los *hijos* no tengan propiedades de dimensión definidas.

```
.padre-flex {  
  display: flex; flex-direction: row;  
  align-items: stretch;  
}
```



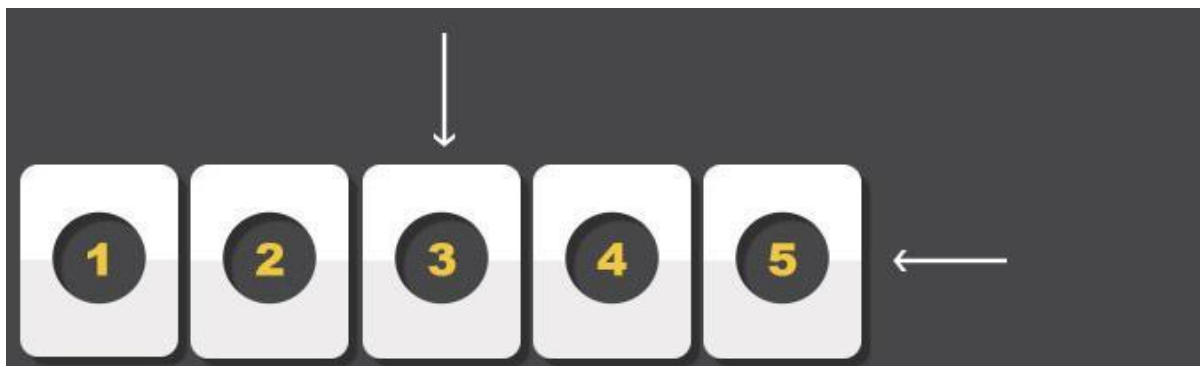
ALIGN-ITEMS: FLEX-START

```
.padre-flex {  
  display: flex; flex-direction: row;  
  align-items: flex-start; /* predeterminado */  
}
```



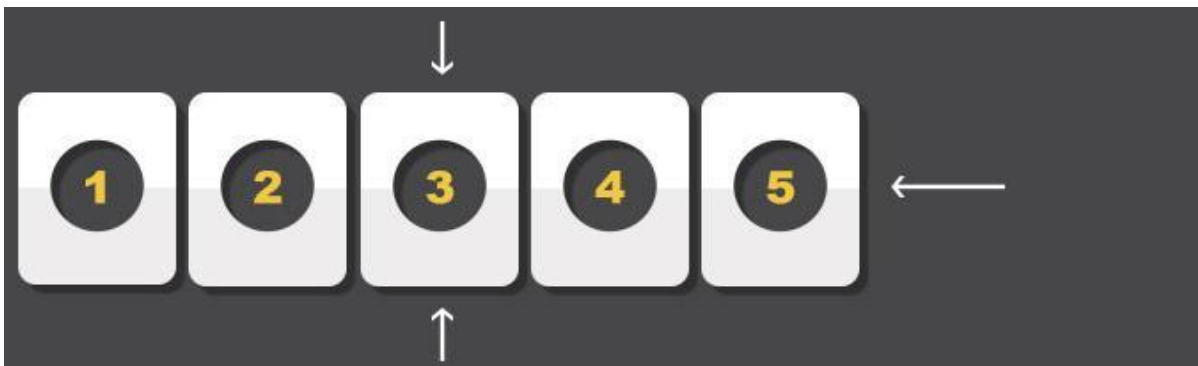
ALIGN-ITEMS: FLEX-END

```
.padre-flex {  
  display: flex; flex-direction: row;  
  align-items: flex-end; /* predeterminado */  
}
```



ALIGN-ITEMS: CENTER

```
.padre-flex {  
  display: flex; flex-direction: row;  
  align-items: center; /* predeterminado */  
}
```

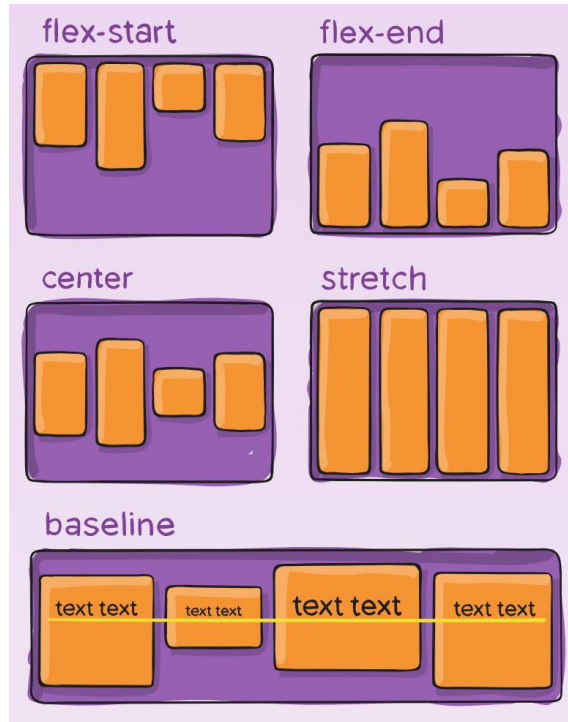


ALIGN-ITEMS: BASELINE

```
.padre-flex {  
  display: flex; flex-direction: row;  
  align-items: baseline; /* predeterminado */  
}
```



ALIGN-ITEMS - RESUMEN



Ejemplo
en vivo



¡VAMOS A PRACTICAR LO VISTO!

ALIGN-CONTENT

Esta propiedad sólo tiene efecto cuando el contenedor flexible tiene varias líneas de *flex items* (*hijos*). Si se colocan en una sola línea, esta propiedad no tiene ningún efecto sobre el diseño.

Para poder aplicarlo se necesita tener el atributo *flex-wrap*, que permita verificar los ejes horizontales.

```
.padre-flex {  
  display: flex; flex-wrap: wrap; flex-direction: row;  
  align-content: stretch; /* predeterminada */  
}
```

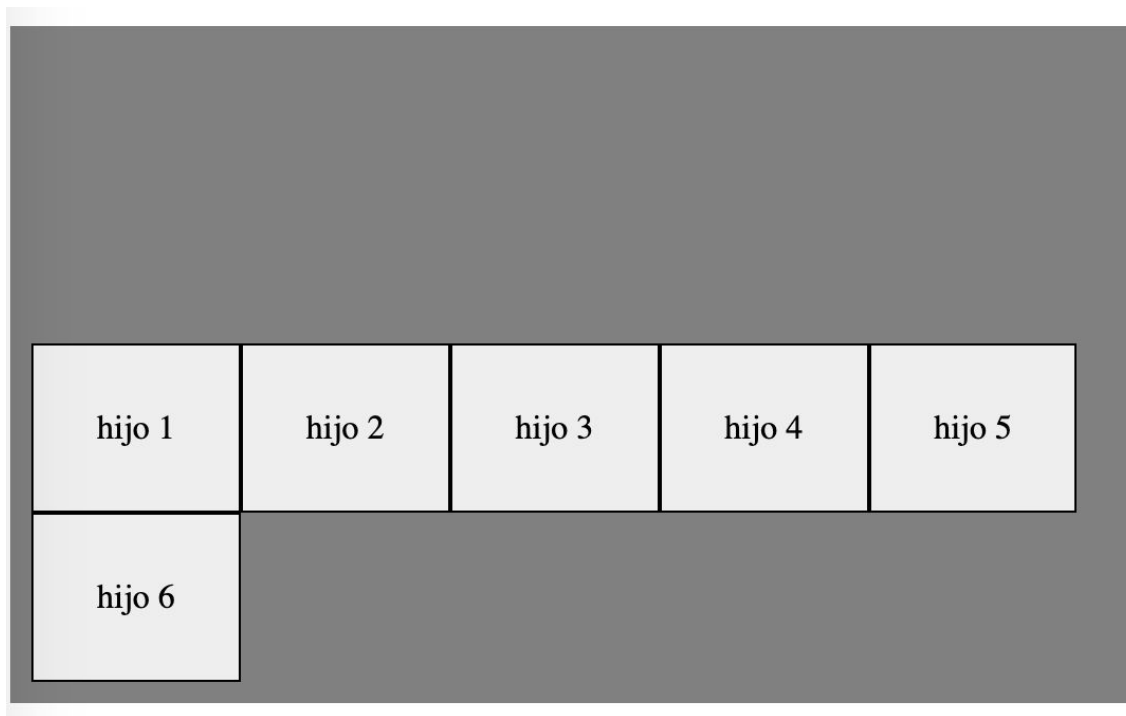
ALIGN-CONTENT: STRETCH

hijo 1	hijo 2	hijo 3	hijo 4	hijo 5
hijo 6				

ALIGN-CONTENT: FLEX-START



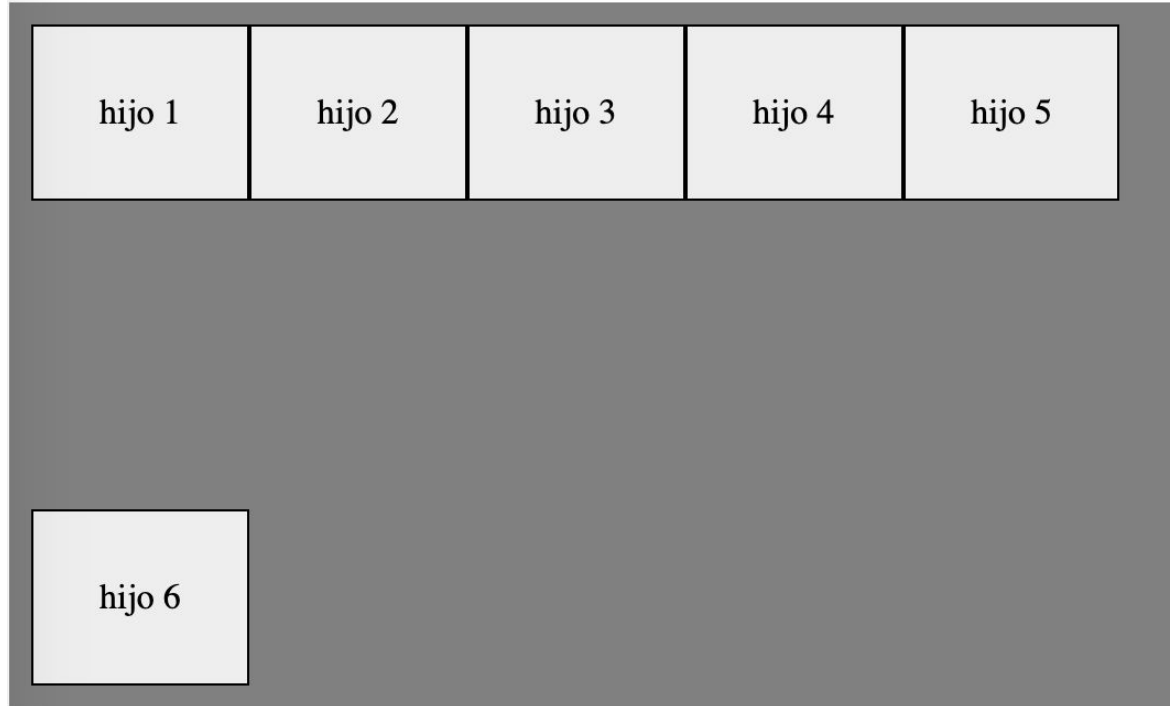
ALIGN-CONTENT: FLEX-END



ALIGN-CONTENT: CENTER

hijo 1	hijo 2	hijo 3	hijo 4	hijo 5
hijo 6				

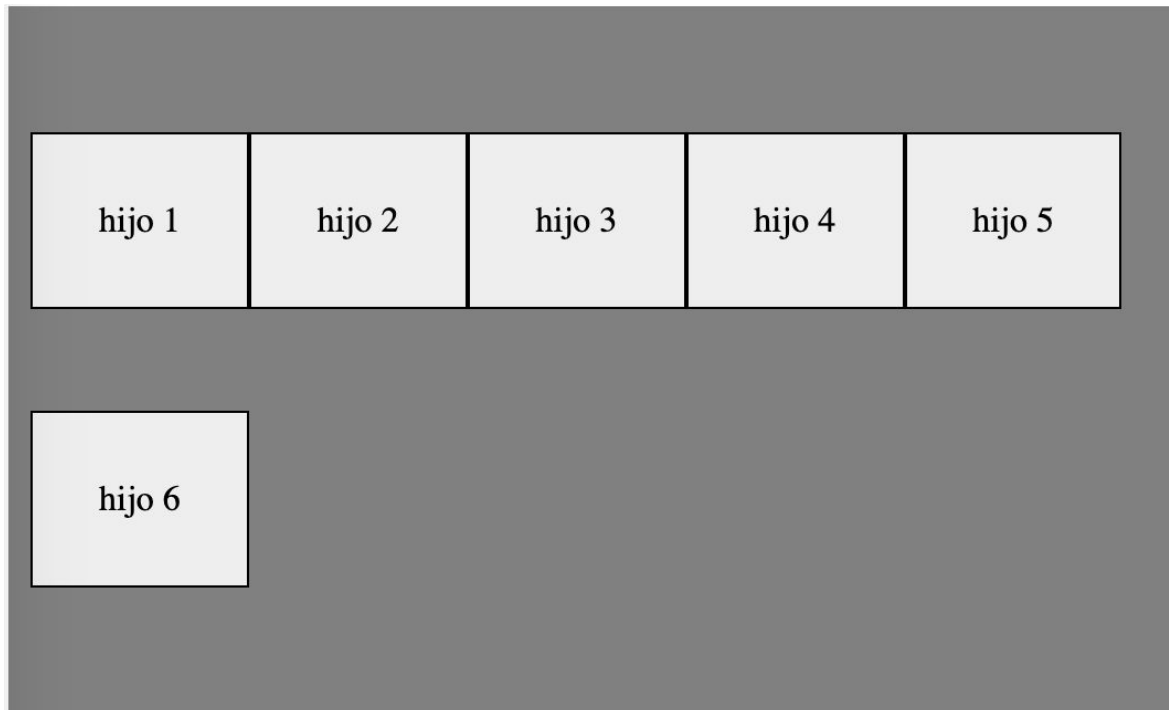
ALIGN-CONTENT: SPACE-BETWEEN



ALIGN-CONTENT: SPACE-AROUND



ALIGN-CONTENT: SPACE-EVENLY

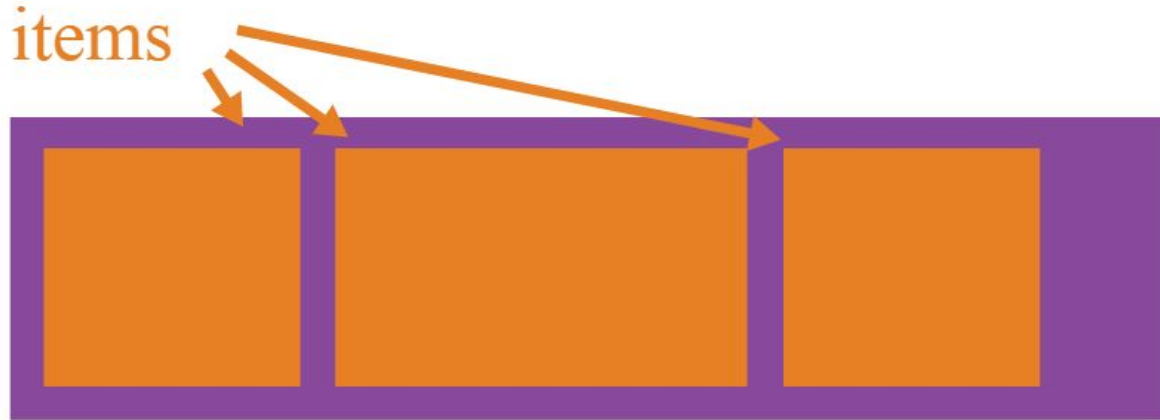


Ejemplo
en vivo



¡VAMOS A PRACTICAR LO VISTO!

PROPIEDADES PARA LOS FLEX ITEMS (HIJOS)



ORDER

Esta propiedad permite **modificar el orden de aparición** de un elemento. Recibe como valor numeros enteros.

```
.hijo-flex {  
  order: -1;  
}
```



Ejemplo
en vivo



¡VAMOS A PRACTICAR LO VISTO!

FLEX-BASIS

(parecido al width o height)

Define el **ancho** de un elemento **inicial**. Este valor por defecto viene configurado en “auto”. Actúa como “máximo” o “mínimo”, al usar flex-grow o flex-shrink.

```
.hijo-flex {  
/* si esta en “row” es como el “width”, si está en  
“column” es cómo “height” */  
  flex-basis: 100px;  
}
```

FLEX-GROW

- Esta propiedad **define la capacidad de un elemento de crecer**, cuando en el contenedor todavía hay espacio sobrante.
- Se configura con un valor numérico entero natural (no acepta negativos).
- Por defecto, el valor viene configurado en "0", por lo tanto el elemento no crecerá de manera horizontal.
- Si este valor es configurado en 1 para todos los ítems, todos estos crecerán de igual manera, por lo que ocuparán la misma cantidad de espacio dentro del contenedor.

FLEX-GROW - HACIENDO CÁLCULOS...

Esto es algo complejo. Veamos cómo
hacer cálculos para usar esta
propiedad...

FLEX-GROW - HACIENDO CÁLCULOS...

Imaginemos que:	El cálculo:
<ul style="list-style-type: none">● El “padre” tiene un ancho de 400px● Tiene 3 hijos, cada uno 100px	Total entre los 3 hijos: 300px Espacio disponible: 100px (400-300)
¿Cuántos tienen la propiedad flex-grow definida?	(espacio disponible dividido la cantidad de “grow” que hay) $100 / 1 = 100\text{px}$ cada fracción (el hijo con valor 1, se lleva $1 \times 100\text{px} = 100\text{px}$)
Hay solo un hijo, y tiene como valor 1.	Hijo con valor 1: $1/1 = 100\text{px}$

FLEX-GROW - HACIENDO CÁLCULOS...

```
.hijo-flex {  
  flex-grow: 0;  
}  
.hijo-flex.mayor {  
  flex-grow: 1;  
}
```

100px

200px

100px

FLEX-GROW - OTRO EJEMPLO

Imaginemos que:	El cálculo (vital calcular cuántos “grow” hay):
<ul style="list-style-type: none">● El “padre” tiene un ancho de 500px● Tiene 2 hijos, cada uno 100px	Total entre los 2 hijos: 200px Espacio disponible: 300px (500-200)
¿Cuántos tienen la propiedad flex-grow definida?	$300 / 3 = 100\text{px}$ cada fracción (el hijo con valor 2, se lleva $2 \times 100\text{px} = 200\text{px}$)
Un hijo tiene “flex-grow: 2” y el otro “flex-grow: 1”	Hijo con valor 2: $2/3 = 200\text{px}$ Hijo con valor 1: $1/3 = 100\text{px}$

FLEX-GROW - HACIENDO CÁLCULOS...

```
.hijo-flex {  
  flex-grow: 1;  
}  
.hijo-flex.mayor {  
  flex-grow: 2;  
}
```

100px

200px

FLEX-GROW - HACIENDO CÁLCULOS...

Usemos la siguiente herramienta para calcular:

<https://www.madebymike.com.au/demos/flexbox-tester/>

FLEX-SHRINK

Básicamente es lo mismo que flex-grow,
pero con el espacio faltante.

FLEX-SHRINK

Imaginemos que:	El cálculo (vital calcular cuantos “grow” hay):
<ul style="list-style-type: none">● El “padre” tiene un ancho de 600px● Tiene 3 hijos, cada uno 300px.	Total entre los 3 hijos: 900px Espacio faltante: -300px
¿Cuántos tienen la propiedad flex-shrink definida?	$300 / 3 = 100\text{px}$ se debe reducir cada hijo.
Todos tienen la propiedad.	Por lo tanto, se reducen todos por igual, ya que el valor por defecto es 1.

UNIFICACIÓN

Podemos unificar estos tres estilos mediante el atributo “flex” de la siguiente manera:

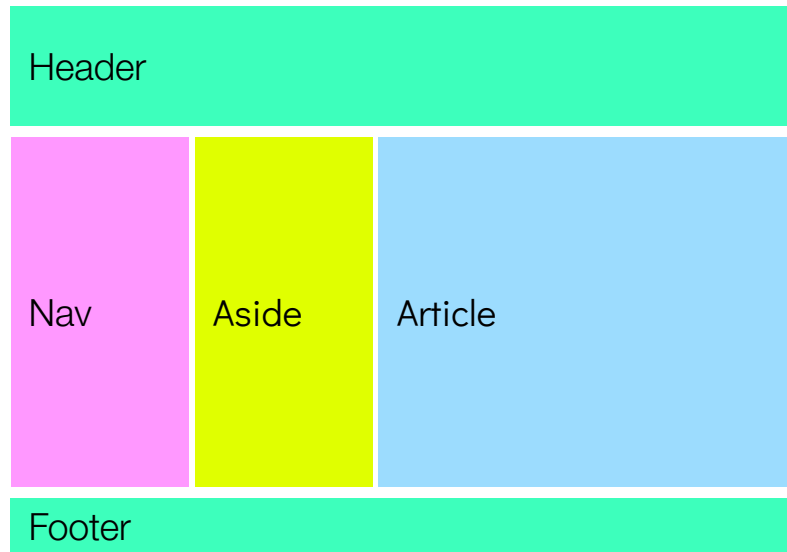
```
div {  
  flex-grow: 1;  
  flex-shrink: 1;  
  flex-basis: 0%;  
}
```

=

```
div {  
  flex: 1;  
}
```

PÁGINA CON FLEXBOX

Con lo aprendido en esta clase, podríamos hacer un layout de página con la siguiente estructura, mediante flexbox:



PÁGINA CON FLEXBOX

HTML

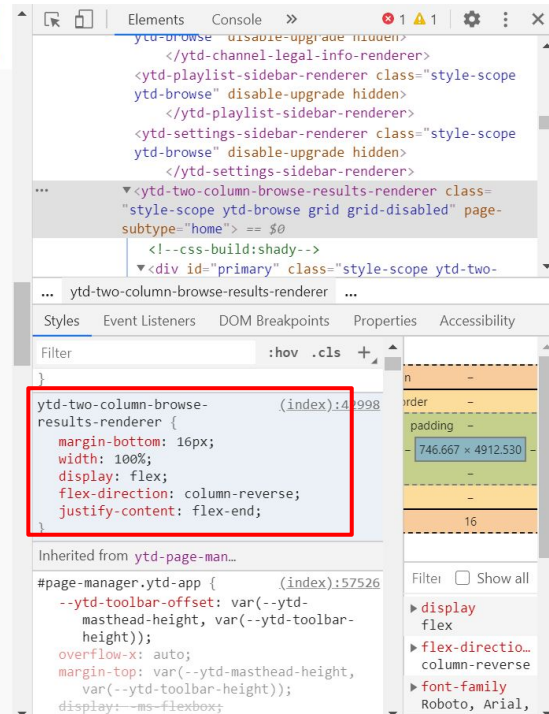
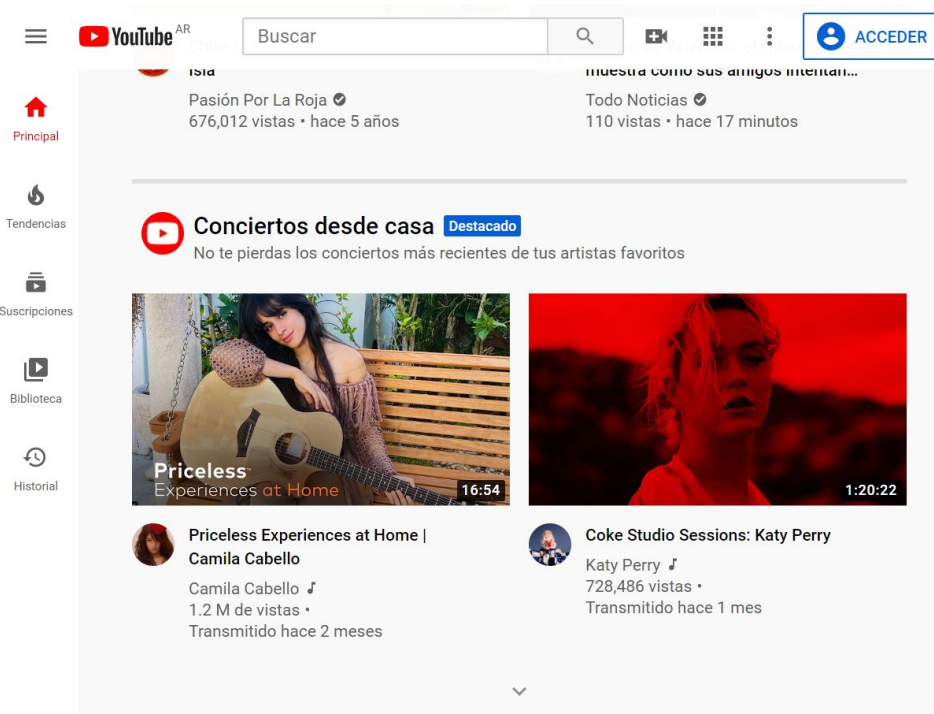
```
<header>Header</header>
<main>
  <nav>Nav</nav>
  <aside>Aside</aside>
  <article>Article
</article>
</main>
<footer>Footer</footer>
```

CSS

```
body {
  display: flex;
  height: 600px;
  flex-direction: column;
  padding: 1em;}
main{
  display: flex;
  flex: 1;}
article, nav, aside{
  background-color: pink;}
header, footer{
  background-color: grey;
  margin: 1px;}
article, nav, aside, header, footer{
  flex: 1;
  padding: 1rem;
  margin: 1px;}
```

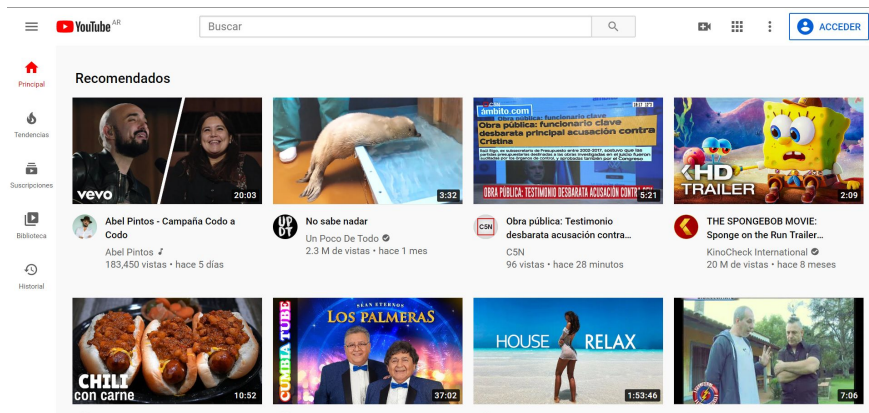
PÁGINA CON FLEXBOX

Ejemplos

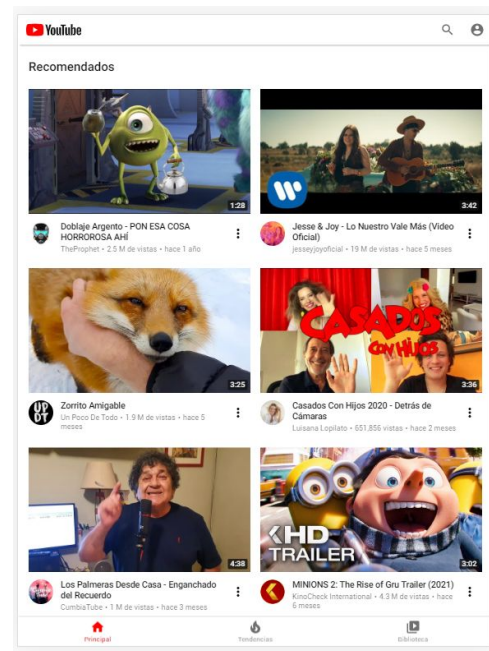


PÁGINA CON FLEXBOX

Ejemplos



Desktop



Tablet



Mobile

PÁGINA CON FLEXBOX

Ejemplos

The screenshot displays the Amazon Argentina homepage, which is a practical example of a web page using Flexbox for its layout. The page is divided into several sections:

- Header:** Includes the Amazon logo, a search bar, and navigation links like "Cuenta y Listas" and "Devoluciones y Pedidos".
- Hero Section:** A green banner with text about delivery options in Argentina.
- Compras por Categoría:** A grid of four categories: Computadoras, Videojuegos, Bebé, and Juguetes y Juegos. Each category has a representative image and a "Ver más" link.
- AmazonBasics:** A section featuring a coiled USB cable with a "Ver más" link.
- Electrónicos:** A section featuring a collection of electronic items like a camera, headphones, and a laptop, with a "Ver más" link.

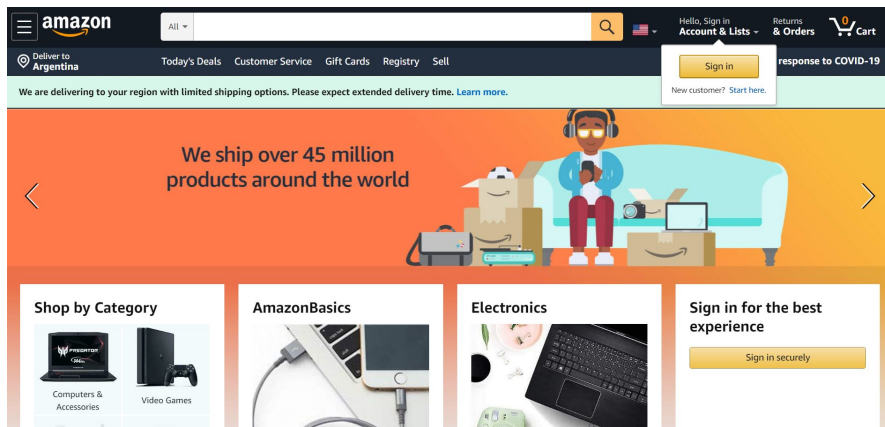
On the right side of the image, a browser's developer tools are open, showing the CSS styles for a specific element. The styles are:

```
.fluid-card {  
  display: flex;  
  -webkit-box-orient: vertical;  
  -webkit-box-direction: normal;  
  flex-direction: column;  
}
```

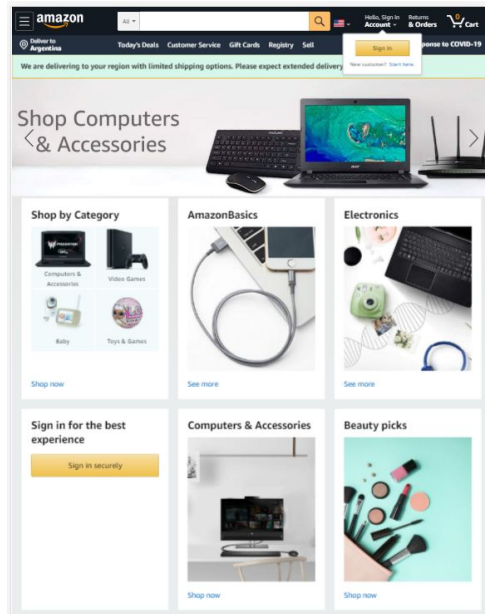
This CSS block is highlighted with a red rectangle, indicating the Flexbox configuration used for the category grid.

PÁGINA CON FLEXBOX

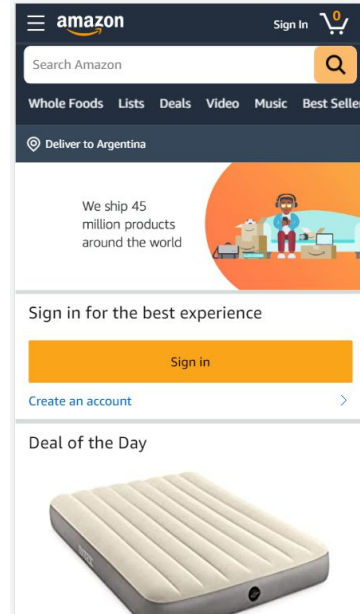
Ejemplos



Desktop



Tablet



Mobile

PÁGINA CON FLEXBOX

Ejemplos



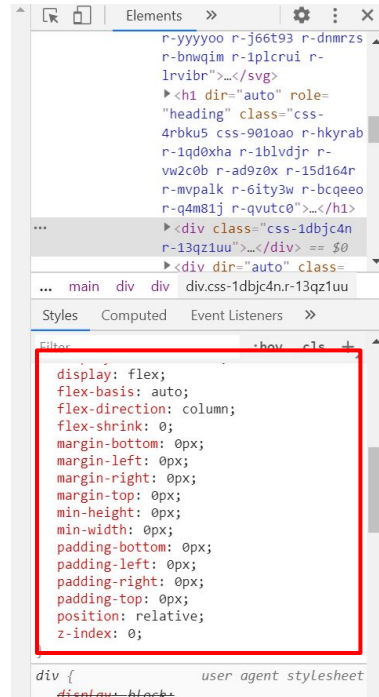
Iniciar sesión en Twitter

Teléfono, correo o usuario

Contraseña

Iniciar sesión

[¿Olvidaste tu contraseña?](#) · [Regístrate en Twitter](#)





APLICAR FLEXBOX

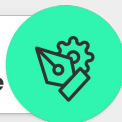
Modela con flexbox alguna sección de tu Proyecto.

APLICAR FLEXBOX

Formato: archivo HTML y CSS. Debe tener el nombre “Idea+Apellido”.

Sugerencia: carpeta en formato zip o rar, con el/los archivos HTML y CSS.

Desafío
entregable



>> Consigna: con los conocimientos adquiridos en la clase de hoy, modela con flexbox alguna sección de un archivo html.

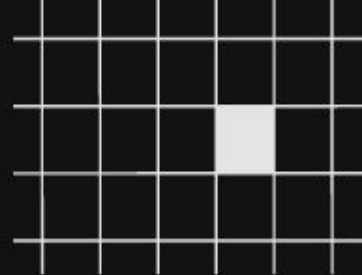
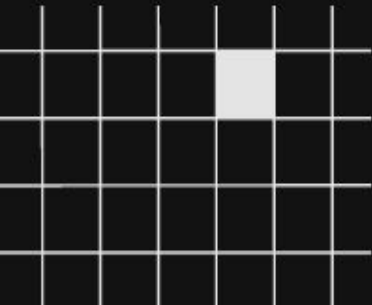
>>Aspectos a incluir en el entregable:

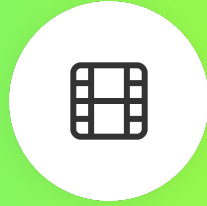
Dentro de una sección o página de tu proyecto, modela con flexbox utilizando alguna/s de las siguientes propiedades: display: flex, flex-direction, flex-wrap, flex-flow, justify-content, align-items, align-content, order, flex-basis, flex-shrink y/o flex-basis.

>>Ejemplo:

[Carpeta comprimida con los archivos de flexbox](#)

¿PREGUNTAS?





***¿QUIERES SABER MÁS? TE DEJAMOS
MATERIAL AMPLIADO DE LA CLASE***



- [Ejemplos de aplicaciones de flexbox para tu proyecto](#) | ***quackit.com***
- [Diferentes estructuras de páginas con flexbox](#) | ***quackit.com***
- [¡Juego para practicar las propiedades!](#) | ***flexboxfroggy.com***
- [Aprende flex jugando \(1\)](#) | ***flexboxdefense.com***
- [Aprende flex jugando \(2\)](#) | ***flexboxdefense.com***
- [Más información sobre Grids](#) | ***css-tricks.com***



¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- Conocer nuevos elementos de flexbox.
 - Aplicar sus diferentes variantes.
- 