



Programación Web Básica.



U
N
Q



Prof. Jorge Nicolás Terradillos
Prof. Fernando Blanco



En la clase de hoy...

-Unidades de medida

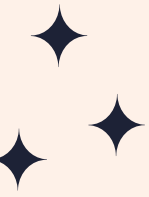
-Modificar prioridad

-Flujo de elementos y display

-Posición relativa y absoluta

-Contenedores flex y grid





01

Unidades de medida





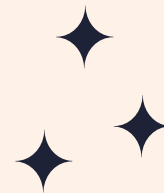
Tipos de unidades

En CSS hay dos formas de medidas (como usamos en img)

- Unidad **absoluta**: pixeles, centímetros
- Unidades **relativas**: relativas a algo más (cómo el tamaño de la pantalla).

| unit | definition |
|------|---------------------------------------|
| 'cm' | centimeters |
| 'mm' | millimeters |
| 'in' | inches; 1in is equal to 2.54cm |
| 'px' | pixels; 1px is equal to 1/96th of 1in |
| 'pt' | points; 1pt is equal to 1/72nd of 1in |
| 'pc' | picas; 1pc is equal to 12pt |

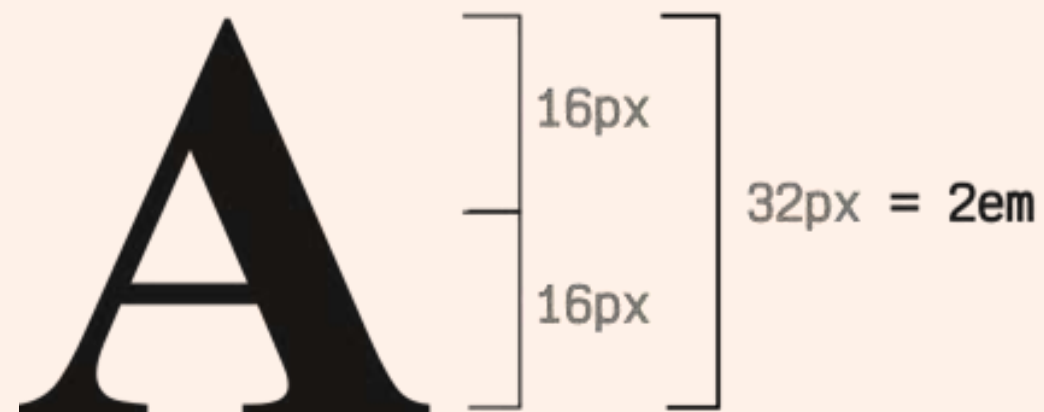




Unidades relativas

Entre las unidades relativas está el porcentaje y la unidad «em»

La unidad **em** es relativa al elemento padre , **vw** es 1% del ancho de la ventana gráfica y **vh** es 1% del alto.



Unidades relativas

Porcentaje

Relativa al tamaño del elemento «padre», siendo 100% el ancho total del padre

Em

Relativa al tamaño de la fuente del elemento padre

Vh

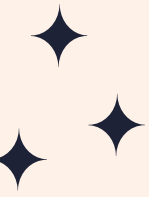
Cada punto equivale a un 1% del tamaño de la ventana del navegador

Rem

Relativo al tamaño de fuente del elemento raíz

Vw

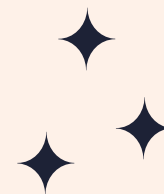
Cada punto equivale al 1% del ancho de la ventana del navegador



02

Prioridad de estilos





Herencia

Como vimos la otra clase, las propiedades de elementos padre se heredan a los hijos

```
<div id="texto">
  <div id="primera-seccion">
    <p>Lorem ipsum dolor sit amet,
      consectetur adipiscing elit, sed
      do eiusmod tempor incididunt.</p>
  </div>
  <div id="segunda-seccion">
    <p>Viverra adipiscing at in tellus
      integer feugiat scelerisque. Sed
      turpis tincidunt id aliquet.</p>
  </div>
</div>
```

```
#texto{
  color: white;
  background-color: black;
  font-family: Arial;
}

#segunda-seccion{
  background-color: red;
}
```



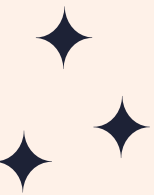

Prioridad tipo vs. id

```
<a href="contacto.html">Mi Contacto</a>  
<a class="sin-subrayado" href="referencias.html">Otras referencias</a>
```

Volviendo al problema de la prioridad, ¿cuál regla tiene prioridad en este caso?

```
a {  
    text-decoration: purple wavy underline;  
}  
  
.sin-subrayado {  
    text-decoration: none;  
}
```





Prioridad id vs class

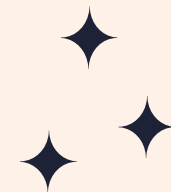
Entre un estilo en id y otro en una clase: ¿cuál prioriza CSS?

Si un elemento tiene #titulo vs .texto-azul ¿Que opción toma CSS de las 2?

CSS siempre prioriza las reglas más **específicas**

```
h1{  
  color: ■ gold;  
  text-align: center;  
  background-color: ■ darkolivegreen;  
}  
  
#titulo{  
  color: ■ red;  
  font-style: italic;  
}  
  
.texto-azul{  
  color: ■ blue  
}
```



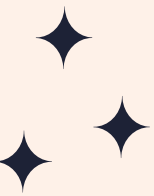


Prioridad en línea vs class





¿Que pasa cuando hay un estilo en línea, id y clase?

```
<h1 id="titulo"  
  class="texto-azul"  
  style="color: ■ green">  
  Mi título  
</h1>
```

```
h1{  
  color: ■ gold;  
  text-align: center;  
  background-color: ■ darkolivegreen;  
}  
  
#titulo{  
  color: ■ red;  
  font-style: italic;  
}  
  
.texto-azul{  
  color: ■ blue  
}
```



Modificar prioridad

```
.texto-azul {  
  color:  blue !important;  
}  
  
.texto-rojo {  
  color:  red;  
}  
  
#titulo {  
  color:  gold;  
}  
  
a {  
  text-decoration:  purple wavy underline;  
}
```

Se puede dar prioridad a una clase o un Id sobre el estilo en línea o cualquier otro estilo con el tag **important**. Al lado de la propiedad ponemos «**!important**»





03

Flujo de elementos





El flujo en HTML

El flujo normal es la forma en que los elementos aparecen por defecto

Por ejemplo, los botones y los links aparecen siempre en la misma línea mientras que los encabezados y párrafos no

Titulo de mi pagina

Menu

[Home](#) [Mi Blog](#) [Mis Redes](#) [Contacto](#)

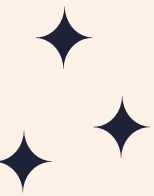
Esta es la presentacion de mi pagina web de imagenes.

Animales

Autos

Paisajes





Flujo inline

Titulo de mi pagina

Menu

[Home](#) [Mi Blog](#) [Mis Redes](#) [Contacto](#)

Esta es la presentacion de **mi pagina web** de imagenes.

[Animales](#)

[Autos](#)

[Paisajes](#)

Elementos como `<a>`, ``, `` son de flujo “**inline**”, aparecen en la misma línea

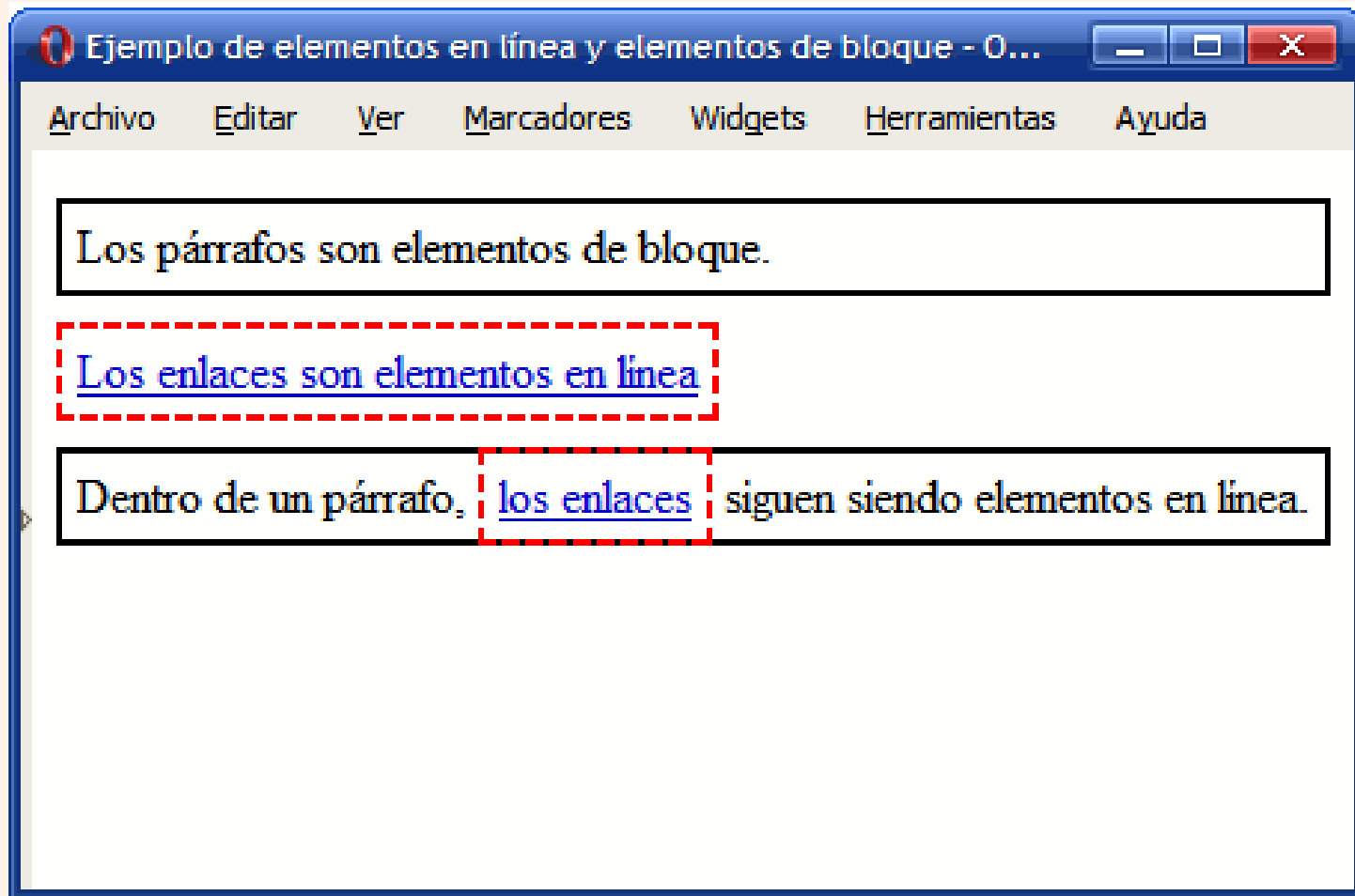
No pueden cambiar su altura (height) y ancho (width)

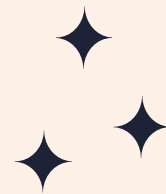




Flujo block

Elementos como `<div>`, `<p>`, `<h1>` son de flujo “**block**”, bloques que aparecen en la siguiente línea, arriba para abajo sucesivamente. Pueden tener atributo `height` y `width` regulable

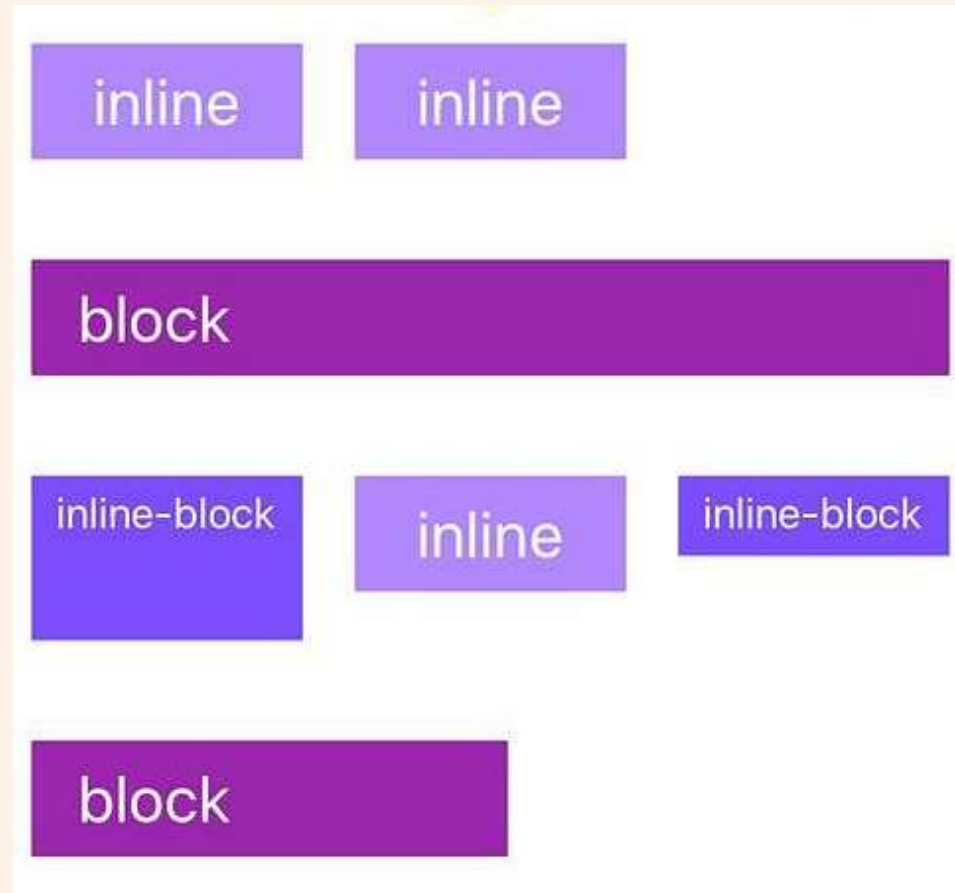


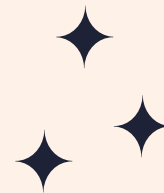


Flujo inline-block

El flujo de botones e imágenes se conoce como “**inline-block**”

Se pueden visualizar en la misma línea pero también pueden cambiar de tamaño

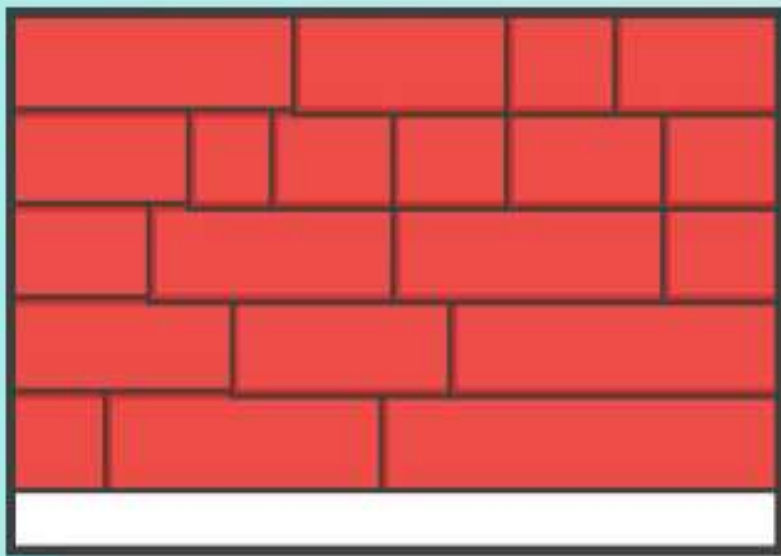




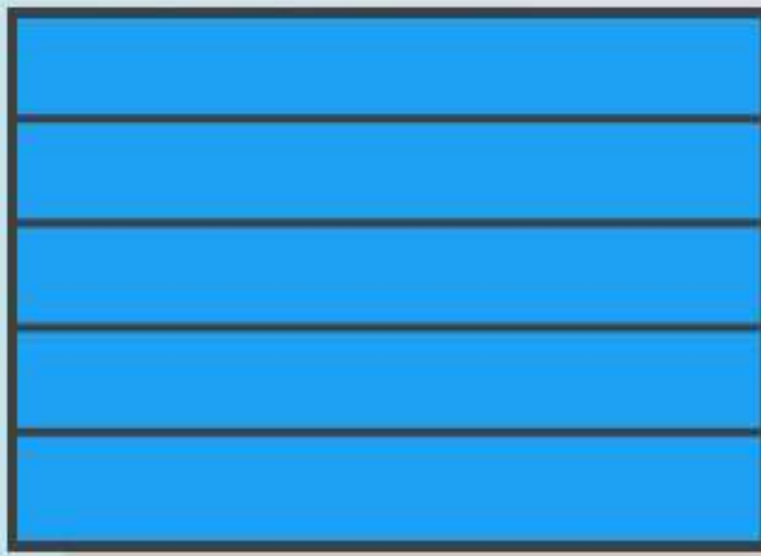
Atributo display

Todos estos flujos son valores de la propiedad **display**

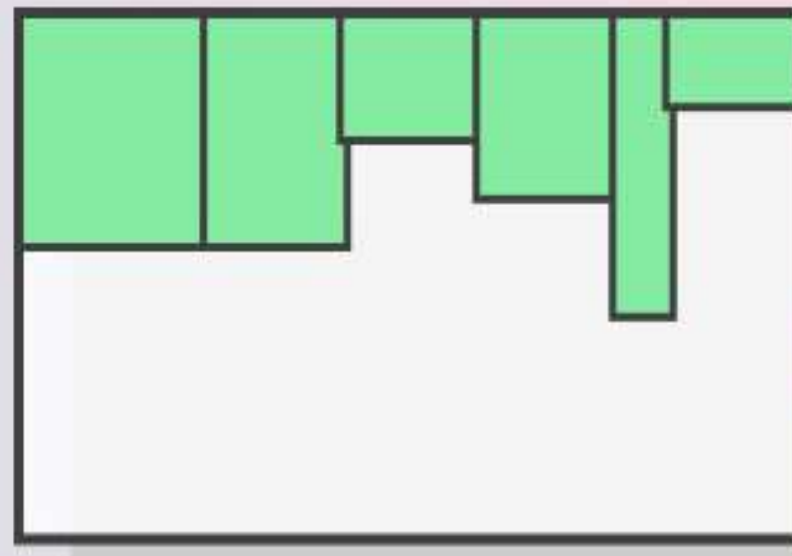
display: inline

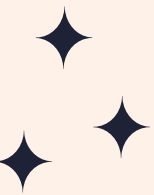


display: block



display: inline-block





Atributo float

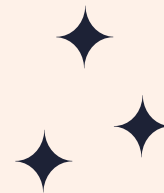
Los inline-block se pueden acomodar relativo a otros elementos. El atributo **float** recibe una dirección (top, right, bottom, left)

```
img{  
  border:solid;  
  border-width: 2px;  
  float: left;  
}
```



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Eu tincidunt tortor aliquam nulla facilisi cras fermentum odio eu. Purus gravida quis blandit turpis. Ultrices tincidunt arcu non sodales. In fermentum et sollicitudin ac orci phasellus egestas tellus rutrum. Sit amet facilisis magna etiam tempor orci. Sit amet aliquam id diam. Ultrices gravida dictum fusce ut placerat orci. Euismod in pellentesque massa placerat dui. Tellus in hac habitasse platea dictumst vestibulum. Nunc vel risus commodo viverra maecenas accumsan.

Justo eget magna fermentum iaculis. Egestas dui id ornare arcu odio ut sem nulla pharetra. Praesent semper feugiat nibh sed pulvinar. Cras fermentum odio eu feugiat pretium nibh ipsum.



Si no queremos que otro elemento se mueva por un esto usamos el atributo **clear** con el mismo valor que float

```
img{  
  border:solid;  
  border-width: 2px;  
  float: left;  
}  
  
#segundo-parrafo{  
  clear: left;  
}
```



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Eu tincidunt tortor aliquam nulla facilisi cras fermentum odio eu. Purus gravida quis blandit turpis. Ultrices tincidunt arcu non sodales. In fermentum et sollicitudin ac orci phasellus egestas tellus rutrum. Sit amet facilisis magna etiam tempor orci. Sit amet aliquam id diam. Ultrices gravida dictum fusce ut placerat orci. Euismod in pellentesque massa placerat dui. Tellus in hac habitasse platea dictumst vestibulum. Nunc vel risus commodo viverra maecenas accumsan.

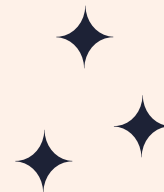
Justo eget magna fermentum iaculis. Egestas dui id ornare arcu odio ut sem nulla pharetra. Praesent semper feugiat nibh sed pulvinar. Cras fermentum odio eu feugiat pretium nibh ipsum.



04

Posición de elementos





Posicion relativa

Para mover un elemento respecto a su posición en el flujo normal debemos cambiar el atributo «**position**» a «**relative**»

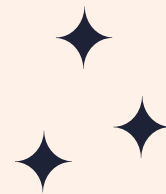
saltemos



No a conclusiones apresuradas

Elegimos una dirección de la que alejarlo y a que distancia lo queremos

```
strong{  
  position:relative;  
  bottom:20px;  
}
```



Posicion absoluta

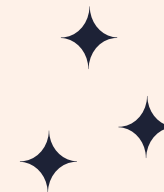
El posicionamiento absoluto es la relación entre el elemento y toda la ventana.

Bienvenidos a mi pagina

[Ver más opciones](#)

Se mide desde el 0px (pegado a uno de los bordes)
en adelante





Usando la propiedad «**position: absolute;**»

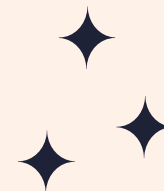
```
.opciones{  
  position:absolute;  
  right:0px;  
  bottom: 30px;  
  border:solid;  
  border-width: 3px;  
  border-color: ■green;  
  background-color: ■aquamarine;  
  border-radius: 30% 0 0 30%;  
  height: 60px;  
}
```

Se usan las direcciones
(top, bottom, right, left)
como propiedades con valores
en px

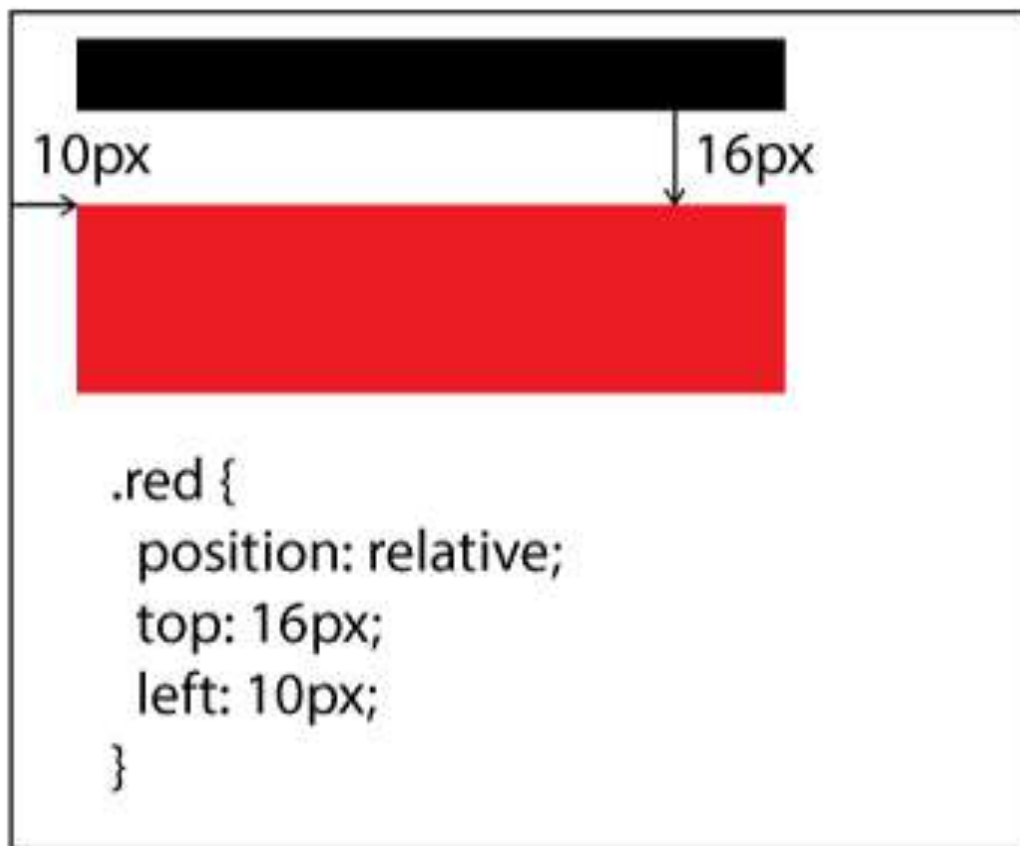
Bienvenidos a mi pagina

[Ver más opciones](#)

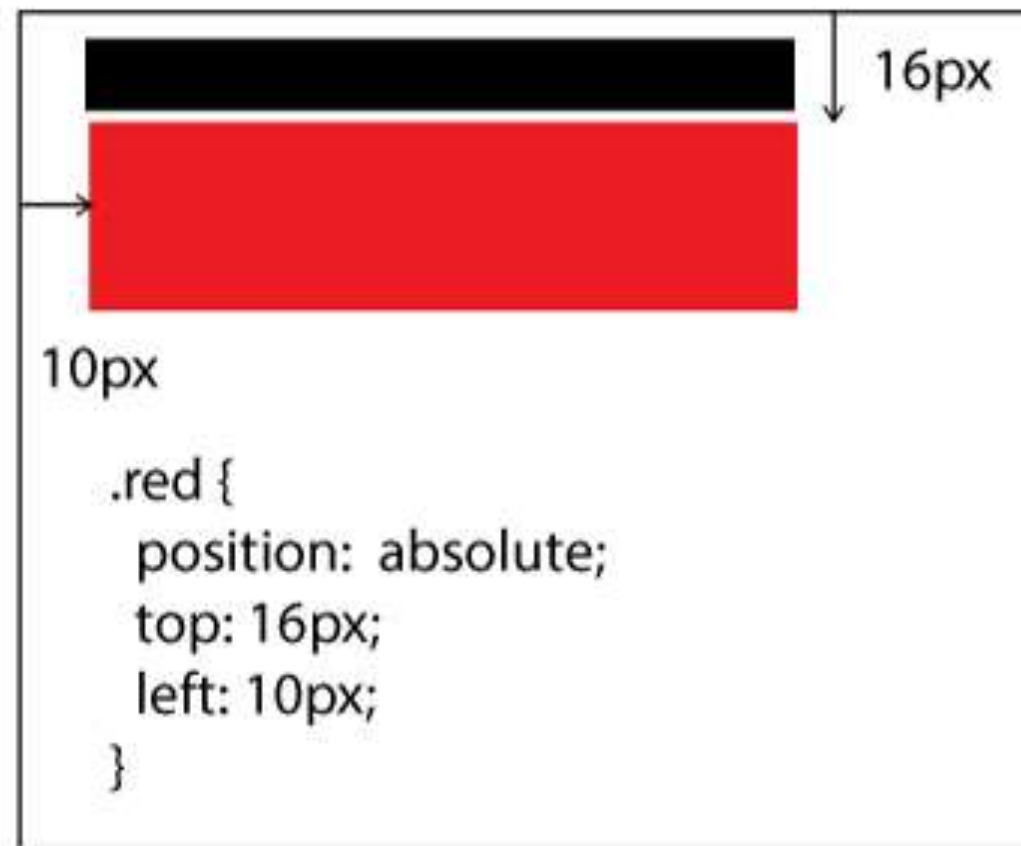




Relative position



Absolute position



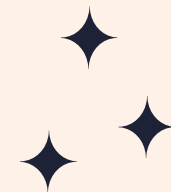


Indice Z

Todo elemento tiene una posición en el eje X (horizontal) y el eje Y (vertical)

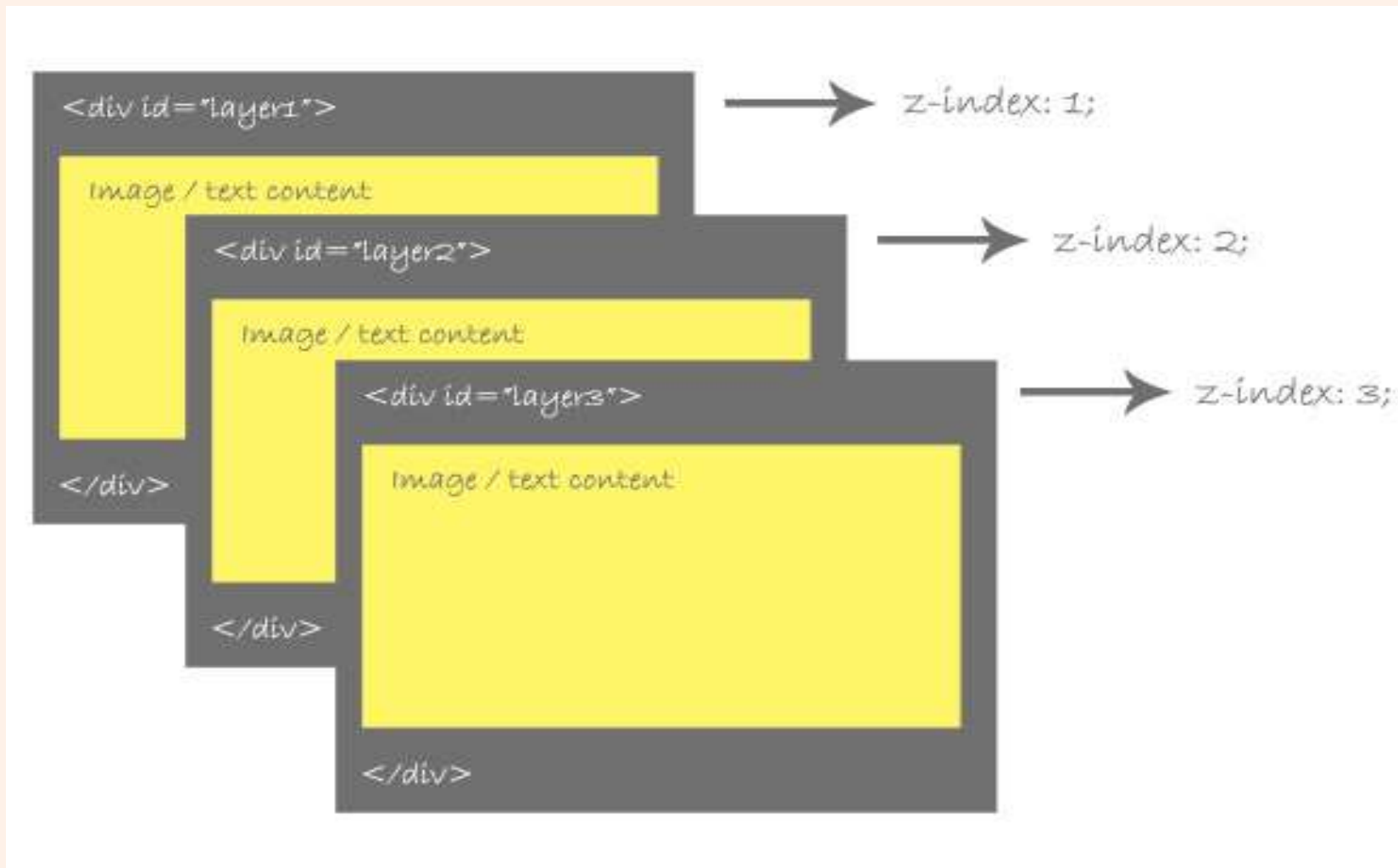
En CSS también está el eje Z que determina la posición de elementos encimados





El atributo es «**z-index**» y por defecto tiene valor 0

Mientras más alto es el valor, el elemento tiene mayor prioridad y queda más adelante





05

Contenedores Flex y Grid

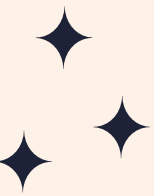




Web responsiva

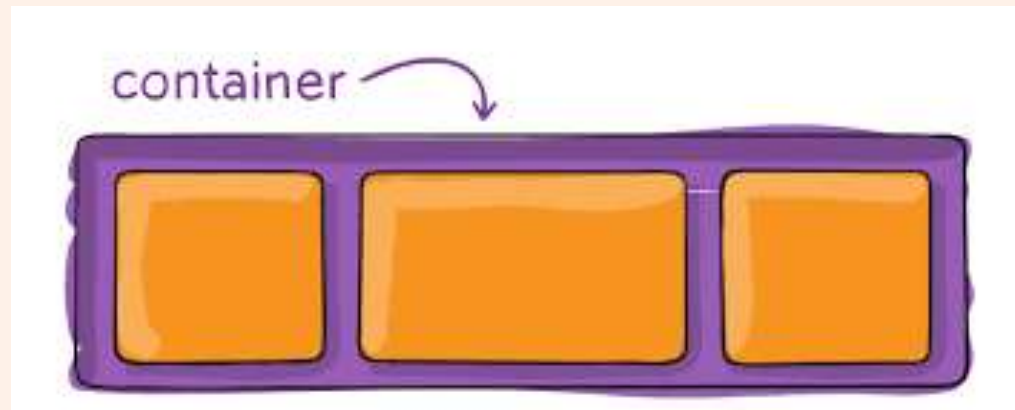
En CSS podemos dar los primeros pasos para hacer una página más flexible y **responsiva**, es decir adaptable a distintos dispositivos





Flexbox

El diseño de caja flexible, también llamado **flexbox** o simplemente **flex** es un contenedor que nos permite hacer varios cambios en el orden, tamaño y dirección de los elementos que contiene

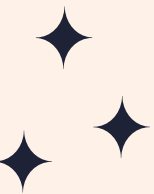




Contenedor e items

Primero necesitamos un div contenedor y otros divs que funcione como items. Les asignamos la clase «**container**» e «**item**» como corresponda

```
8 <div class="container">
9   <div class="item">
10     <h1>Ítem 1</h1>
11   </div>
12   <div class="item">
13     <h1>Ítem 2</h1>
14   </div>
15   <div class="item">
16     <h1>Ítem 3</h1>
17   </div>
18   <div class="item">
19     <h1>Ítem 4</h1>
20   </div>
21 </div>
```

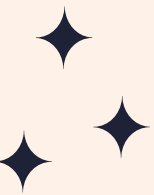


Luego creamos una regla para la clase «container» que diga **display: flex** (podemos agregar bordes y color para ver más clara la separación)



```
.container {  
  display: flex;  
  border:solid;  
  border-width: 4px;  
}  
  
.item {  
  border:solid;  
  border-width: 4px;  
  color: ■ brown;  
  margin: 10px;  
}
```

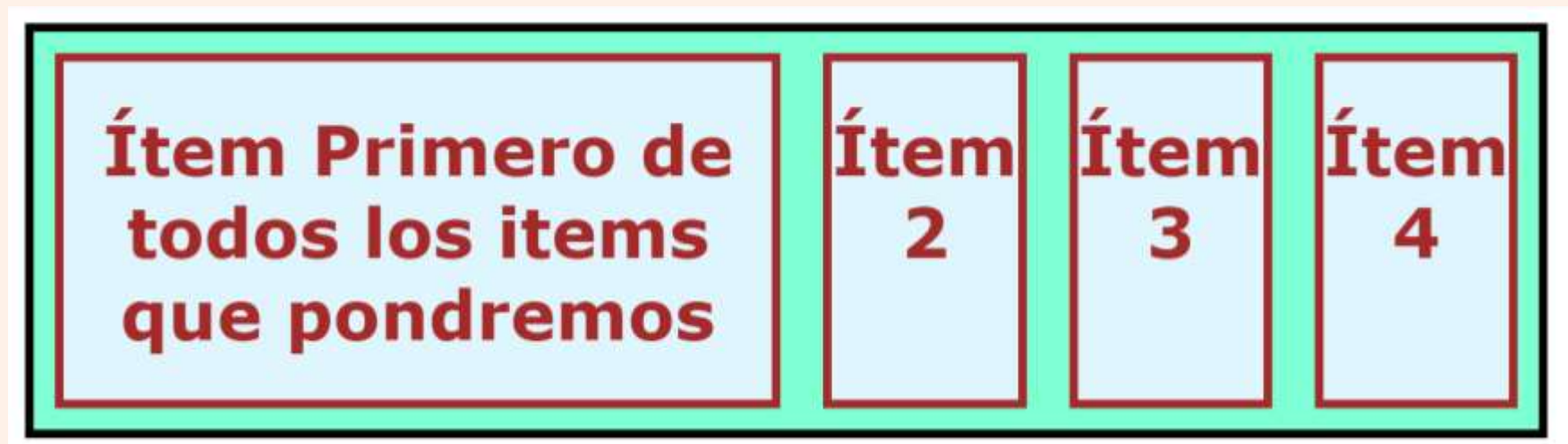




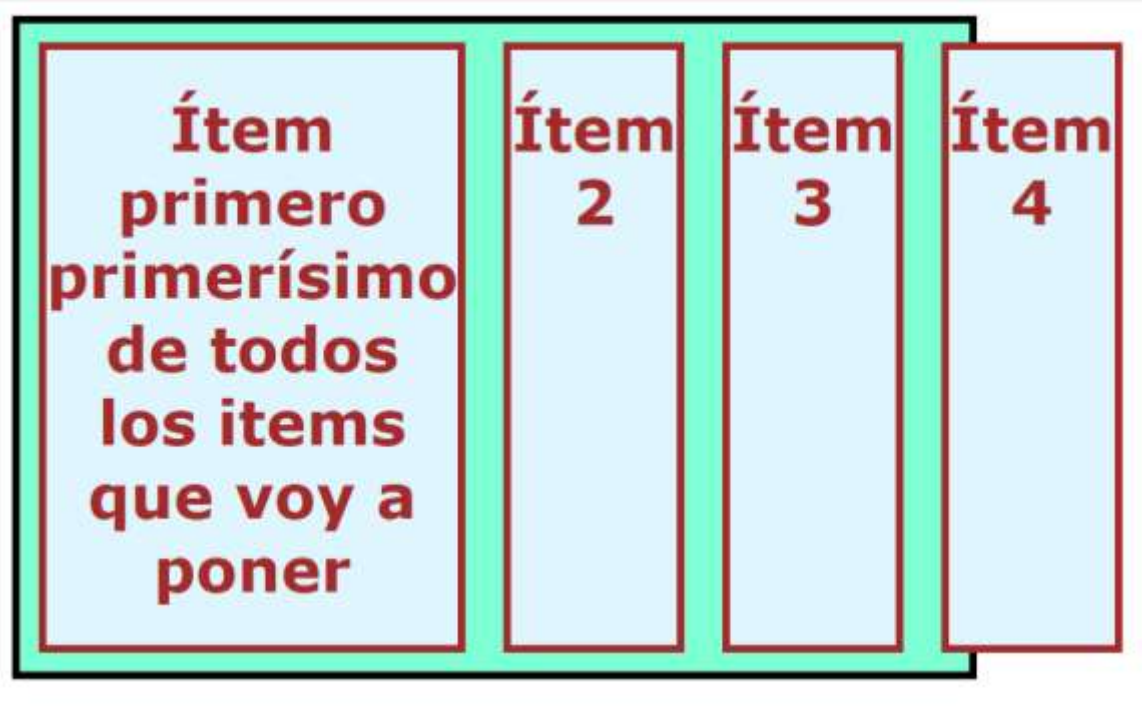
Items flex

Todo item, por defecto, va a ser tan grande como su contenido y se ajusta al alto o ancho del contenedor

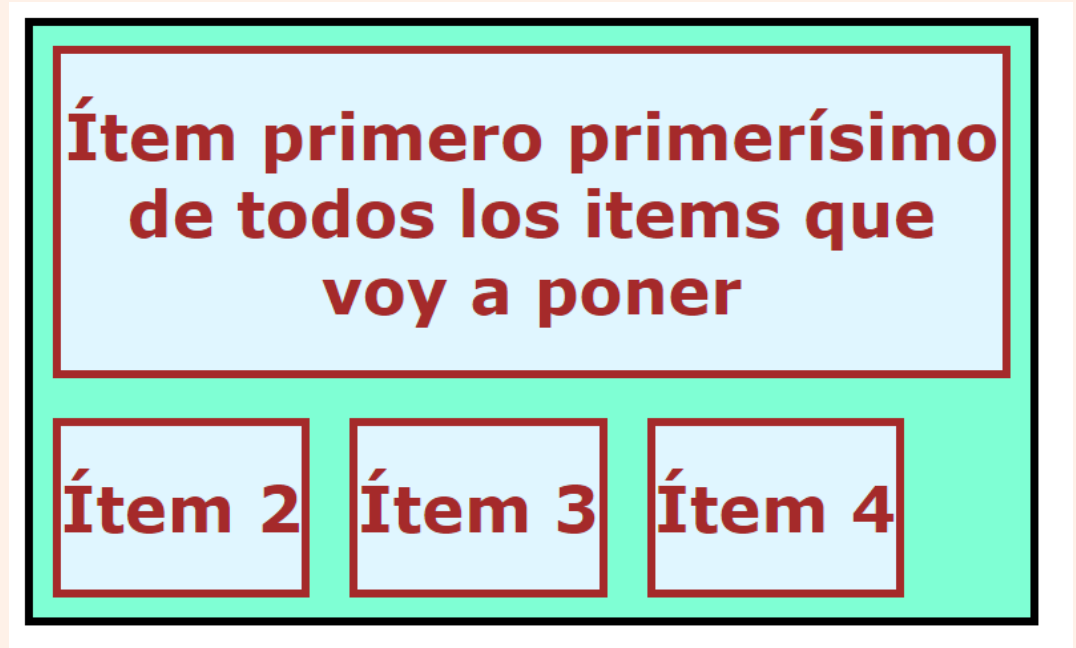
Tambien se puede ajustar el tamaño de item manualmente



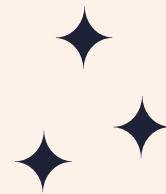
Si fijo el tamaño del contenedor, pero el item es demasiado grande, se sobrepasará



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



Para evitarlo, ajustamos la propiedad «**flex-wrap: wrap;**» para que envuelva todo los items

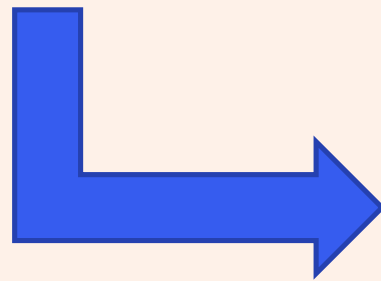


Filas y columnas

Por defecto, flex acomoda items horizontalmente, pero se puede cambiar desde la propiedad flex-direction

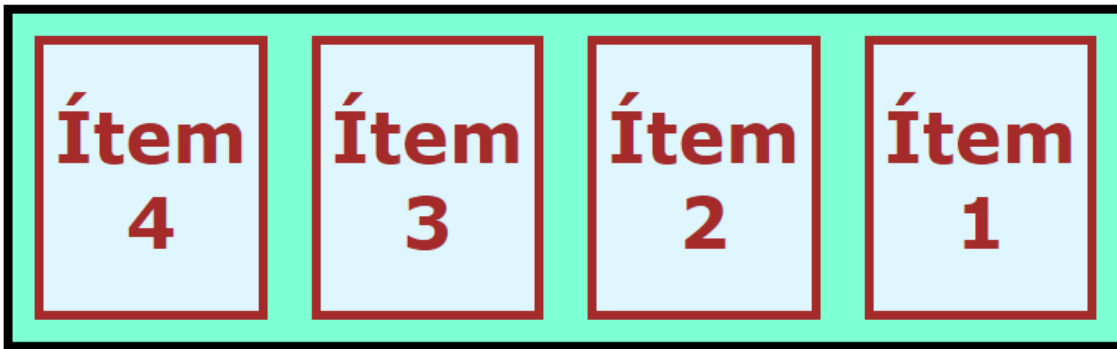
Usando el valor «**column**», hacemos que sea vertical

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

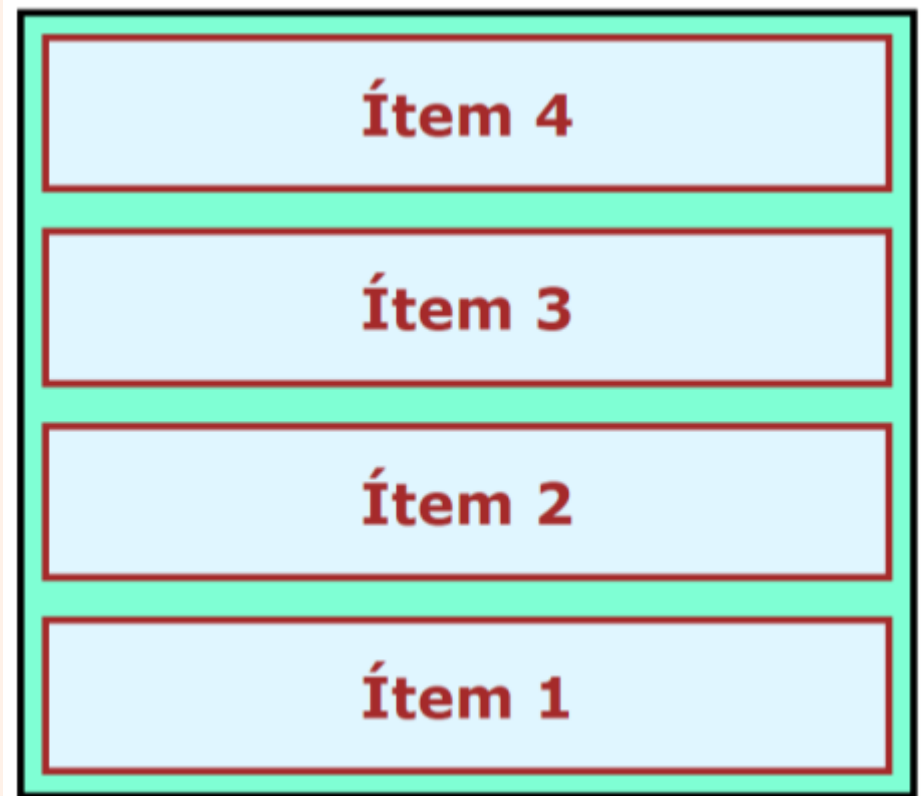


Se puede invertir el orden con “**row-reverse**” (horizontal) y “**column-reverse**” (vertical). El valor por defecto es “**row**”

```
.container {  
  display: flex;  
  flex-direction: row-reverse;
```



```
.container {  
  display: flex;  
  flex-direction: column-reverse;
```

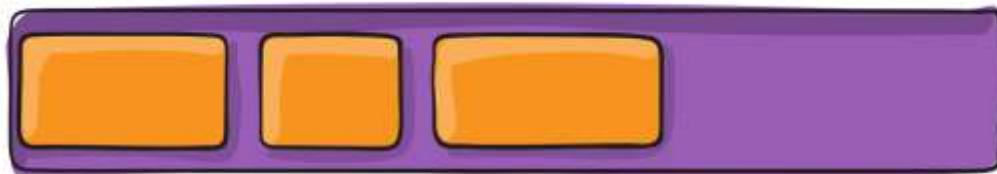




Justificar flex

Para justificar (controlar como se distribuyen los items) usamos «**justify-content**» y tenemos varias opciones:

flex-start



flex-end



center



space-between

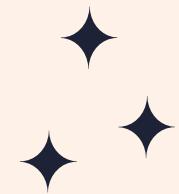


space-around



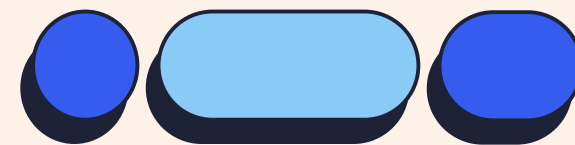
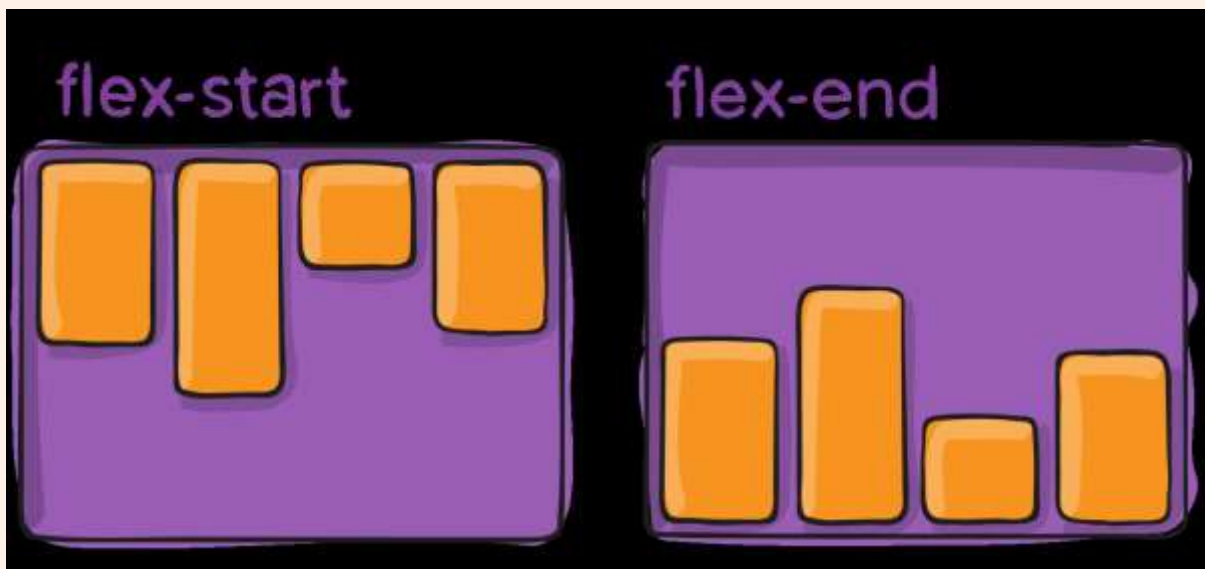
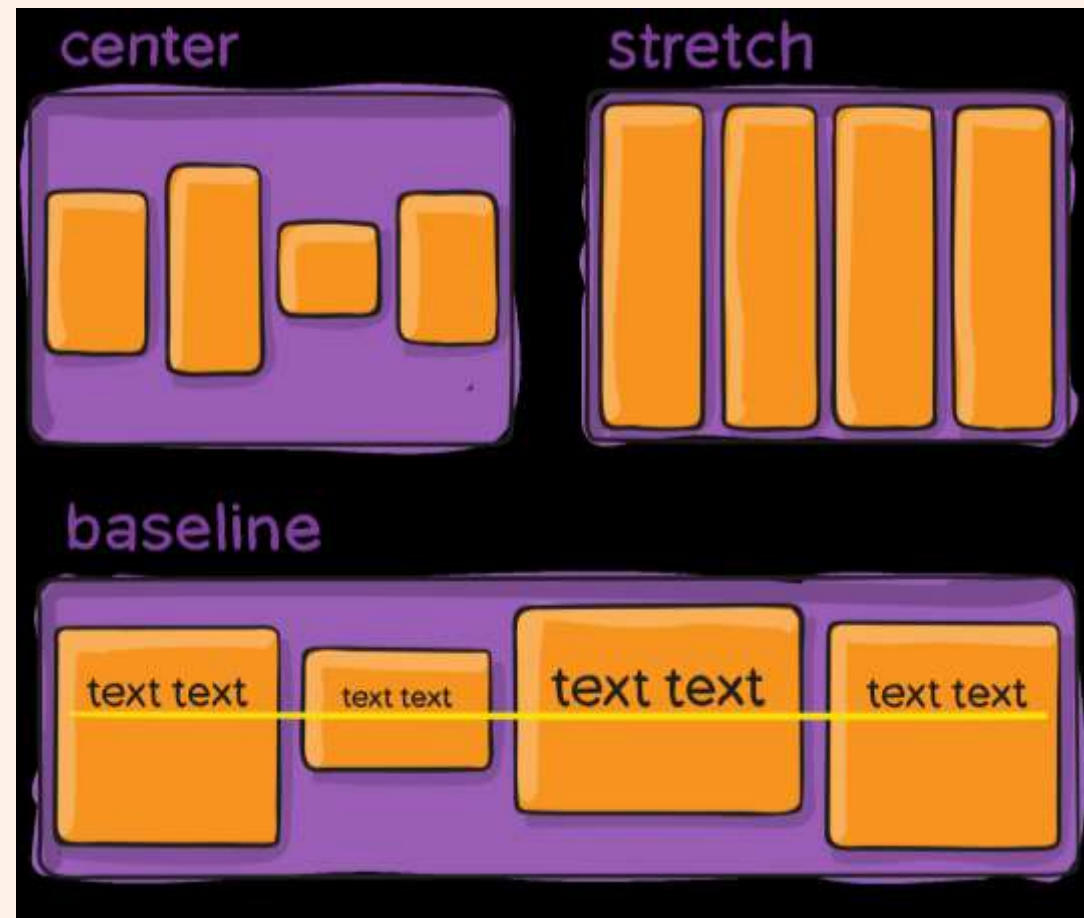
space-evenly





Alinear flex

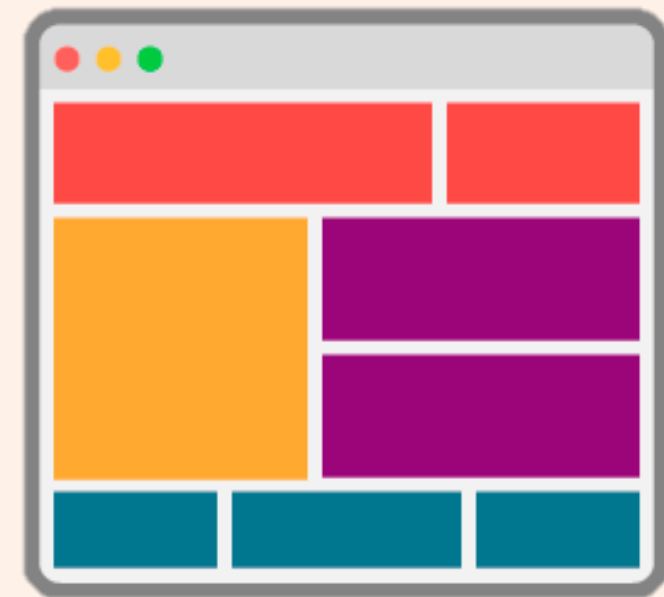
Parecido al anterior, solo que en el eje vertical, usando la propiedad «**align-items**»





Grid

Para acomodar elementos inline-block de forma dinámica, podemos incluirlos en una cuadrícula o «**grid**». Este permite mover divs en los ejes X e Y.



Igual que flex, se usa el concepto de **herencia**: existe una división «container» y dentro divisiones de clase «item», padre e hijo respectivamente

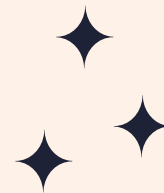




Inicia con atributo
«**display: grid;**» en
el elemento container

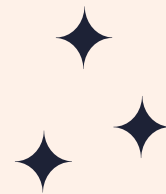
```
.container{  
  display:grid;  
  border:solid;  
}
```

```
<div class="container">  
  
  <div id="item">  
    <p>  
      <h1>Bienvenidos a mi pagina</h1>  
    </p>  
  </div>  
  
  <div class="item">  
    <h6><a href="">Ver más opciones</a></h6>  
  </div>  
  
  <div class="item">  
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing  
      elit. Donec fermentum sem vel congue ultrices.  
    </p>  
  </div>  
  
</div>
```

```
<div class="grid-container">  
  <div class="grid-item">1</div>  
  <div class="grid-item">2</div>  
  <div class="grid-item">3</div>  
  <div class="grid-item">4</div>  
  <div class="grid-item">5</div>  
  <div class="grid-item">6</div>  
  <div class="grid-item">7</div>  
  <div class="grid-item">8</div>  
  <div class="grid-item">9</div>  
</div>
```

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

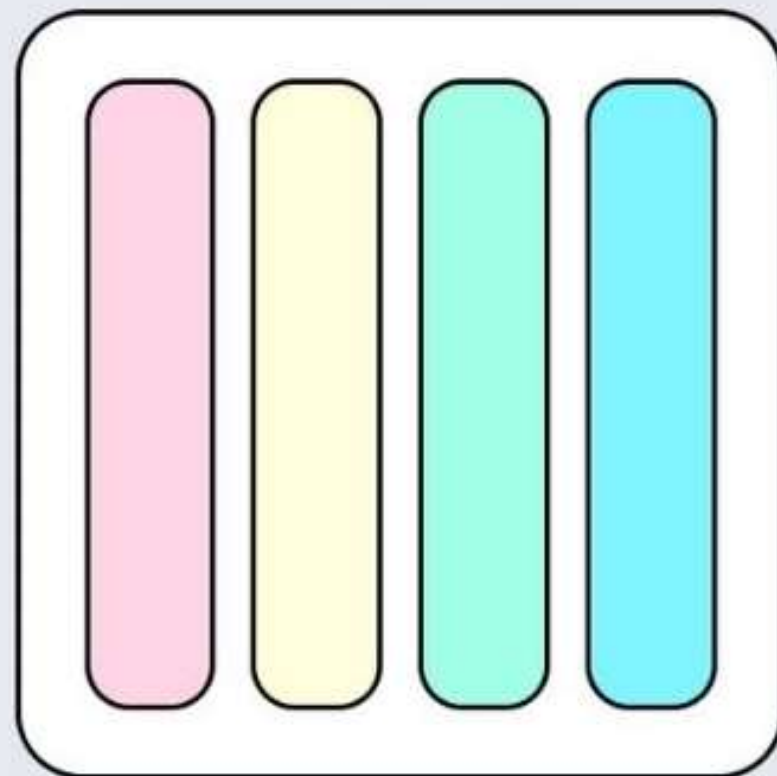


Filas y columnas

El grid se divide los elementos en filas y columnas

Atributo «**grid-template-columns**» y los valores son las medidas y cantidad de columnas que queremos

```
grid-template-columns:  
75px 75px 75px 75px;
```



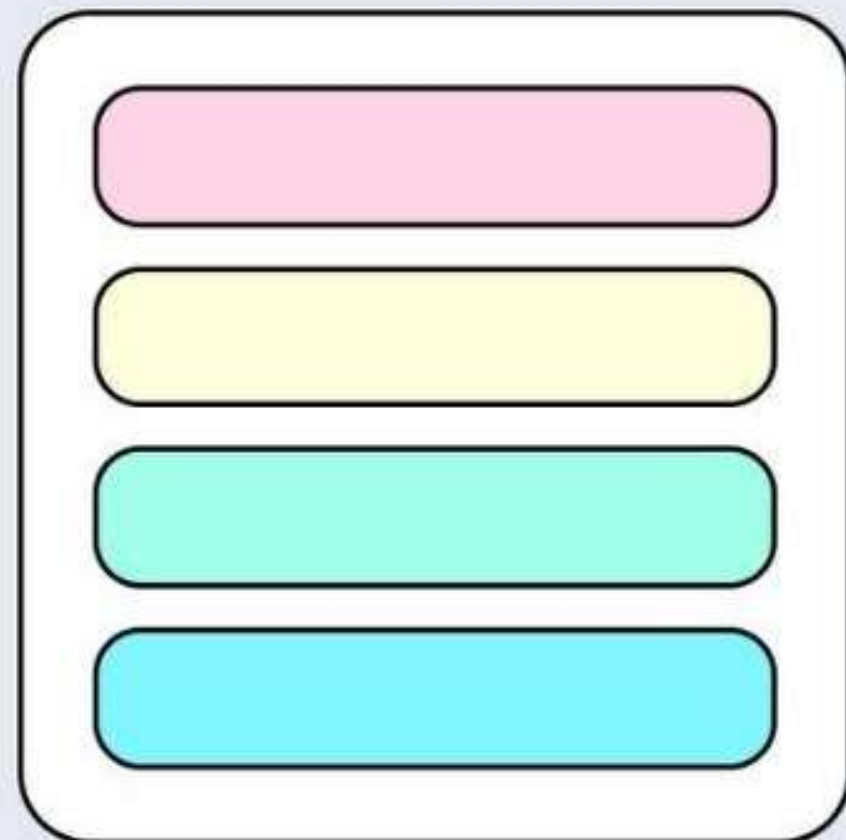


Filas

Igual con las columnas con el atributo «**grid-template-columns**»

Recuerden que se pueden usar medidas absolutas o relativas

```
grid-template-rows:  
75px 75px 75px 75px;
```





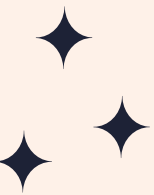
Un ejemplo de medidas relativas para esto son los porcentajes: si damos 70% a una columna , ocupará la mayoría de la pantalla

| | |
|--|----------------------------------|
| Bienvenidos a mi pagina | Ver más opciones |
| Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec fermentum sem vel congue ultrices. | |

```
.container{
  display:grid;
  border:solid;
  grid-template-columns: 70% 30%;
}

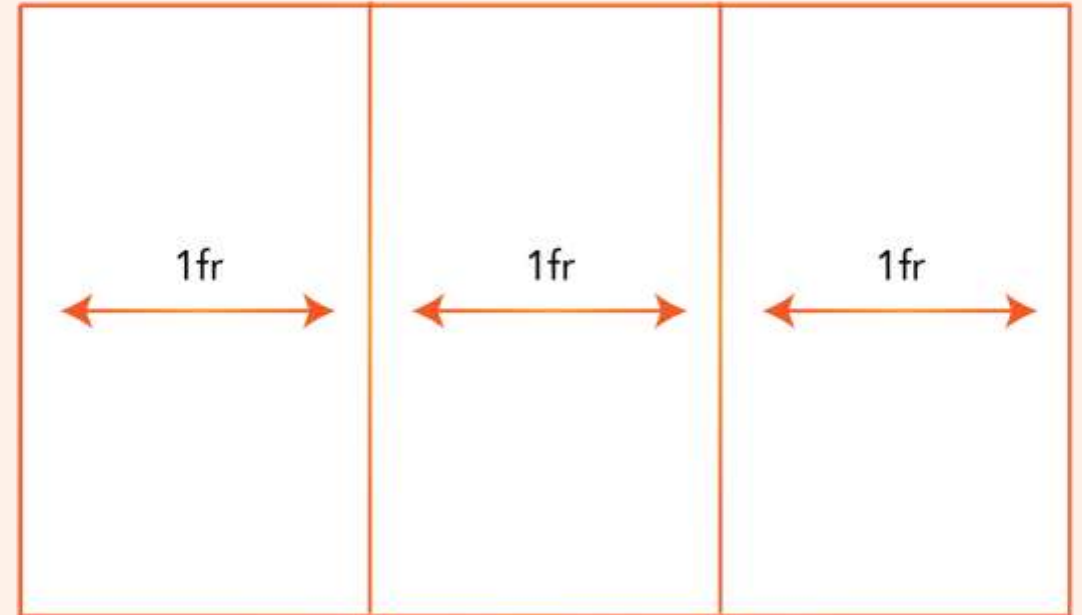
.item{
  border:solid;
}
```





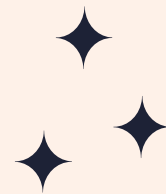
Unidad fr

Los fr o unidades fraccionales son relativas al espacio disponible



«**grid-template-columns: 1fr 1fr 1fr;**» crea tres columnas idénticas que ocupan toda la pantalla

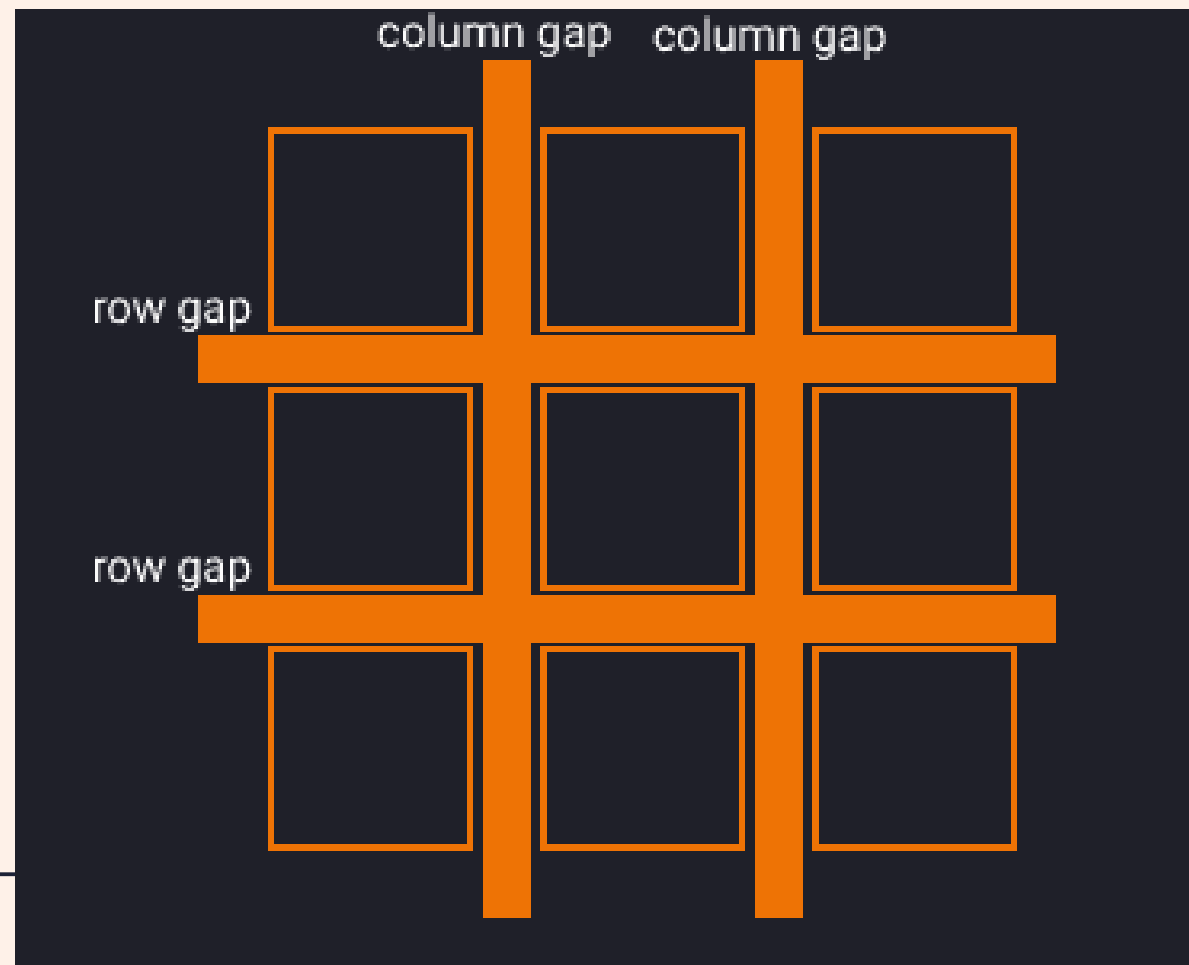


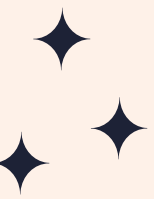
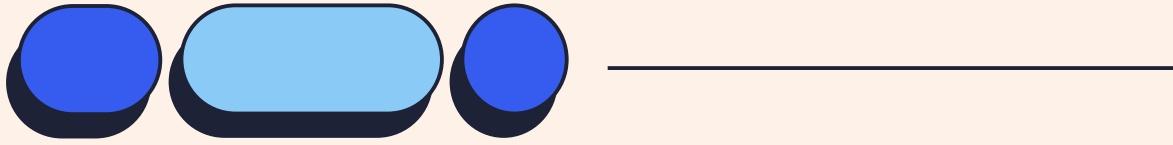


Concepto de gap

Por defecto, los ítems de un grid no tienen espacio entre si


Podemos agregar una distancia o gap entre los ítems






Se diferencia el gap de columnas y de filas

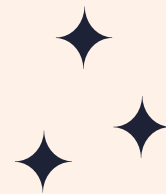
```
.container{
  display:grid;
  grid-template-columns: 200px 200px 200px;
  column-gap: 5px
}

.item{
  background-color:  aqua;
  border:solid;
  border-width: 3px;
}
```

```
.container{
  display:grid;
  grid-template-rows: 400px 400px;
  row-gap: 5px
}

.item{
  background-color:  aqua;
  border:solid;
  border-width: 3px;
}
```

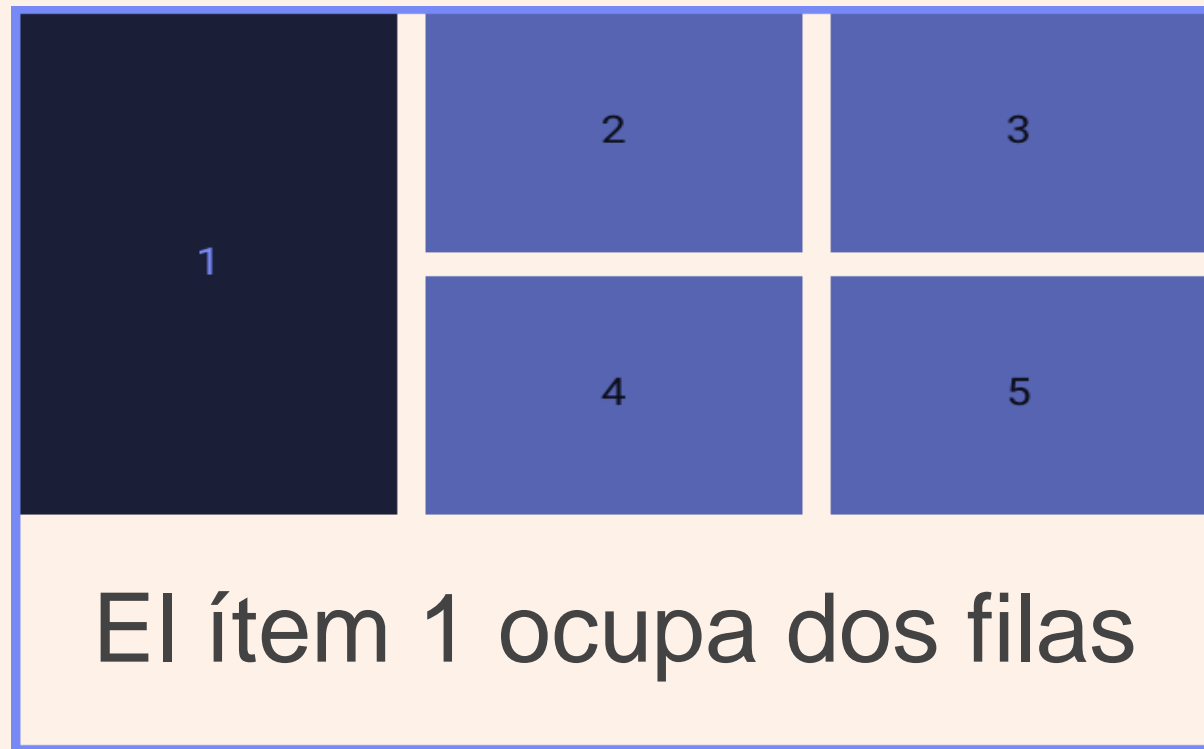
Atributo «**column-gap**» para espacio entre las columnas y «**row-gap**» para las filas

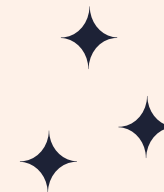


Espacio en grid

Los ítems pueden ocupar mas de una columna o mas de una fila

Se debe modificar el atributo «**grid-row**» para más filas o «**grid-column**» para más columnas en el ítem





Para agrandar un ítem se debe usar valores de tipo **span**

```
<div class="container">
  <div class="item item-1">
    1
  </div>
  <div class="container">
    <div class="item">
      2
    </div>
  </div>
</div>
```

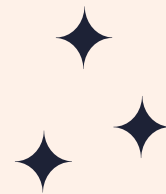
```
.item-1{
  grid-column: span 2;
}
```

Ocupa 2
columnas

```
.item-1{
  grid-row: span 3;
}
```

Ocupa 3
filas

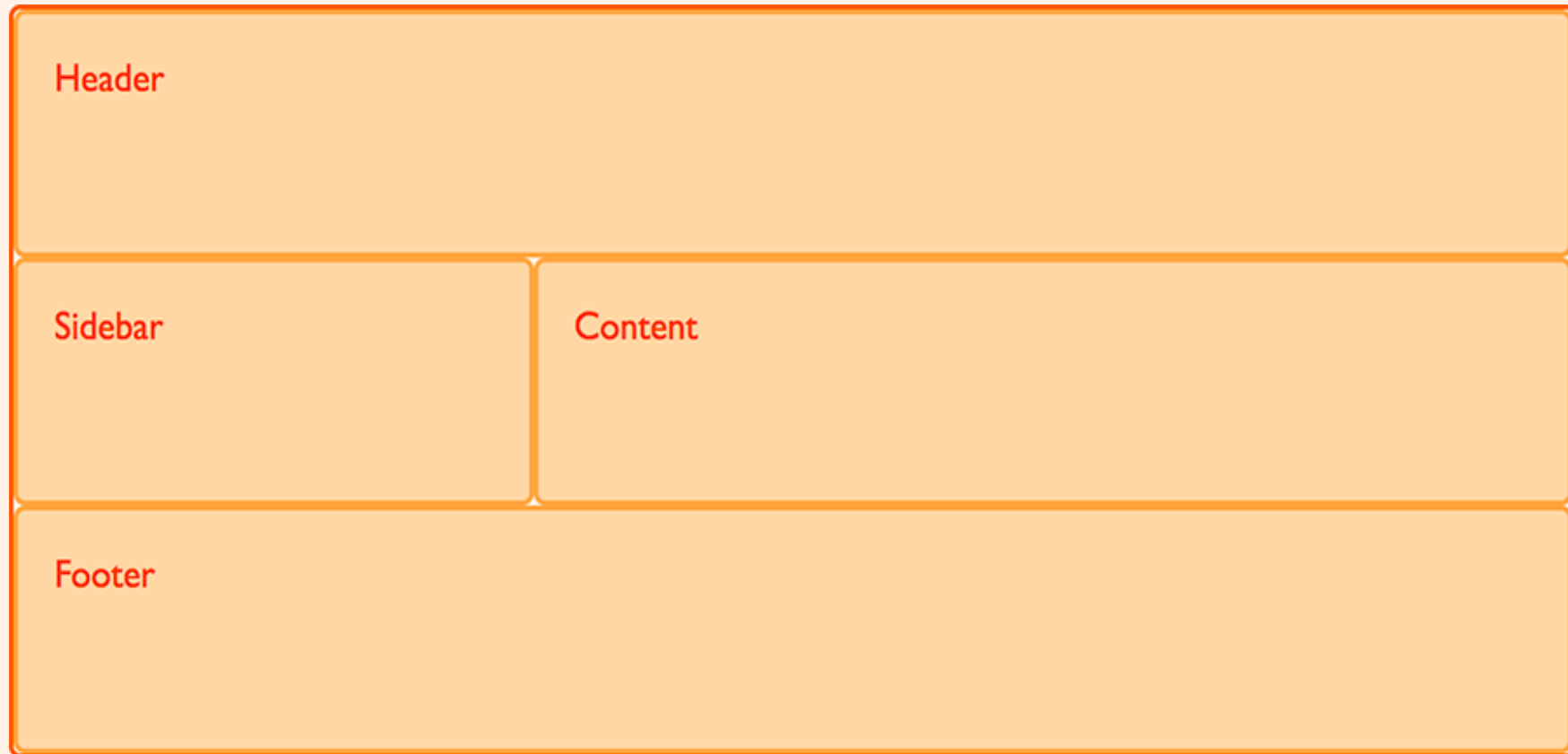


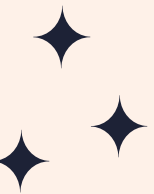
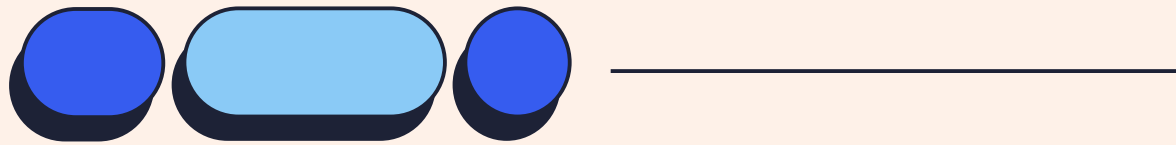


Area grid

Si bien hablamos de ítems, cada celda puede tener nombre propio

4 áreas (header, sidebar, content y footer) que son ítems de un grid





Primero hay que ponerle una clase además de la clase “item” en el HTML (nota: todo elemento HTML puede tener más de una clase)

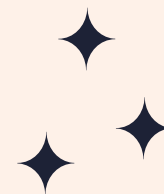
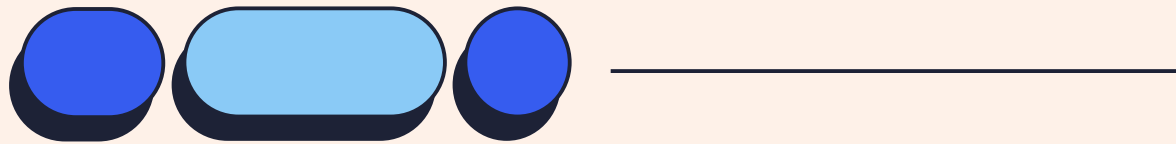
```
<div class="container">
  <div class="item item1">
    <h1>Área 1</h1>
  </div>
  <div class="item item2">
    <h1>Área 2</h1>
  </div>
</div>
```

Atributo «**grid-area:**
nombreArea;» para cada ítem

```
.item {
  background-color: ■ #e0f6ff;
  border:solid; border-width: 4px;
  color: ■ brown
}

.item1{
  grid-area: item1;
}

.item2{
  grid-area: item2;
}
```

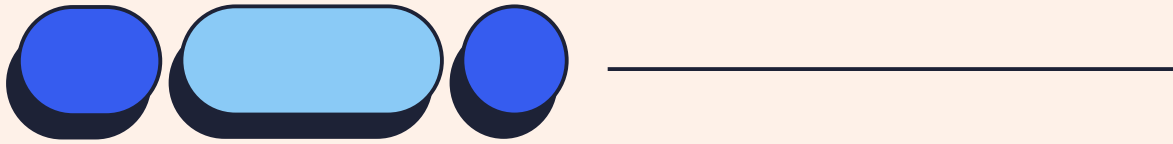


Para manejar la distribución de las áreas, usamos el atributo `grid-template-areas` en el container

```
.container {  
  display: grid;  
  font-family: Verdana; text-align: center;  
  border:solid; border-width: 4px;  
  grid-template-areas: "item1 item2";  
}
```



El nombre de dos áreas seguido las acomoda en más columnas

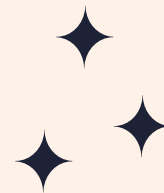
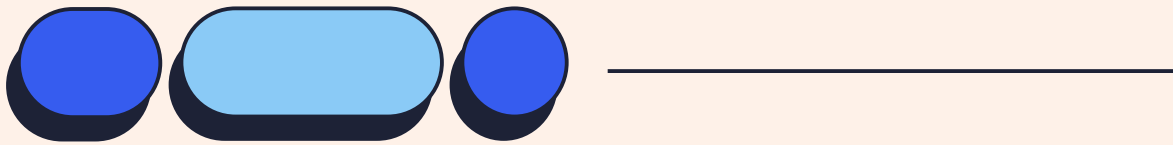


Para hacer filas volvemos a abrir comillas y colocar las areas que serían la próxima fila

| | |
|--------|--------|
| Ítem 1 | Ítem 2 |
| Ítem 3 | Ítem 4 |

```
.container {  
  display: grid;  
  font-family: Verdana; text-align: center;  
  border:solid; border-width: 4px;  
  grid-template-areas: "item1 item2" "item3 item4";  
}
```

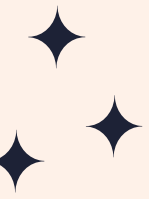
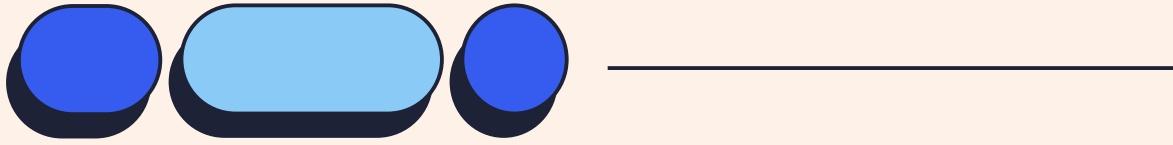
Fila 1 Fila 2



Para que un área ocupe varios espacios, solo hay que repetir su nombre

```
.container {  
  display: grid;  
  font-family: Verdana; text-align: center;  
  border:solid; border-width: 4px;  
  grid-template-areas: "item1 item1 item1"  
                       | "item2 item3 item4";  
}
```





Podemos poner cualquier nombre a las áreas

```
grid-template-areas: "header header header"  
                    "sidebar content content"  
                    "footer footer footer";
```



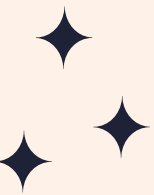


Siempre debemos mantener la misma cantidad de columnas nombradas en cada fila

```
grid-template-columns: 200px 1fr;
```

```
grid-template-areas: "header header"  
                    "sidebar main"  
                    "sidebar footer";
```





Areas vacías

Podemos crear áreas vacías con un punto .

| | | |
|--------|--------|--------|
| | Ítem 1 | |
| Ítem 2 | Ítem 3 | Ítem 4 |

```
.container {  
  display: grid;  
  font-family: Verdana; text-align: center;  
  border:solid; border-width: 4px;  
  grid-template-areas: ". item1 . "  
                        "item2 item3 item4";  
}
```



Actividad

-Punto 1: Crear un formulario

Que tenga 1 placeholder y un campo obligatorio

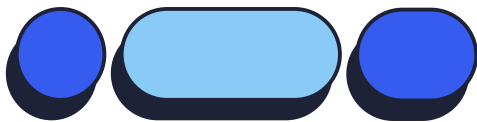
-Punto 2: Radio buttons

Agregar un grupo de radio buttons con un h2.

El título debe ser : "mi helado preferido es", y 3 opciones de gustos.

Importante: al elegir uno debe elegirse ese solamente.

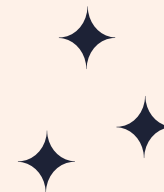
Pista: usar atributo 'name'



¿Qué son las tablas?

Son elementos visuales
compuestos por columnas
verticales y filas horizontales
divididas en celdas

| Nombre | | Edad | Curso |
|---------|---------------|------|-----------|
| Eduardo | Soria Gómez | 12 | Primero D |
| Carlos | Pérez Díaz | 14 | Tercero A |
| Martín | Pereira Silva | 11 | Primero B |



Actividad

Prácticas.



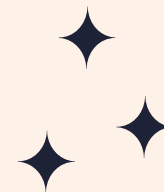
A lo largo de las clases se les dejarán archivos con ejercicios prácticos opcionales

Estos mismos se trabajarán en clase y servirán para ensayar todo lo visto en clase

Proyecto integrador

Desarrollar e implementar todo lo visto en un proyecto grupal (necesario para completar el curso)





Resumiendo

Unidades

Manejamos unidades absolutas (px, cm, etc.) y unidades relativas (% , em, fr)

Prioridad

La regla más particular es la prioritaria, salvo que usemos “!important”

Flujos

Existen los flujos block, inline e inline-block y se manipulan con display y float

Posición

Hay posiciones absolutas y relativas, así como posición en eje Z respecto a otros elementos

Grid y flex

Contenedores dinámicos que permiten cambiar orden y posición fácilmente





Eso es todo

Se escuchan dudas, quejas y
propuestas

F
i
n
d
e
l
a
c
l
a
s
e

