Esta clase va a ser

• grabada

Certificados oficialmente por

PedidosYa

CODERHOUSE

Clase 04. Desarrollo Web

CSS + BOX MODEL

Certificados oficialmente por

CODERHOUSE

CLASE N°03

Glosario

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.

CLASE N°03

Glosario

Reset CSS: una regla CSS que contiene 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).

Objetivos de la clase

- Conocer y aplicar familias tipográficas desde distintas fuentes externas.
- Comprender y modificar elementos en bloque y en línea.
- Identificar las cajas y sus propiedades.
- Aplicar las propiedades del modelo de cajas para la distribución de los contenidos del sitio.

Temario

03

Incluyendo CSS a nuestro proyecto

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

04

CSS + Box Model

- ✓ Tipografías
- ✓ Box Model
- ✓ Display
- ✓ Posiciones

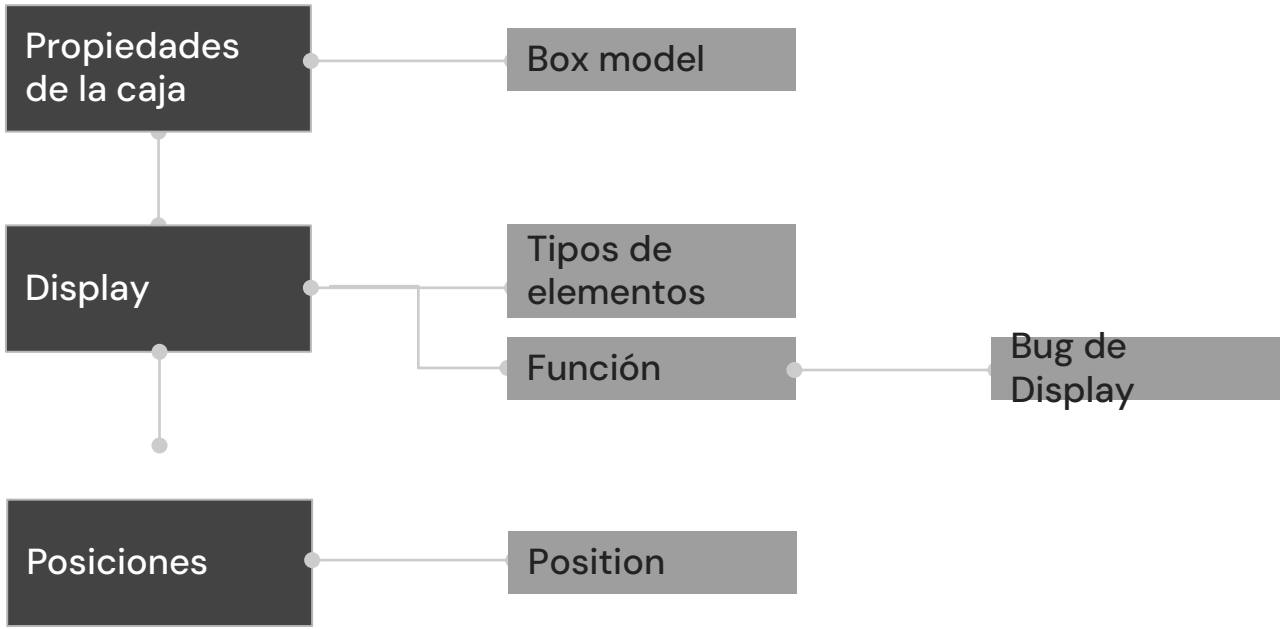
05

Flexbox

- ✓ Flexbox
- ✓ Propiedad de padres e hijos

CODERHOUSE

MAPA DE CONCEPTOS



Recapitulación de la clase anterior

PARA RECORDAR

Conceptos previos

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

- Hacerlo directamente sobre la etiqueta HTML, con el atributo `style=""` ✗
- En el head, insertar la etiqueta `<style>` ✗
- Vincular un archivo externo mediante una etiqueta `<link />`. En su interior incluimos los atributos *rel="stylesheet"*, y *href=""* con la ruta al archivo. ✓

Selección de HTML mediante CSS

Tres posibilidades

Por etiqueta

Por clase
(anteponiendo el ".")

Por ID
(anteponiendo el "#")

```
h1 {
   propiedad: valor;
}
.clase {
   propiedad: valor;
}
#id {
   propiedad: valor;
}
```

CODERHOUSE

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>
```

CODERHOUSE

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>
```

CODERHOUSE

Los nombres de las clases e IDs

No es posible crear nombres separados por espacios.
La escritura *“camel case”* o “*joroba de camello*”, permite que se puedan leer de forma más simple palabras compuestas.

```
claseDeMaquetacion

conBorde

productosMasVendidos
```

Tipografías

Tipografía local

Habíamos visto que usando “font-family”, es posible agregar algunas limitadas fuentes, pero... podemos usar muchísimas opciones de fuentes si las descargamos y las agregamos al directorio de nuestro proyecto.

Tipografía local

CSS

```
@font-face {
    font-family: "Mystery Quest";
    src: url("./fonts/mystery-
quest.ttf");
}
p {
    font-family: "Mystery Quest",
cursive;
}
```

El valor de la propiedad *src* debe indicar en qué parte de nuestro directorio raíz guardamos nuestra tipografía post descarga.
Lo adecuado es crear una carpeta, generalmente llamada ***fonts***, donde almacenaremos todos los archivos de fuentes.

Tipografía web

Veamos qué opciones de fuentes nos ofrece Google Fonts.

CSS

```
h1 {
   font-family: 'Roboto Slab',
serif;
}
```

HTML

```
<head>
   <link
href="https://fonts.googleapis.com
/css?family=Roboto+Slab&display=sw
ap" rel="stylesheet">
   <title>Document</title>
</head>
```

Ejemplo en vivo

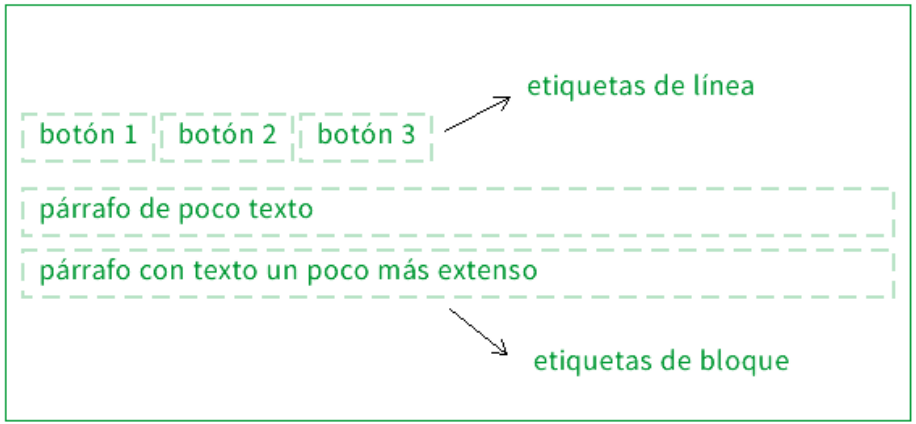
Crearemos un archivo HTML y otro CSS. Agregaremos un estilo de tipografía a un archivo.

Duración: **15 minutos**

CODERHOUSE

Box Model
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 un elemento al lado del otro.
- En cambio en los de **bloque**, uno debajo del otro.

CODERHOUSE

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.

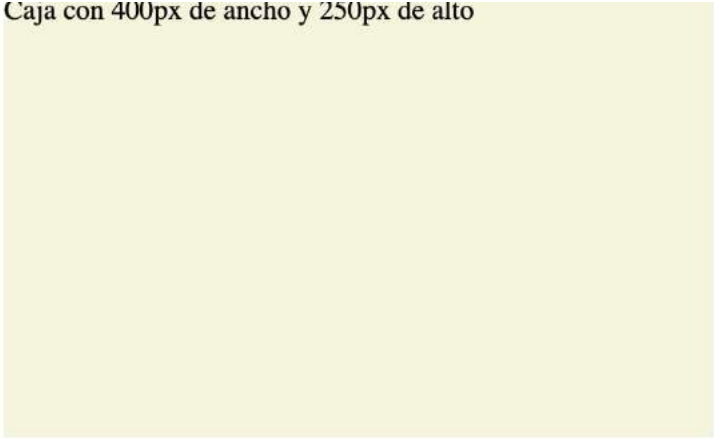
🔎 No admite valores negativos, y aquellos en porcentaje se calculan a partir de la altura de su elemento padre.

Alto y ancho

CSS

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

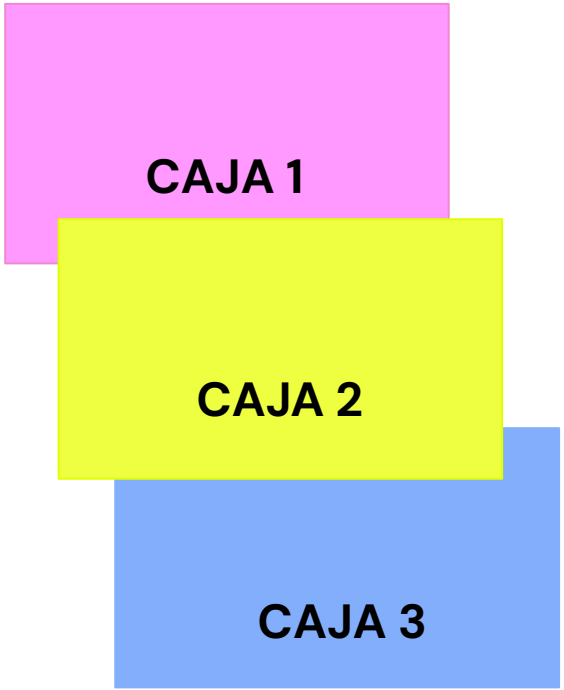
Se ve así



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.



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

HTML

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

CSS

```
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

HTML

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


```
div {
          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 */
     overflow: hidden;
}
```

PARA RECORDAR

Box Model

👉 El concepto de que “todo es una caja”, da lugar al modelo de posicionamiento **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.

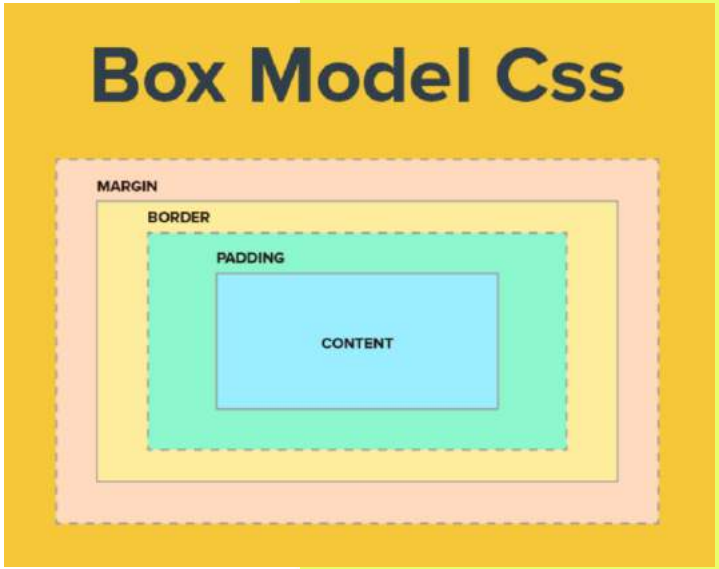
Propiedades en común

CONTENT: el espacio para el texto o imagen.

PADDING: es el espacio libre o “aire” que se deja entre el contenido y el borde de la caja. Es parte del espacio interno del elemento.

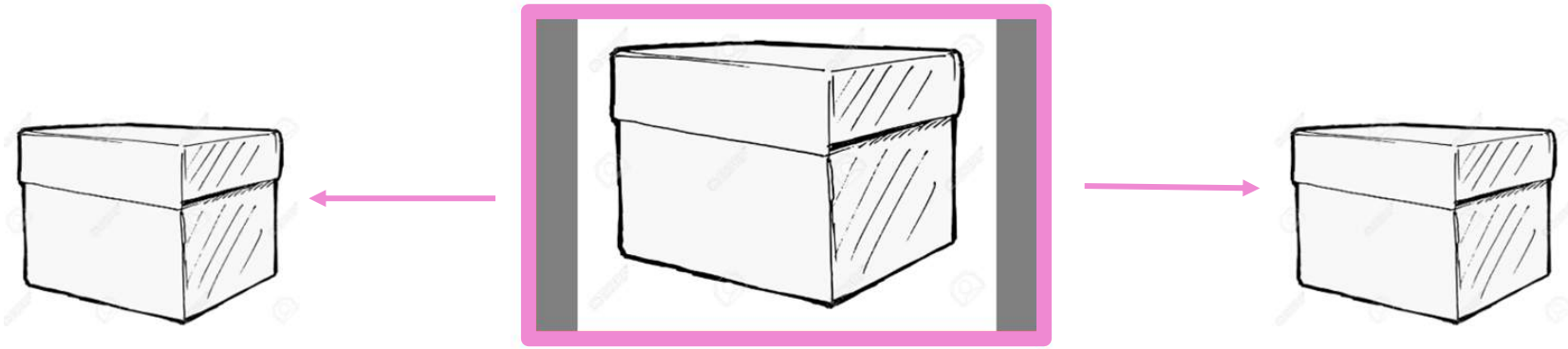
BORDER: el límite entre el elemento y el espacio externo.

MARGIN: separación entre el borde y el afuera de la caja. Es parte del espacio externo.

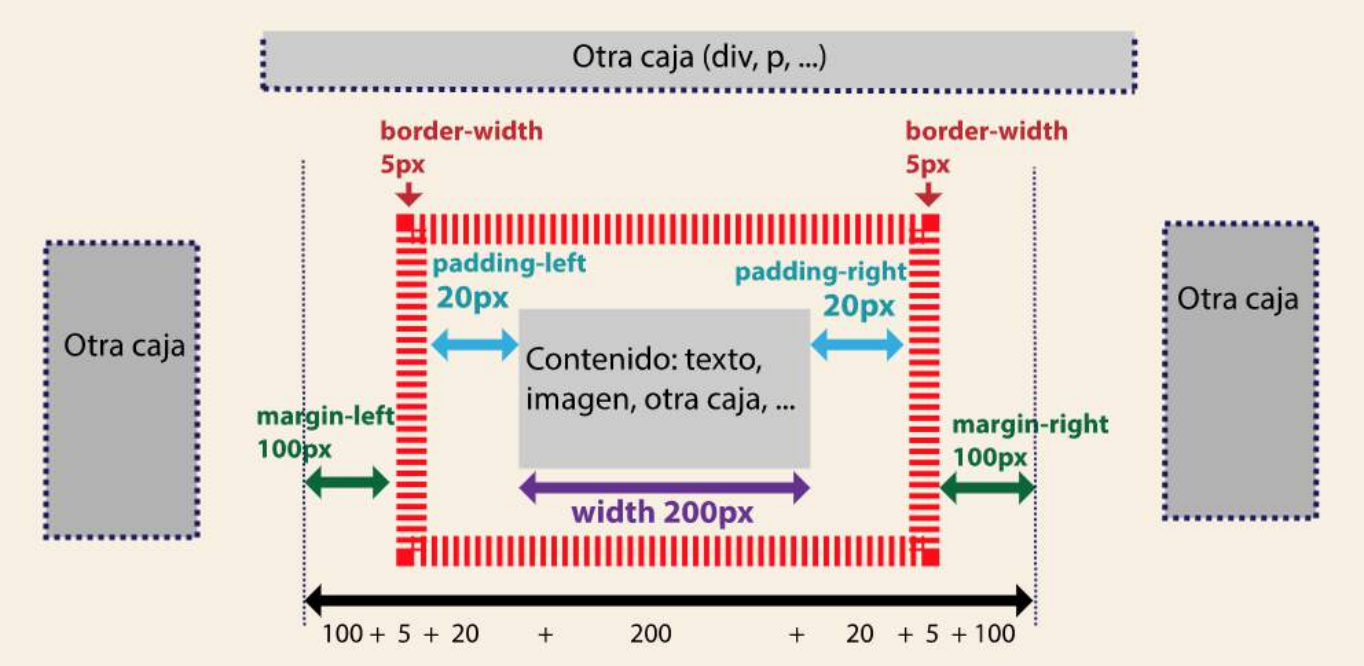


CODERHOUSE

EJEMPLO 1



EJEMPLO 2

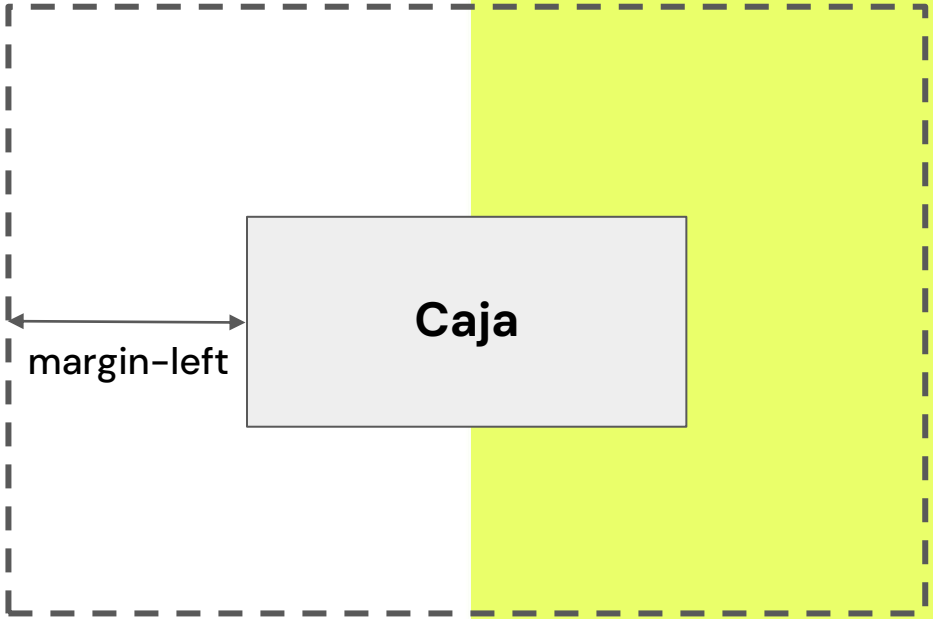


Espacio externo

Margin (márgenes)

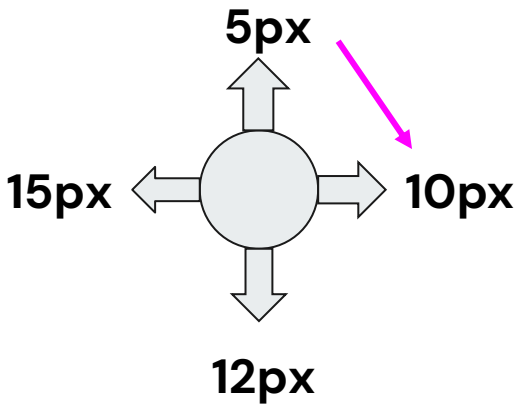
Las propiedades *margin-top, margin-right, margin-bottom y margin-left* se utilizan para definir los márgenes de cada uno de los lados del elemento por separado.

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



CODERHOUSE

Código ejemplo



Nota: se pueden resumir los 4 lados poniendo solo “margin: valor” | Más información sobre Margin

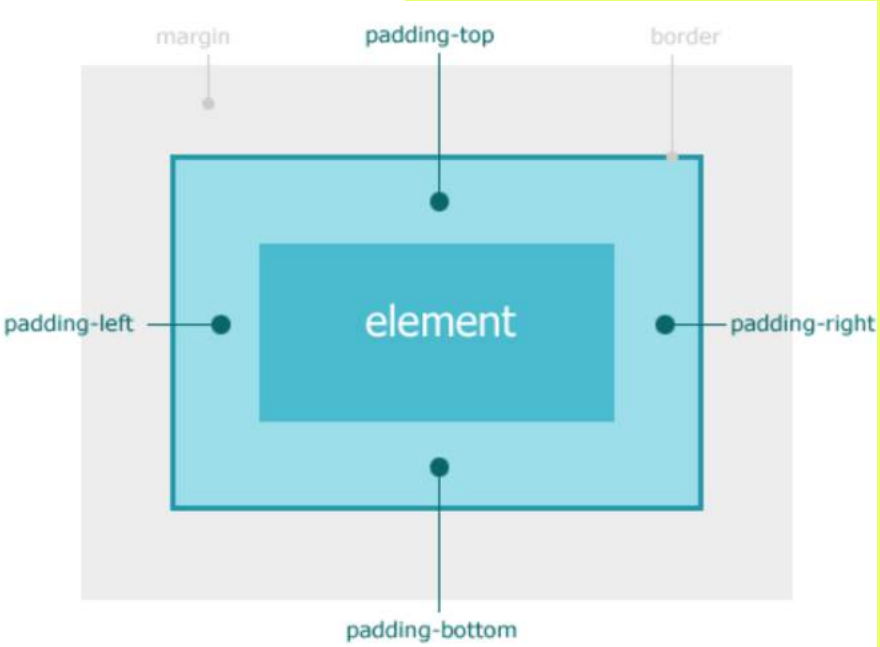
```
div {
    margin-top: 5px;
    margin-right: 10px;
    margin-bottom: 12px;
    margin-left: 15px;
}

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

Espacio interno

Padding (relleno)

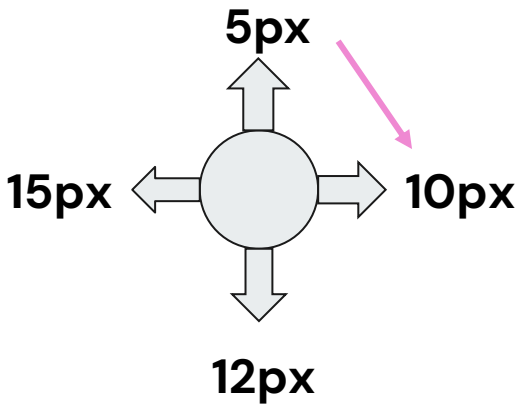
Las propiedades padding-top, padding-right, padding-bottom y padding-left se utilizan para definir los espacios internos de cada uno de los lados del elemento, por separado.

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



CODERHOUSE

Código ejemplo

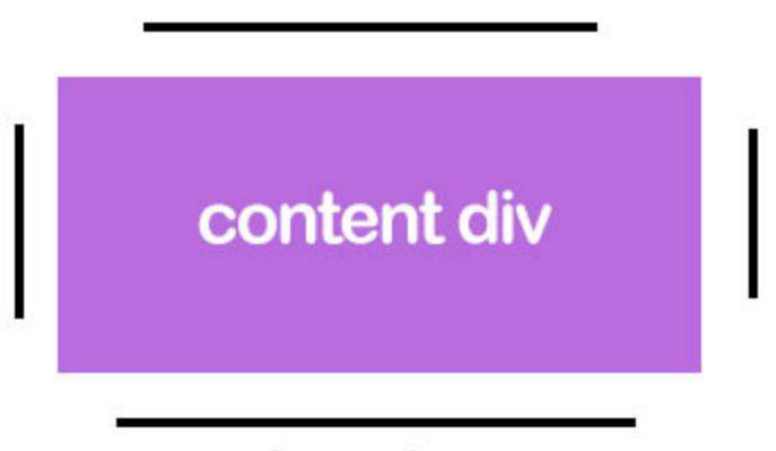


Nota: se pueden resumir los 4 lados poniendo solo “padding: valor” | Más información sobre padding

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

/* forma abreviada */

div {
 padding: 5px 10px 12px 15px;
}
```

CODERHOUSE

Bordes



Border

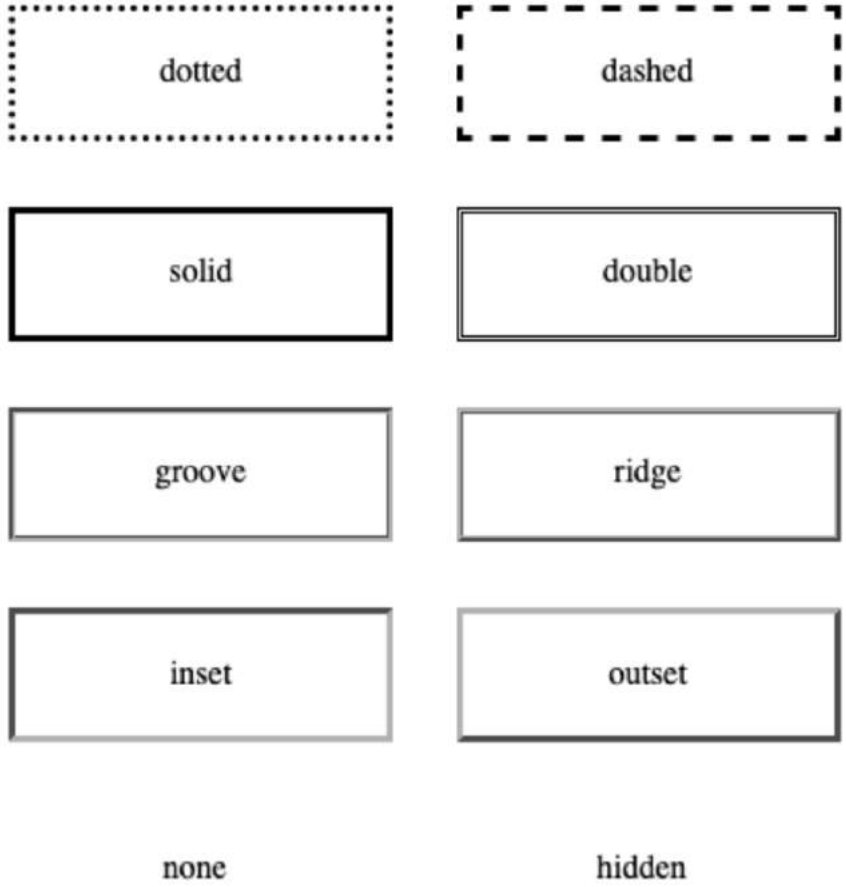
Las propiedades border-top, border-right, border-bottom, y border-left se utilizan para definir los bordes de cada lado del elemento por separado.

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).



Bordes

Se ve así


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

Valores comunes: estilo grosor color|
Ejemplos y más información

Ejemplo en vivo

¡Vamos a practicar lo visto hasta ahora!

Break

¡10 minutos y volvemos!

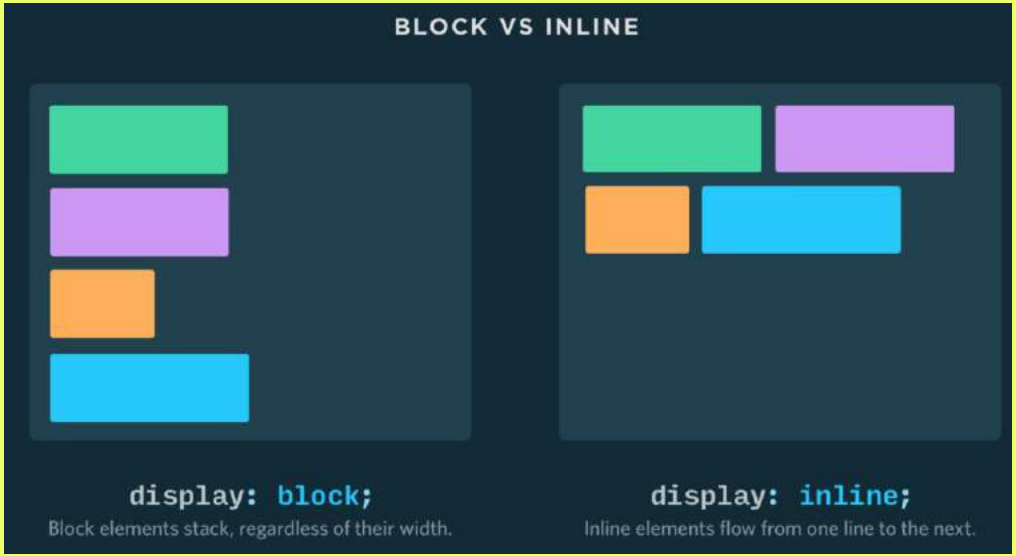
Display

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



Tipos de elementos

- Los elementos en línea definidos por HTML son aquellos que se usan para marcar texto, imágenes y formularios.
  Ver listado de elementos “en línea”.

- Los elementos de bloque definidos por HTML se utilizan para marcar estructura (división de información/código)
  Ver listado de elementos “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
adipisicing 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 de 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: static (es el valor por defecto), relative, absolute, fixed, o sticky.

Position

Al aplicar esta propiedad, puedes usar cuatro propiedades extra para posicionar los elementos, a las cuales debes darles un valor numérico.

- ***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.

¿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 relative

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

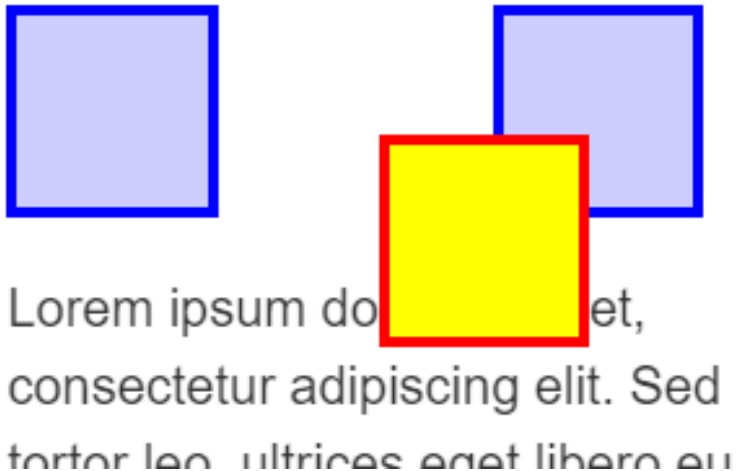
El desplazamiento no afecta la posición de ningún otro elemento, provocando que se pueda superponer sobre otro.

Position relative

CSS

```
div {
    width: 100px;
    height: 100px;

    position: relative;
    top: 40px;
    left: 40px;
}
```

Se ve así



CODERHOUSE

Position absolute

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

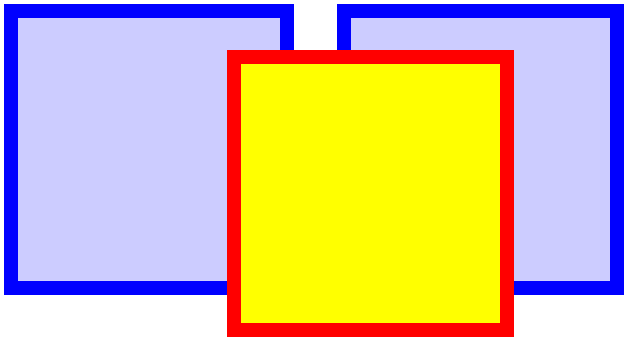
Es posicionado relativo a su padre, siempre y cuando su padre tenga “position:relative”. De lo contrario, se ubica relativo al body. Se recomienda establecer un ancho y alto (width, height).

Position absolute

CSS

```css
div {
    width: 100px;
    height: 100px;

    position: absolute;
    top: 40px;
    left: 40px;
}
```

Se ve así



CODERHOUSE

Position: fixed y sticky

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

CODERHOUSE

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.

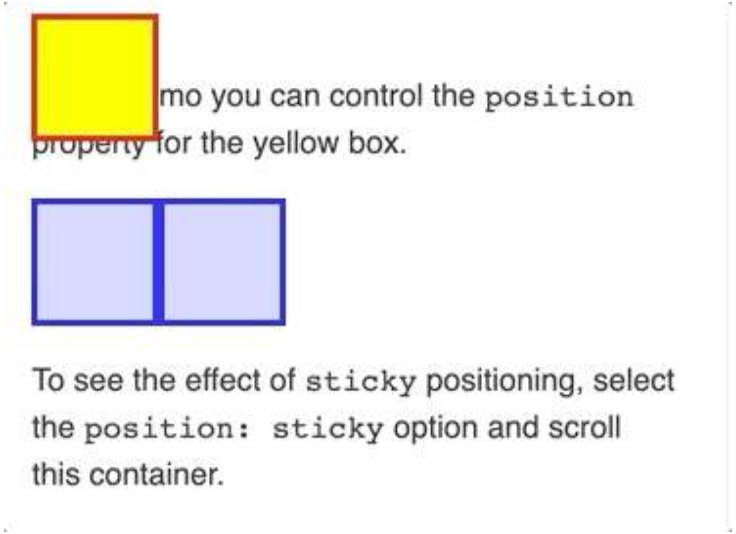
Puede ser usada para crear elementos que floten, y que queden en la misma posición aunque se haga scroll.

Fixed

CSS

```
div {
    width: 300px;
    background-color: yellow;

    position: fixed;
    top: 0;
    left: 0;
}
```

Se ve así



CODERHOUSE

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.

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.

Profundiza y conoce atajos en nuestra Cheat Sheet

CODERHOUSE

#Coderalert

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

Actividad Nº 3

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 de tu Proyecto final

CODERHOUSE

ENTREGA DEL PROYECTO FINAL

Primera entrega

Aclaraciones: Debe tener el nombre “PreEntrega1+Apellido”

Sugerencias: Activar comentarios en el archivo

Consigna

- ✓ Deberás hacer entrega de tu proyecto donde aparezca: Wireframe/prototipo (wireframe de todas tus páginas para vista mobile y desktop) + HTML (uso de etiquetas contenedoras, etiquetas multimedia y etiquetas de texto) + CSS (con modificadores de texto, colores, listas y box modeling).

Objetivos

- ✓ **Prototipar la web** para tener una idea clara del resultado al que quieres llegar.
- ✓ **Maquetar la web:** utilizar los tags, en especial los semánticos, para describir la estructura de la web desde el código.
- ✓ **Crear un estilo inicial:** comenzar a darle estilo básico a la web.

Primera pre-entrega

Se debe entregar

- ✓ **Prototipo de la web:** versión que muestre cómo se verá el sitio cuando esté productivo.
  Crear una estructura donde se organicen los elementos que van a estar en la web. El nivel de detalle no es importante, sino más bien las posiciones que los elementos van a tener y su tamaño aproximado.
  Formato: Archivo PDF o de Imagen

- ✓ **Estructura inicial de la web en HTML:** llevar el contenido a la estructura HTML haciendo uso de los tags que corresponden para el contenido a insertar.
  Incluyendo: **etiquetas semánticas** (tags semánticos de HTML5), **contenido** (agregar etiquetas que van a servir para denotar dónde va a haber contenido como imágenes, párrafos y titulares) y **páginas** (incluir las secciones del sitio ya maquetadas con la estructura propia de cada página)
  Formato: Archivos HTML

Primera pre-entrega

- ✓ **Estilo inicial de la web en CSS:** comenzar a darle estilo básico a una página de su sitio web (ej: index.html) aplicando con CSS las propiedades vistas hasta el momento para modificar textos, encabezados, img, colores, background y box modeling.
  El documento debe estar linkeado en las páginas del proyecto.
  Formato: Archivo CSS

*Flexbox **no debe incluirse en esta entrega.**

¿Preguntas?

Encuesta

Por encuestas de Zoom

Cuéntanos qué temas te resultaron más complejos de entender. **Puedes elegir más de uno**. Vamos a retomar aquellos temas que resultaron de mayor dificultad en el próximo AfterClass.

CODERHOUSE

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

CODERHOUSE

MATERIAL AMPLIADO

Recursos multimedia

Título

- ✓ Referencias de reglas tipográficas | **CSS Reference**
- ✓ Referencias de box model | **CSS Reference**
- ✓ Aplicación para generar paletas de colores | **Palette App**
- ✓ Aplicación para generar paletas de colores | **Lyft Design**

Resumen
de la clase hoy

- ✓ Tipografía
- ✓ Box modeling
- ✓ Display
- ✓ Posiciones

Opina y valora esta clase

CODERHOUSE

Muchas gracias.

#DemocratizandoLaEducación

CODERHOUSE