

Esta clase va a ser

- grabada

Clase 04. Desarrollo Web

CSS + BOX MODELING

Objetivos de la clase

- 1 Comprender y modificar elementos en bloque y en línea.
- 2 Conocer las cajas y sus propiedades.
- 3 Entender la metodología BEM

Glosario

Etiqueta IFRAME: es un elemento HTML que permite insertar o incrustar un documento HTML dentro de un documento HTML principal.

CSS (Cascading Style Sheets): es un lenguaje web para aplicar formato visual (color, tamaño, separación y ubicación) al HTML. Con él puedes cambiar por completo el aspecto de cualquier etiqueta HTML.

Padre e hijos: es un concepto que se aplica cuando tienes una etiqueta “dentro” de otra. Esto te habilita a agregar atributos específicos a “hijos”, sin alterar los del “padre”.

Class: generalmente se utiliza para darle estilos a cierta parte del código.

Atributo ID: suele usarse para nombrar porciones de código y sectores, como por ejemplo cuando quieres nombrar distintas secciones.

Glosario

Joroba de camello: permite que se puedan leer de forma más simple palabras compuestas.

Reset CSS: contienen en su código fuente definiciones para propiedades problemáticas, que los diseñadores necesitan unificar desde un principio.

Unidades de medidas

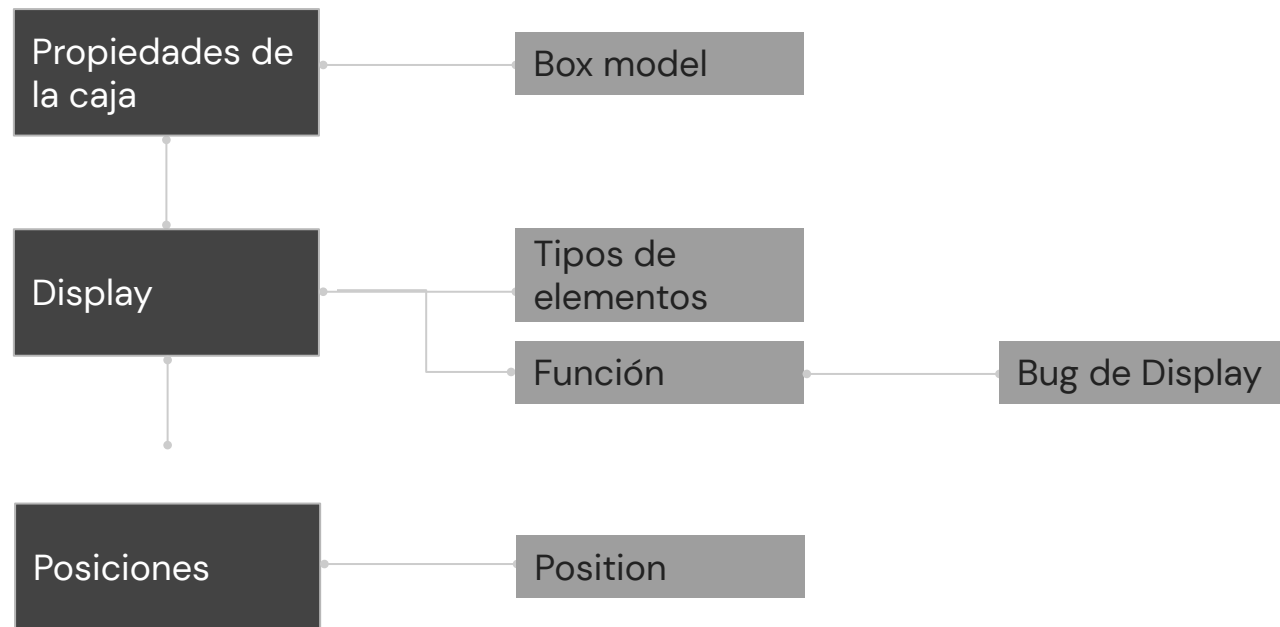
Absolutas

- **Px (pixels):** es la unidad que usan las pantallas.

Relativas

- **Rem:** relativa a la configuración de tamaño de la raíz (etiqueta html).
- **Porcentaje:** tomando en cuenta que 16px es 100%.
- **Viewport:** se utilizan para layouts responsivos (más adelante).

MAPA DE CONCEPTOS



Temario

03

Incluyendo CSS a nuestro proyecto

- ✓ Bases de CSS
- ✓ Insertar CSS
- ✓ Primeras propiedades

04

CSS + Box Modeling

- ✓ Bem
- ✓ Box Modeling
- ✓ Display
- ✓ Posiciones

05

Flexbox

- ✓ Flexbox
- ✓ Propiedad de padres e hijos

Recapitulación de la clase anterior



PARA RECORDAR

Conceptos previos

Sabemos que existen **tres maneras** de aplicar CSS a un documento HTML:

- Hacerlo sobre la etiqueta con el atributo `style=""`
- En el head, insertar la etiqueta `<style>`
- Buscar un archivo externo con un `<link />`
(Es de las etiquetas que se cierran solas. Requiere `el="stylesheet"` para funcionar. Además un `href=""` con la ruta al archivo.)

Selección de HTML mediante CSS

Tres posibilidades

Por etiqueta



```
h1 {  
    propiedad: valor;  
}
```

Por clase

(anteponiendo el ".")



```
.clase {  
    propiedad: valor;  
}
```

Por ID

(anteponiendo el "#")



```
#id {  
    propiedad: valor;  
}
```

Selección de HTML mediante CSS

Ejemplos



p a

```
<p>  
  Párrafo con <a href="">enlace</a>  
</p>
```



a.foo

```
<a href="" class="foo">Enlace</a>
```



a#foo

```
<a href="" id="foo">Enlace</a>
```



.foo .bar

```
<etiqueta class="foo">  
  <etiqueta class="bar"></etiqueta>  
</etiqueta>
```

Selección de HTML mediante CSS

Ejemplos



.foo .foo #foo



p.foo a.bar

```
<etiqueta class="foo">  
  <etiqueta class="foo">  
    <etiqueta id="foo"></etiqueta>  
  </etiqueta>
```

```
<p class="foo">  
  <a href="" class="bar">Enlace</a>  
</p>
```

Unidades de medidas

Unidades de medida

Hay una amplia variedad de absolutas y relativas, pero nos centraremos en:

Absolutas

- ✓ Px (pixels): es la unidad que usan las pantallas.

Relativas

- ✓ Rem: relativa a la configuración de tamaño de la raíz (etiqueta html).
- ✓ Porcentaje: tomando en cuenta que 16px es 100%.
- ✓ Viewport: se utilizan para layouts responsivos (más adelante).

Unidades de medida

Ahora veamos qué medida es más conveniente para los textos.

```
html { /* etiqueta raíz */  
  font-size: 62.5%;  
}  
p {  
  font-size: 2rem; /* 20px */  
}
```

Texto simulando
20px

62.5%, hace que en vez de que 16px sea el valor a tomar en cuenta para calcular las unidades relativas, se use 10px.

Los nombres de las clases e IDs

No es posible crear nombres separados por espacios.

La “*joroba de camello*”, permite que se puedan leer de forma más simple palabras compuestas.

claseDeMaquetacion

conBorde

productosMasVendidos

BEM

BEM

Todos queremos hacer que **nuestro código sea más fácil de leer**. Esto nos ayuda a trabajar más rápidamente y de manera eficiente, y cuando otros trabajen con nosotros podremos mantener claridad y coherencia.

Hoy vamos a descubrir la **metodología BEM**, que nos ayudará a entender estructuras de CSS, y a mejorar las nuestras.

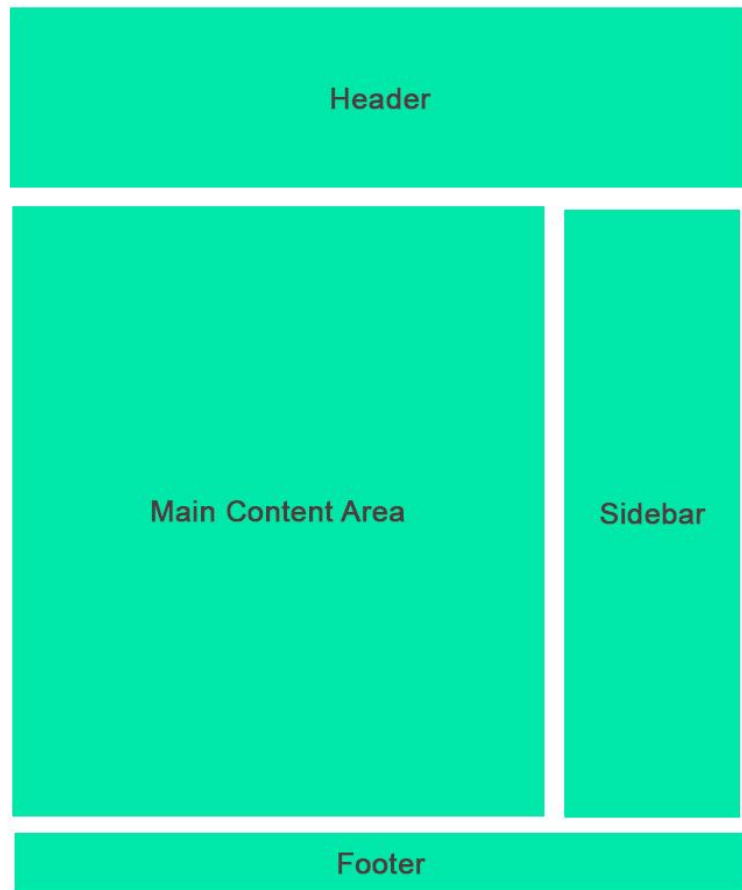
BEM

👉 **BEM** significa Modificador de Bloques de Elementos (Block Element Modifier) por sus siglas en inglés. Sugiere una manera estructurada de nombrar tus clases, basada en las propiedades del elemento en cuestión.

👉 **BEM** tiene como horizonte modularizar lo máximo posible cada uno de los bloques de código dispuesto. Se centra en tres parámetros o variables posibles: **bloques** (div, section, article, ul, ol, etc.), **elementos** (a, button, li, span, etc.) y **modificadores**. Estos últimos se definen de acuerdo a la posterior utilización que haga el desarrollador a cargo.

BLOQUE 1

El bloque es un **contenedor o contexto donde el elemento se encuentra presente**. Piensa como si fueran partes estructurales de código más grandes. Puede que tengas un encabezado, pie de página, una barra lateral y un área de contenido principal; cada uno de estos sería considerado como un bloque.



BLOQUE 1

El bloque de elementos forma la raíz de la clase y siempre irá primero. Solo debes saber que una vez que has definido tu bloque, estarás listo para comenzar a nombrar tus elementos.

```
.block__element {  
    background-color:  
#FFFFFF;  
}
```

BLOQUE 1

La doble barra baja te permite visualizar, navegar rápidamente y manipular tu código. Aquí hay algunos ejemplos de cómo funciona la metodología de elementos:

```
.header__logo {}  
.header__tagline {}  
.header__searchbar {}  
.header__navigation  
{}
```

BLOQUE 1

HTML

```
11 <section>
12 |   <article class="noticia">
13 |     <!--Bloque contenedor-->
14 |   </article>
15 </section>
```

CSS

```
1 .noticia{
2 |   background: lightgray;
3 }
```

BLOQUE 1

El punto es mantener los nombres **simples, claros, y precisos.**

No lo pienses demasiado. 🚀

Actualizar el nombre de las clases no debería ser un problema cuando encuentras una mejor semántica que funcione (sólo debes tratar de ser consistente, y apegarte a ella).

ELEMENTOS

2

El elemento es una de las piezas que compondrán la estructura de un bloque. El bloque es el todo, y los elementos son las piezas de este bloque.

De acuerdo a la metodología BEM, cada elemento se escribe después del bloque padre, usando dos guiones bajos.

ELEMENTOS 2



HTML

```
11 <section>
12   <article class="noticia">
13     <h1 class="noticia__titulo">Título de la noticia</h1>
14   </article>
15 </section>
```

CSS

```
5 .noticia__titulo{
6   font-family: 'Times New Roman', serif;
7 }
```

MODIFICADORES

3

Cuando nombras una clase, la intención es ayudar a que ese elemento pueda ser repetido, para que no tengas que escribir nuevas clases en otras áreas del sitio si los elementos de estilo son los mismos.

Cuando necesitas modificar el estilo de un elemento específico, puedes usar un modificador. Para lograr esto, añade un doble guión -- luego del elemento (o bloque). Aquí tenemos un corto ejemplo:

```
.block--modifier {}  
.block__element--modifier {}
```

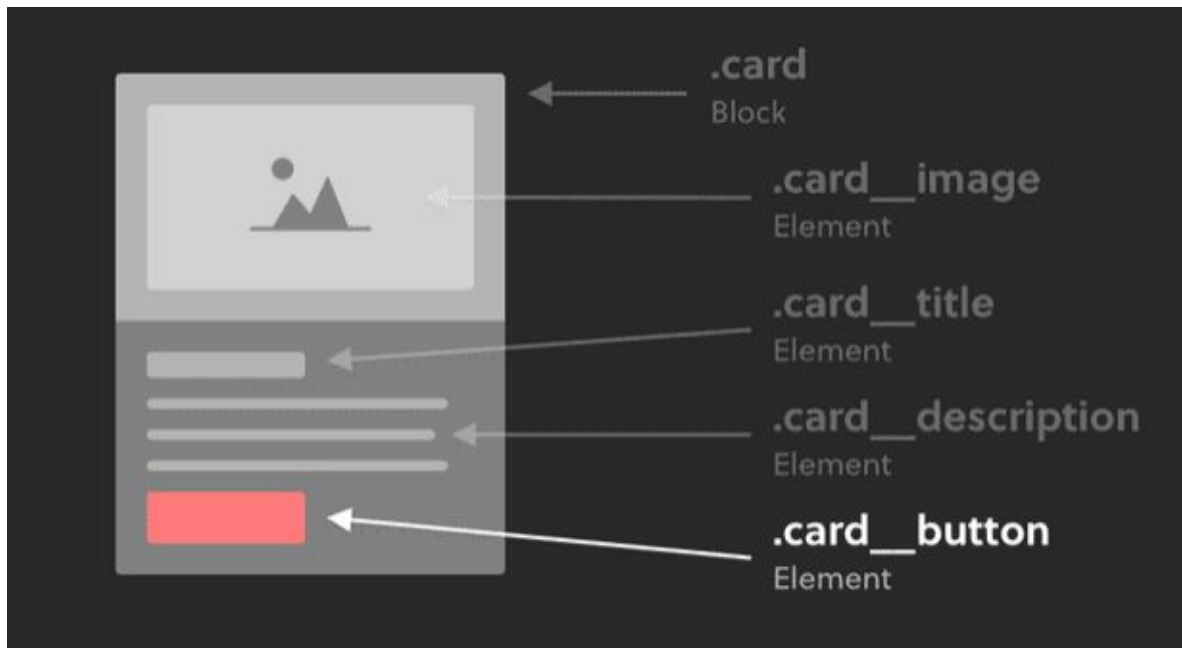
MODIFICADORES 3

HTML

```
11 <section>
12   <article class="noticia--destacada">
13     <h1 class="noticia__titulo--uppercase">Título de la noticia</h1>
14   </article>
15 </section>
```

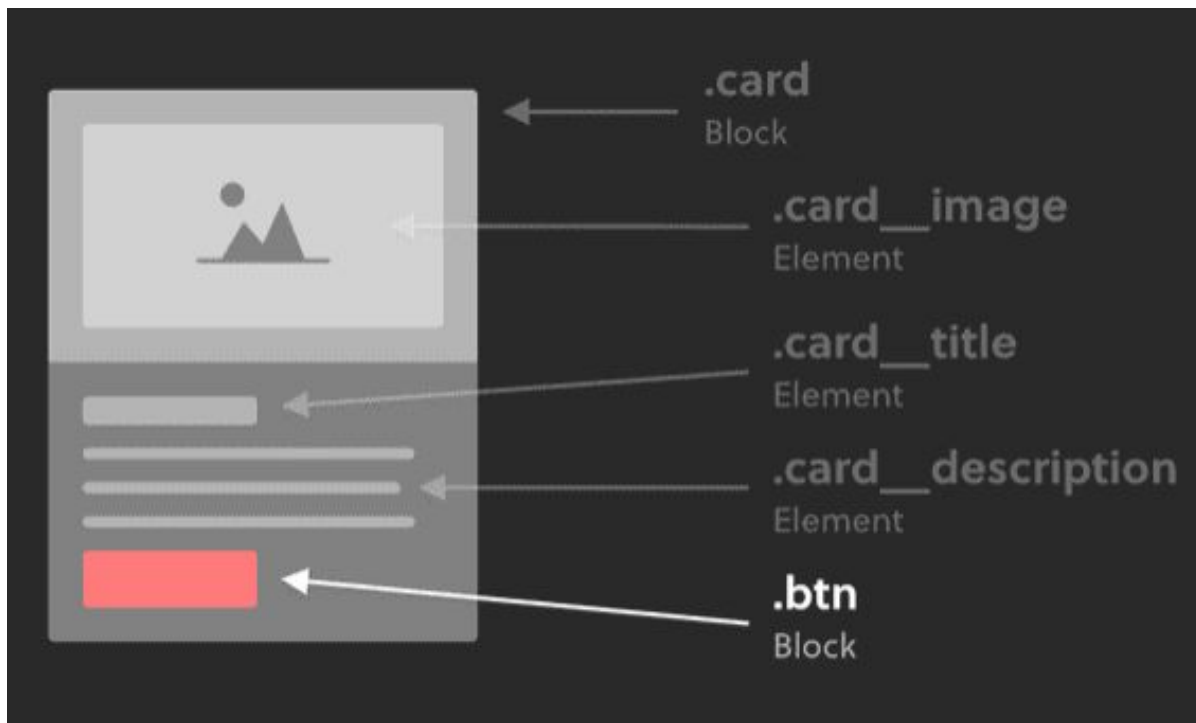
CSS

```
9  .noticia--destacada{
10    background: #d3d3d3;
11  }
12  .noticia__titulo--uppercase{
13    text-transform: uppercase;
14  }
```



.card__button esta bien nombrada pero un botón es un componente que podríamos reutilizar en otro contexto (carousel, con un texto, etc).

Ejemplo donde trabajamos con BEM pero que podría ser refinado para poder reutilizar código más adelante.



Ejemplo donde gracias al uso de BEM vamos a poder hacer reuso de clase.

El **código** de la slide anterior sería:

```
<div class="card">
  <img class="card__image">
  <h2 class="card__title">I am a card</h2>
  <p class="card__description">I am the card paragraph</p>
  <!-- The button is now it's own block -->
  <a class="btn">Learn more</a>
</div>
```

BEM – Ventajas



- Añade especificidad.
- Aumenta la independencia.
- Mejora la herencia múltiple.
- Permite la reutilización.
- Entrega simplicidad.

BEM – Desventajas



- Las convenciones pueden ser muy largas.
- A algunas personas les puede tomar tiempo aprender la metodología.
- El sistema de organización puede ser difícil de implementar en proyectos pequeños.

BEM: ¿Para qué lo usarías?

- Para simplificar nuestro CSS y conseguir un estilo consistente, por lo que nuestro código será mucho más legible y fácil de mantener.
- Si estamos usando Bootstrap y queremos modificar ciertas clases.
- Cuando trabajamos en equipo y cada miembro tiene una manera distinta de escribir CSS.



Break

¡10 minutos y volvemos!



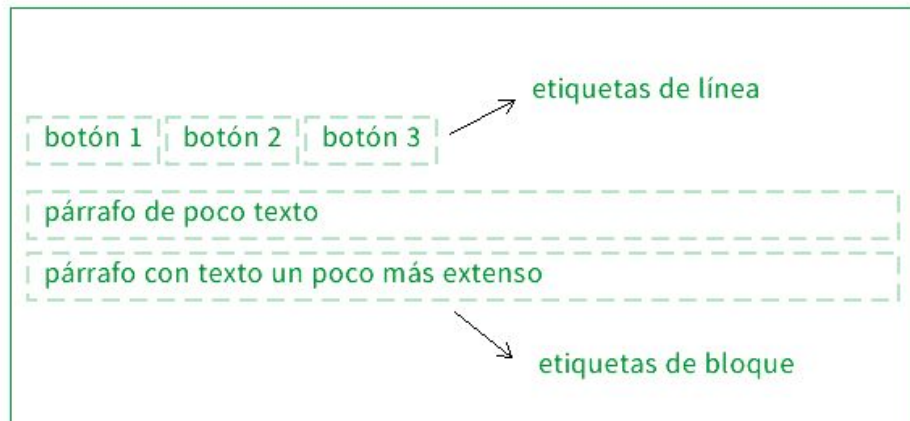
Ejemplo en vivo

¡Vamos a practicar lo visto!

Box Modeling

Propiedades de la caja

Propiedades de la caja



Todos los elementos del HTML son cajas. Un ``, un `<h2>` y demás, son rectangulares:

- En los elementos de **línea**, se verá uno al lado del otro.
- En cambio en los de **bloque**, uno debajo del otro.



PARA RECORDAR

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.

Propiedades en común

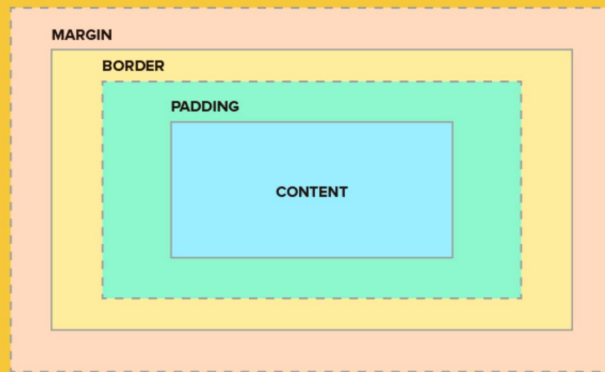
CONTENT: el espacio para el texto o imagen.

PADDING: separación entre el borde y el contenido de la caja. Es un espacio interior.

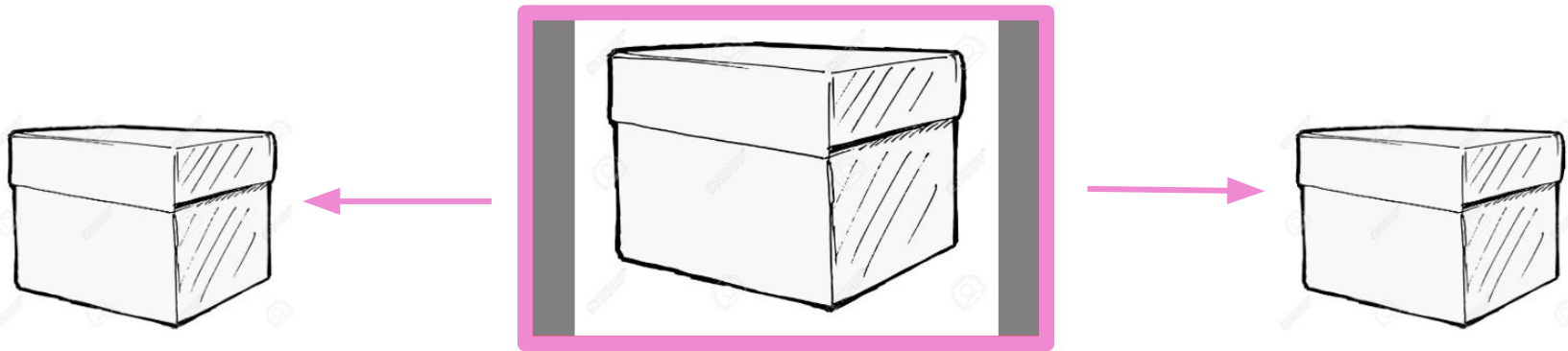
BORDER: el límite entre el elemento y el espacio externo.

MARGIN: separación entre el borde y el afuera de la caja. Es un espacio exterior.

Box Model Css



EJEMPLO 1



Alto y ancho de los elementos

Ancho

🔍 Se denomina `width` a la propiedad CSS que controla la anchura de la caja de los elementos.

🔍 Dicha propiedad no admite valores negativos, y aquellos en porcentaje se calculan a partir de la anchura de su elemento padre.

Alto

🔍 La propiedad CSS que controla la altura de la caja de los elementos se denomina `height`.

🔍 No admite valores negativos, y aquellos en porcentaje se calculan a partir de la altura de su elemento padre.

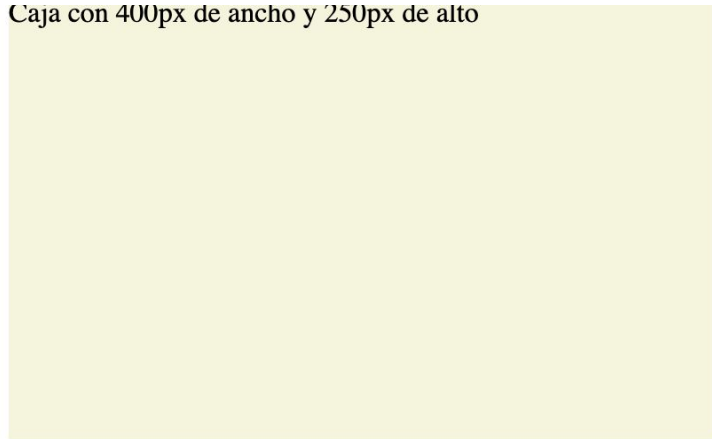
Alto y ancho

CSS

```
div {  
    background-color: beige;  
  
    width: 400px; /* ancho */  
    height: 250px; /* alto */  
}
```

Se ve así

Caja con 400px de ancho y 250px de alto



Valores comunes: unidad (px, porcentaje, rem, viewport) | [Ejemplos y más información.](#)

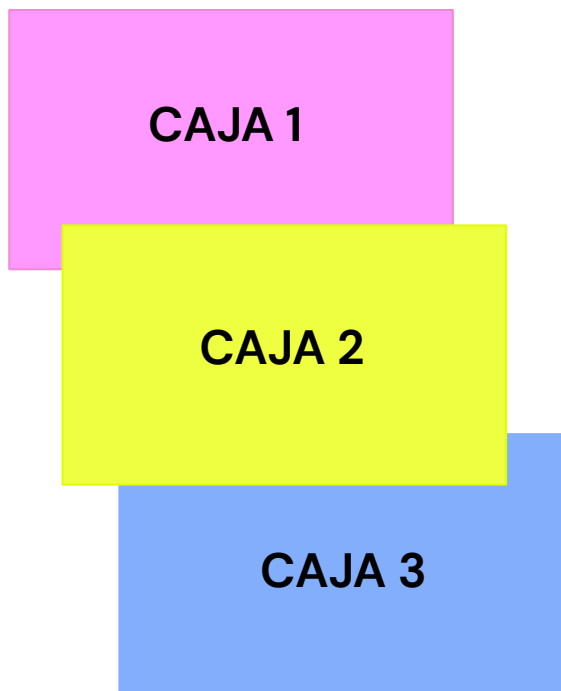
Overflow

Propiedad: **overflow**. Tiene 4 valores posibles:

- **Visible:** valor por defecto. El excedente es visible.
- **Hidden:** el excedente no se muestra (lo corta) → **recomendado.**
- **Scroll:** genera una barra de scroll en los dos ejes (x/y) del elemento, aunque no se necesite.
- **Auto:** genera el scroll solo en el eje necesario.

Veamos cómo se ve aplicando el **overflow: hidden.**

Algo para aclarar



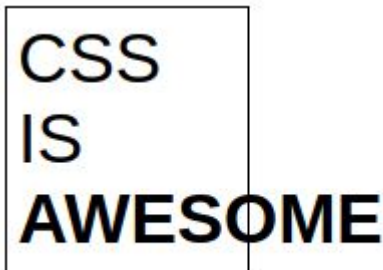
Cuando un elemento tiene un alto o ancho fijos, cualquier contenido que exceda la caja será visible. El inconveniente que esto genera es que, si luego se suma otro contenido, los mismos se van a superponer.

Ejemplo

CSS

HTML

```
<div>  
  CSS IS <strong>AWESOME</strong>  
</div>
```



```
div {  
  /* propiedades decorativas */  
  border: solid 1px black;  
  padding: 5px;  
  display: inline-block;  
  font-size: 32px;  
  font-family: Arial;  
  
  /* propiedades que hacen el "problema" */  
  width: 100px;  
  height: 110px;  
}
```

Solución

CSS

HTML

```
<div>  
  CSS IS <strong>AWESOME</strong>  
</div>
```



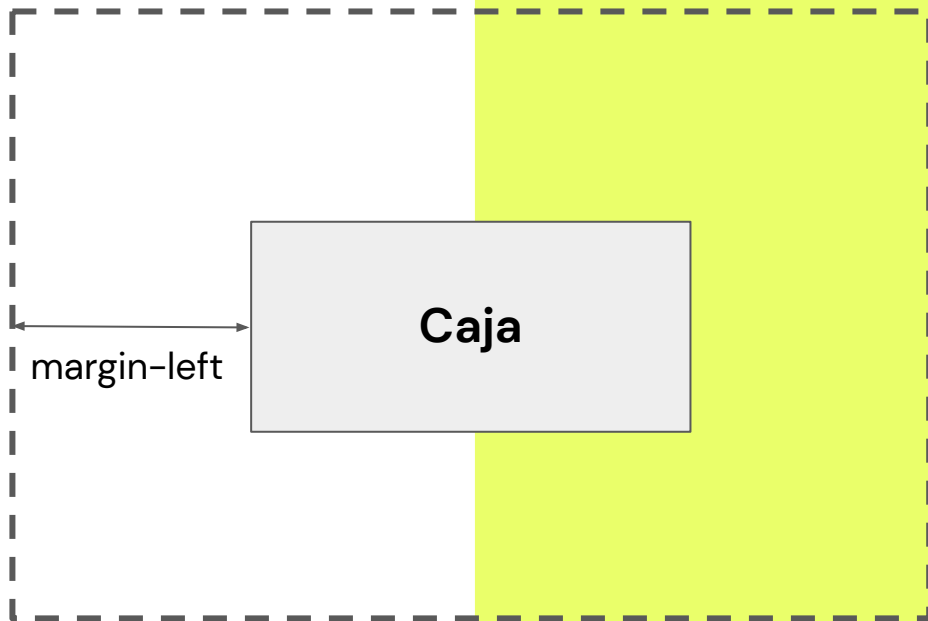
```
div {  
  /* propiedades decorativas */  
  border: solid 1px black;  
  padding: 5px;  
  display: inline-block;  
  font-size: 32px;  
  font-family: Arial;  
  
  /* propiedades que hacen el "problema" */  
  width: 100px;  
  height: 110px;  
  
  /* solucion */  
  overflow: hidden;  
}
```


Espacio exterior

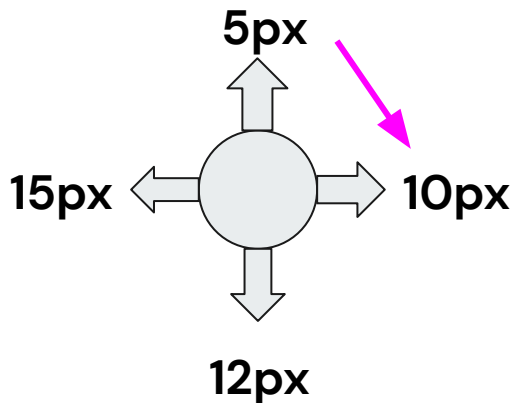
Margin (márgenes)

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.

Puedes definir los 4 lados (forma abreviada "*margin*") o sólo aquellos que necesites.



Código ejemplo



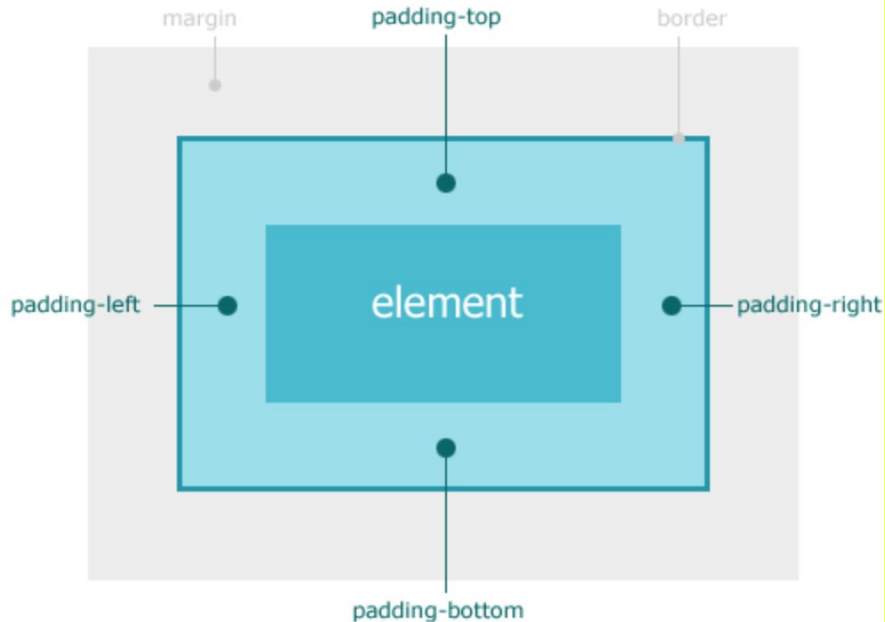
```
div {  
    margin-top: 5px;  
    margin-right: 10px;  
    margin-bottom: 12px;  
    margin-left: 15px;  
}  
  
/* forma abreviada pone en top, right, bottom, left */  
div {  
    margin: 5px 10px 12px 15px;  
}
```

Espacio interior

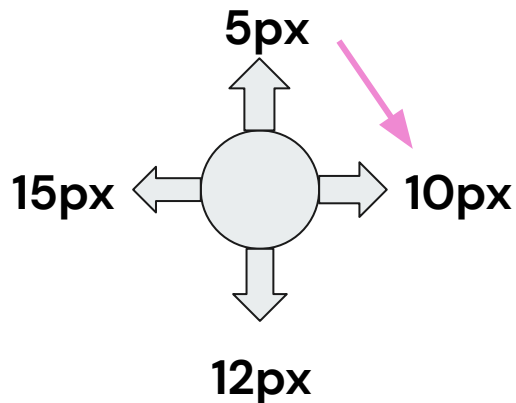
Padding (relleno)

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.

Puedes definir los cuatro lados (forma abreviada "padding") o sólo aquellos que necesites.



Código ejemplo



```
div {  
    padding-top: 5px;  
    padding-right: 10px;  
    padding-bottom: 12px;  
    padding-left: 15px;  
}  
  
/* forma abreviada */  
  
div {  
    padding: 5px 10px 12px 15px;  
}
```

[Más información sobre padding](#) | Nota: se pueden resumir los 4 lados poniendo solo "margin: valor"

Bordes



Border

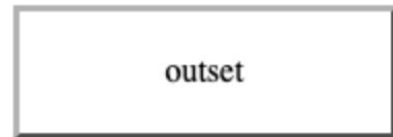
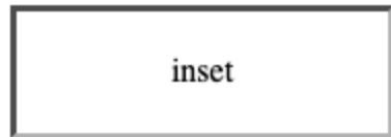
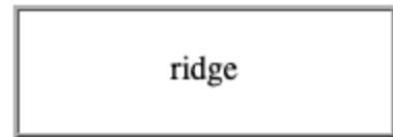
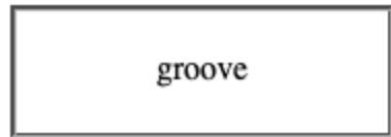
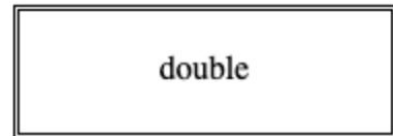
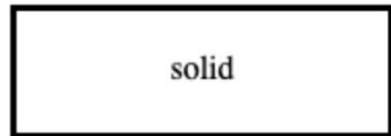
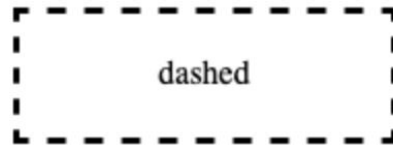
Las propiedades `border-top`, `border-right`, `border-bottom`, y `border-left` se utilizan para definir los bordes de cada lado del elemento por separado.

Puedes definir los cuatro lados (forma abreviada `"border"`) o sólo aquellos que necesites.

Bordes

⇒ A diferencia de los márgenes y padding, los bordes se forman con 3 valores:

- Tipo de borde ([border-style](#)).
- Grosor (-width).
- Color (-color).



none

hidden

Bordes

```
div {  
  border-top:solid 5px red;  
  border-right:solid 10px cyan;  
  border-bottom:solid 7px green;  
  border-left:solid 12px yellow;  
}
```

Se ve así





Ejemplo en vivo

¡Vamos a practicar lo visto hasta ahora!

Display


Tipos de elementos

- El estándar HTML clasifica a todos sus elementos en dos grandes grupos: elementos **en línea (inline)** y **de bloque (block)**.
- Los **elementos de bloque** siempre empiezan en una nueva línea, y ocupan todo el espacio disponible hasta el final de la misma (100%).
- Por otra parte, **los elementos en línea** no empiezan necesariamente en nueva línea y sólo ocupan el espacio necesario para mostrar sus contenidos.

Fuente: <https://developer.mozilla.org/es/>

Tipos de elementos

BLOCK VS INLINE



`display: block;`
Block elements stack, regardless of their width.

`display: inline;`
Inline elements flow from one line to the next.

Tipos de elementos

- Los elementos **en línea** definidos por HTML son aquellos que se usan para marcar texto, imágenes y formularios.
[Ver listado de etiquetas de “en línea”.](#)
- Los elementos **de bloque** definidos por HTML se utilizan para marcar estructura (división de información/código)
[Ver listado de etiquetas de en bloque](#)

Display

Se encarga de definir **cómo se ve un elemento HTML**. Los dos comportamientos más importantes son:

- Pasar un elemento de bloque a uno de línea.
- Pasar un elemento de línea a uno de bloque.

Eso se hace con los valores `block` e `inline` respectivamente:

- **Block:** convierte el elemento en uno de bloque.
- **Inline:** transforma el elemento en uno de línea.

Display

HTML

```
<p>Lorem ipsum dolor sit  
amet, consectetur  
adipiscing elit.  
<span>Laudantium </span>  
perspiciatis itaque  
veritatis ea fugit qui.  
</p>
```

CSS

```
p { /*es un elemento en bloque que  
convierto en línea*/  
  display: inline;  
  background-color: yellow;}  
span { /*es un elemento en línea que  
convierto en bloque*/  
  display: block;  
  background-color: grey;}
```

Con este ejemplo podemos verificar cómo modifico el display de las etiquetas, puedes probar más [acá](#).

Display

Inline-block

Hay una propiedad que permite tomar lo mejor de ambos grupos, llamada *"inline-block"*. Brinda la posibilidad tener *"padding"* y *"margin"* hacia arriba y abajo.

```
li {  
  display: inline-block;  
}
```

Haz clic [aquí](#) para ver más ejemplos.

Tabla comparativa

Dependiendo de si la etiqueta de HTML es “**de bloque**” o “**en línea**”, algunas propiedades serán omitidas ([más información](#)).

	Width	Height	Padding	Margin
Bloque	SI	SI	SI	SI
En línea	NO	NO	Solo costados	Solo costados
En línea y bloque	SI	SI	SI	SI

Quitar un elemento

El display tiene también un valor para quitar un elemento del layout `display: none;` lo oculta, y además lo quita (no ocupa su lugar).

```
div {  
    display: none;  
}
```

Posiciones

Position

Es una propiedad CSS pensada para ubicar un elemento, con una libertad muy flexible. Algunos **ejemplos** de uso:

- Superponer elementos.
- Crear publicidades que te sigan con el scroll o un menú.
- Hacer un menú con submenú dentro.

Valores posibles: relative, absolute, fixed, o sticky (cualquiera excepto static).

¿Cómo ubicar un elemento?

1

Define qué tipo de posición quieres usar.

2

Indica desde dónde calcular la distancia (si será desde arriba, derecha, abajo o izquierda).

3

Determina un valor numérico para las propiedades ***top***, ***bottom***, ***left***, ***right***.

Position

Al aplicar esta propiedad, puedes usar cuatro propiedades para posicionar los elementos, y debes darles un valor numérico.

Elas son:

- **top:** calcula desde el borde superior (ej: `top: 100px`).
- **right:** calcula desde el borde derecho (ej: `right: 50px`).
- **bottom:** calcula desde el borde inferior (ej: `bottom: 100px`).
- **left:** calcula desde el borde izquierdo (ej: `left: 50%`).

Haz clic [aquí](#) para acceder a más información.

Position relative

El elemento es posicionado de acuerdo al flujo normal del documento, y luego es **desplazado *en relación a sí mismo***.

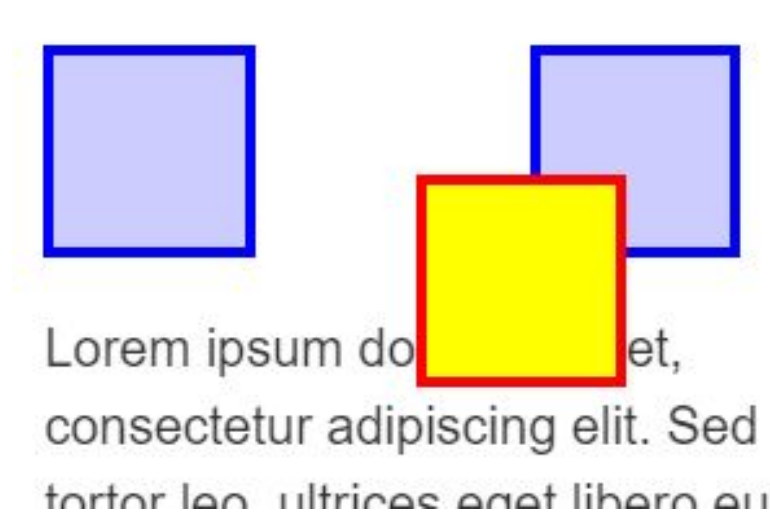
El desplazamiento no afecta la posición de ningún otro elemento, provocando que se pueda superponer sobre otro.

Position relative

CSS

```
div {  
  width: 100px;  
  height: 100px;  
  
  position: relative;  
  top: 40px;  
  left: 40px;  
}
```

Se ve así



Position absolute

El elemento es removido del flujo normal del documento, sin crearse espacio alguno para el mismo en el esquema de la página.

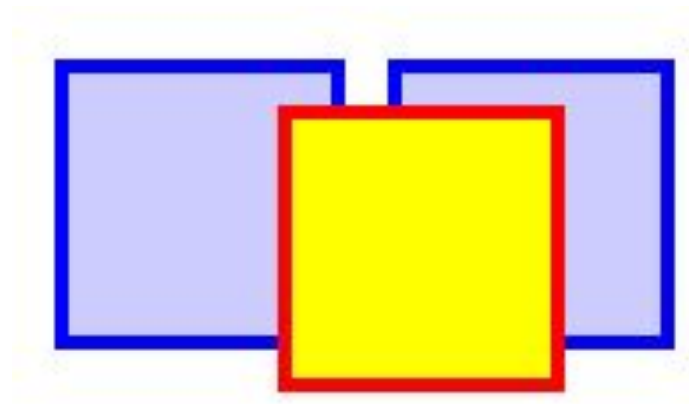
Es posicionado relativo a su padre, siempre y cuando su padre tenga "position:relative". De lo contrario, se ubica relativo al body. Se recomienda establecer un ancho y alto (width, height).

Position absolute

CSS

```
div {  
  width: 100px;  
  height: 100px;  
  
  position: absolute;  
  top: 40px;  
  left: 40px;  
}
```

Se ve así



Position: fixed y sticky

Ambos métodos permiten que el elemento se mantenga visible, aunque se haga scroll.

Fixed

Esta posición es similar a la absoluta, con la **excepción** de que el elemento contenedor es el “**viewport**”, es decir, la ventana del navegador.

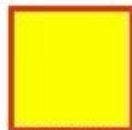
Puede ser usada para crear elementos que floten, y que queden en la misma posición aunque se haga scroll.

Fixed

CSS

```
div {  
  width: 300px;  
  background-color: yellow;  
  
  position: fixed;  
  top: 0;  
  left: 0;  
}
```

Se ve así



Now you can control the position property for the yellow box.



To see the effect of sticky positioning, select the position: sticky option and scroll this container.

Sticky

El elemento es posicionado en el “flow” natural del documento, podría decirse que es un valor que **funciona de forma híbrida, es decir, como “relative” y también “fixed”**.

Esto es, cuando llega el “viewport” (la ventana del navegador) hasta donde se encuentra, se “pegará” sobre el borde superior.

Sticky

CSS

```
div {  
  position: sticky;  
  top: 20px;  
}
```

Se ve así

In this demo you can control the `position` property for the yellow box.



To see the effect of `sticky` positioning, select the `position: sticky` option and scroll this container.

Menú con submenú

- El **position** (tanto *relative* como *absolute*) se usa, entre otras, para hacer un menú que tenga un submenú emergente. Los ítems del primero son relativos, sirven como borde de cualquier hijo.
- La **lista** dentro de un list-item es absoluta. Por defecto, la sublista tiene *display: none*. Recién cuando un list-item detecte el *:hover*, si adentro tiene una lista, dale *display: block*

Menú con submenú

Ejemplo

Este list-item es relative

La lista es absolute.

Como está en el relative, lo usará como límites del top, bottom, left y right

Por defecto la lista es display none
Cuando se pasa el mouse por arriba del li se muestra la sub-lista con display block

```
ul li:hover ul{ display: block }
```

Home	A propos	Contact	Archives	Catégories	
				Apple	
				iPod/iPhone	
				Mac	
				GNU/Linux et Open-Source	
				Scripts Shell	
				Internet Explorer	
				Windows Vista	
				Windows XP	

HTML

```
<ul>
  <li><a href="">Item</a></li>
  <li><a href="">Item</a></li>
  <li><a href="">Item</a>
    <ul>
      <li><a href="">Subitem</a></li>
      <li><a href="">Subitem</a></li>
    </ul>
  </li>
  <li><a href="">Item</a></li>
</ul>
```

CSS

```
ul {
  list-style: none;
  font-size: 0 /* truco por el uso de inline-block */
}
li {
  display: inline-block;
  width: 25%;
  position: relative;
  font-size: 14px
}
ul ul {
  position: absolute;
  display: none;
}
ul ul li {
  display: block;
}
ul li:hover ul {
  display: block;
}
```

Propiedad Z- index

(para el orden de superposición)

El z-index entra en juego cuando dos elementos que tienen *position* se **superponen**. Esta propiedad 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.

Por defecto, todos los objetos tienen z-index:1. Si dos objetos tienen el mismo valor de z-index y se superponen, el que fue creado después en el HTML se verá encima del otro.



Estilos

Crear un archivo HTML y otro CSS. Agregar un estilo de tipografía a un archivo.

Duración: **15 minutos**



ACTIVIDAD EN CLASE

Estilos

Descripción de la actividad:

Crea un archivo HTML con un elemento párrafo dentro de él, con texto Lorem Ipsum.

Además, crea un archivo CSS y agrégale estilo al párrafo con los siguientes valores:

- Color de texto: #0033ff
- Indexar una google font a elección.
- Espacio entre renglones: 15px
- Tamaño de texto: 12px



#Coderalert

Ingresa al manual de prácticas y realiza la segunda actividad “Agregando CSS a nuestro HTML”. Ten en cuenta que el desarrollo de la misma será importante para la resolución del Proyecto Final.



Agregando CSS a nuestro HTML

Descripción de la actividad.

- ✓ Aplicar con CSS las propiedades vistas hasta el momento para modificar textos, encabezados, img, colores, background y box modeling a un archivo HTML de la actividad N°1 "Wireframe y estructura del proyecto".
- ✓ Ejemplo: modificar el valor de los ítems de la lista para que estén ubicados de manera horizontal de nuestro index.

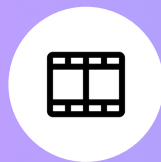
Podrás encontrar un ejemplo en la carpeta de clase.



Primera pre-entrega

En la clase que viene se presentará la primera parte del Proyecto Final, que **nuclea temas vistos entre las clases 1 y 4**.
Recuerda que tendrás 7 días para subirla en la plataforma.

¿Preguntas?



¿Quieres saber más?
**Te dejamos material
ampliado de la clase**



MATERIAL AMPLIADO

Recursos multimedia

Título

- ✓ [Referencias de reglas tipográficas](#) | CSS Reference
- ✓ [Aplicación para generar paletas de colores](#) | Palette App
- ✓ [Aplicación para generar paletas de colores](#) | Lyft Design

Resumen de la clase hoy

- ✓ Tipografía
- ✓ Tipo de fuentes
- ✓ Listas
- ✓ Fondos

Opina y valora
esta clase

Muchas gracias.

#DemocratizandoLaEducación