

INCLUYENDO CSS A NUESTRO PROYECTO

OBJETIVOS DE LA CLASE

- Comprender la sintaxis de CSS
- Incluir CSS en nuestro Proyecto
- Conocer el uso de medidas, colores, fuentes y fondos en CSS

PREMISAS

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

SINTAXIS

```
selector {  
    propiedad1: valor;  
    propiedad2: valor;  
}
```

Ejemplo

```
h1 {  
    color: red;  
}
```

REGLAS SINTÁCTICAS

- Cada declaración CSS está formada por un juego de pares **propiedad: valor;**
- No se ve afectado por el espacio en blanco. Las propiedades se pueden escribir de corrido o una debajo de la otra.
- Siempre que la propiedad represente un número, el valor debe indicar en qué unidad se expresa.

PADRE E HIJOS

Cuando tienes una etiqueta “dentro” de otra, lo que haces es aplicar el concepto de padres e hijos.

En este caso, **section** es padre de **article** y, a su vez, **article** es padre del **h2** y del **p**.

```
<section>
  <article>
    <h2> Título </h2>
    <p> Lorem ipsum dolor sit amet,
      consectetur adipiscing elit, sed
      do eiusmod tempor incididunt ut
      labore et dolore magna aliqua. Ut
      enim ad minim veniam, quis
      nostrud exercitation ullamco
      laboris nisi ut aliquip ex ea
      commodo consequat.
    </p>
  </article>
</section>
```

PADRE E HIJOS

Esto habilita a agregar atributos específicos a “hijos”, sin alterar los del “padre”.
Un padre puede tener muchos hijos, y todos ellos heredan sus características, pudiendo tener también características particulares.

Selector
PADRE

Selector **HIJO**

```
section article {  
  background-color: #cccccc;  
  width: 500px;  
  height: 500px;  
}
```

PADRE E HIJOS

En este caso, se observa la forma correcta de declarar cada estilo. Cuando quieres seleccionar una etiqueta, debes incluir las etiquetas padre/s para que sean más específicas a la hora de aplicar estilos.

```
<section>
  <article>
    <h2> Título </h2>
    <p> Lorem ipsum dolor sit amet,
      consectetur adipiscing elit, sed do
      eiusmod tempor incididunt ut labore et
      dolore magna aliqua. Ut enim ad minim
      veniam, quis nostrud exercitation
      ullamco laboris nisi ut aliquip ex ea
      commodo consequat.
    </p>
  </article>
</section>
```

```
section {
  padding: 50px 30px 20px 60px;
  margin-left: 40px;
}

section article {
  background-color: #cccccc;
  width: 500px;
  height: 500px;
}

section article p {
  line-height: 4;
}
```


INSERTAR CSS EN EL HTML

Forma externa

1

Forma **EXTERNA**: dentro de la etiqueta **<head>**, llamas al archivo CSS que necesites (recuerda el uso de rutas relativas y absolutas).

```
<link rel="stylesheet" href="archivo.css" />
```

INSERTAR CSS EN EL HTML

2

Forma interna

Forma **INTERNA**: es recomendable que esté dentro de la etiqueta **<head>**. Puede estar en **<body>**, pero sería más desordenado.

```
<style>  
    /* comentario de CSS, dentro de esta etiqueta, va el  
    codigo CSS, */  
</style>
```

INSERTAR CSS EN EL HTML

3

Forma interna

Otra forma **INTERNA**, muy poco recomendable, consiste en usar para “parches” específicos, o pruebas. Se hace difícil mantenerlo.

```
<h1>Un encabezado sin formato</h1>  
<h2 style="CODIGO CSS">H2 con formato CSS</h2>  
  
<p>Párrafo sin formatear</p>  
<p style="CODIGO CSS">Párrafo formateado</p>  
<p>Otro párrafo sin formatear</p>
```

CLASS

👉 Generalmente se utiliza para darle estilos a cierta parte del código. Por ejemplo, si quieres que una imagen tenga bordes, y que además sean redondeados.

CLASS DESDE CSS

Desde CSS, **puedes usar los nombres que quieras**, siempre y cuando empiecen con **LETRAS**, y pongas un “.” adelante. Lo recomendable es poner un nombre que haga referencias a los estilos que tendrá. Por ejemplo:

```
.bordesRedondeados {  
  /* codigo CSS */  
}
```

HTML: ATRIBUTO CLASS=""

En el HTML, para aplicar una clase debes usar el atributo **“class”**, y luego colocar en el **valor** el **nombre de la clase** (que has especificado en CSS).

```
<img src="" class="bordesRedondeados" />
```

MÁS DE UNA CLASS

Puedes aplicar **más de una clase** a cada etiqueta separada por un espacio. De esta manera, podrás tener estilos diferenciados para cada clase.

```
<img src="" class="bordesRedondeados imgChica"/>
```

ID

👉 Generalmente se usa para nombrar porciones de código y sectores, como por ejemplo cuando quieres nombrar distintas secciones.

👉 Es posible ponerle ID a cualquier elemento HTML para darle un "nombre". Y así como el ID, todos los elementos también aceptan el atributo `class=""`.

- Dicha clase se utiliza cuando quieres aplicar el mismo estilo a más de un elemento, y la búsqueda por etiqueta no sirve para lograrlo.
- 👉 No necesitas escribir varias veces el mismo CSS, ni repetir el ID.

ID DESDE CSS

Desde CSS, **puedes usar los nombres que quieras**, siempre y cuando empiecen con **LETRAS**, y pongas un “#” adelante. Lo recomendable es poner un nombre que haga referencias a los estilos que tendrá. Por ejemplo:

```
#productos {  
  /* codigo CSS */  
}
```

HTML: ATRIBUTO ID=""

Para aplicar un ID en el HTML, debes usar el atributo “id”, y luego en el valor el nombre del ID (que has especificado en CSS). Por ejemplo:

```
<section id="productos">
```

```
</section>
```

COMPARACIÓN CLASS VS. ID

	¿Se puede reutilizar su nombre en el HTML?	¿Se puede usar varias veces en un atributo en el HTML?	¿Cuándo lo uso?
ID	NO	NO	Nombrar secciones, divisiones de código
CLASS	SI	SI	Especificar diseño aparte del código
Ejemplo ID	id="productos" id="productos2"		<section id="productos">
Ejemplo CLASS	class="bordes" class="bordes"	class="bordes destacado"	<p class="destacado">

EJEMPLO

HTML:

```
<section id= "prod">
  <article class= "rojo">
  </article>
  <article id= "prod">
  </article>
</section>
```

HTML:

```
<section id= "prod">
  <article class= "rojo">
  </article>
  <article class= "rojo">
  </article>
</section>
```

Tanto **ID** como **Class** pueden ser utilizadas dentro del html en diferentes etiquetas. Sin embargo, **los nombres otorgados a las clases se pueden repetir**, mientras que utilizados en **los IDs no**.

HERENCIA

En general, estas propiedades son intuitivas. Por ejemplo, podrás heredar de un elemento padre el tamaño de letra y color de la misma, *a menos que el elemento hijo tenga otros estilos aplicados*. Puedes ver más al respecto [aquí](#).

```
div {  
  color: red;  
}
```

```
<div>  
  <p>Este párrafo quedará  
  en rojo, por herencia</p>  
</div>
```

¡PARA PENSAR!- CASCADA

El navegador lee de arriba hacia abajo (forma de cascada)

¿De qué color crees que se aplicará al párrafo (p) al ver el siguiente código?

```
p {  
    color: red;  
}  
  
p {  
    color: green;  
}
```

PRECEDENCIA DE DECLARACIONES

Cuando reglas distintas apuntan al mismo objeto:

- Si son **propiedades distintas**, se suman (se combinan).
- Si tienen **alguna propiedad repetida**, sólo una queda.

Esto es lo que se denomina *precedencia*.

- **ID** pisa cualquier otra regla.
- **Class** sobrescribe las reglas de etiqueta, pero no las de **ID**.
- **Etiquetas** tienen la menor precedencia.

ID > Class > Etiquetas

ESTILOS INLINE

Si utilizas estilos inline, sobrescribirán cualquier estilo de las páginas externas de CSS.

Se podría decir que los estilos inline son los que tienen una mayor especificidad, por lo tanto:



no es recomendable utilizarlos en tu página.

```
<p style= "color: red">Párrafo rojo</p>
```

!IMPORTANT;

- Si tienes 3 reglas CSS, es poco probable que “choquen”, pero en un CSS extenso es más común.
- La declaración ***!important;*** corta la precedencia. Se escribe después del valor de la propiedad CSS que se quiere convertir en la más importante. Se utiliza un ***!important;*** por cada valor a pisar.

 Si necesitas más de 5 ***!important;*** en todo tu CSS, algo estás haciendo mal.

RESETEO CSS

👉 Los **reset CSS** contienen en su código fuente definiciones para propiedades problemáticas, que los diseñadores necesitan unificar desde un principio.

Por ejemplo, la mayoría de navegadores establece un margen por defecto entre el contenido de la página web y su propia ventana, cuyo valor varía de un navegador a otro.

RESETEO CSS

- 🙌 Para subsanar esa diferencia, los diseñadores y las diseñadoras de sitios webs suelen declarar la siguiente línea al comienzo de sus hojas de estilo:

```
* {  
    Margin:0;  
    padding:0;  
}
```

PROPIEDAD: COLOR

👉 Mediante esta propiedad, podrás agregar color a los textos de tu sitio

pero... **¿cómo se eligen los colores?** 😞

TIPOS DE VALORES PARA COLOR

Existen distintos valores, pero **nos centraremos en 3:**

- Por nombre del color (ej: red).
- Hexadecimal (ej: #ffffff).
- RGB (por ejemplo: 50, 212, 227). Si agregas un valor más, puedes manejar su opacidad. (red, green, blue) cada color permite hasta 256 valores.

LIST-STYLE-TYPE

CSS

```
ol {  
  list-style-type: none;  
}  
ul {  
  list-style-type: none;  
}
```

Aplicando esta propiedad y este valor, vamos a poder eliminar las bullets y los números.

Valores posibles: ([ver aqui](#))

FONT-STYLE

CSS

```
.normal {  
    font-style: normal;  
}
```

```
.italica {  
    font-style: italic;  
}
```

Valores comunes: normal | italic

FONT-WEIGHT

CSS

```
.negrita {  
    font-weight: bold;  
}  
  
.normal {  
    font-weight: normal;  
}
```

Valores comunes: normal | bold

FONT-SIZE

CSS

```
.textoGrande {  
    font-size: 20px;  
}
```

```
.textoRelativo {  
    font-size: 200%;  
}
```

Valores posibles: <medida de longitud> | <porcentaje>

FONT-FAMILY

CSS

```
.impact {  
    font-family: Impact, sans-serif;  
}  
  
.comicSans {  
    font-family: "Comic Sans MS",  
    sans-serif;  
}
```

Valores posibles: <familia o nombre genérico>

FONT-FAMILY

Cada sistema operativo y navegador interpretan de distinta forma las fuentes predeterminadas.

- **Serif:** «Times New Roman» en Windows, y «Times» en Macintosh (diferente a la de Windows).
- **Sans serif:** «Arial» en Windows, y «Helvetica» en Macintosh.
- **Monospace:** «Courier New» en Windows, «Courier» en Macintosh, y por lo general «VeraSans» o «DejaVuSans» en Linux.

TEXT-ALIGN

CSS

```
.centrar {  
    text-align: center;  
}
```

```
.aLaDerecha {  
    text-align: right;  
}
```

Valores posibles: left | right | center | justify

TEXT-DECORATION

CSS

```
.subrayado {  
    text-decoration: none;  
}  
  
.tachado {  
    text-decoration: line-through;  
}
```

Valores posibles: none | underline | overline | line-through

BACKGROUND-COLOR

CSS

```
.fondoFuerte {  
    background-color: yellow;  
}
```

Valores posibles: [color]

BACKGROUND-IMAGE

CSS

```
.catsandstars {  
    background-image:  
url("https://mdn.mozillademos.org/  
files/11991/startransparent.gif"),  
url("https://mdn.mozillademos.org/  
files/7693/catfront.png");  
}
```

Link para descargar iconos:
<https://www.flaticon.com/>

Valores posibles: url | none

UNIDADES DE MEDIDAS

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

UNIDADES DE MEDIDAS

Ahora veamos qué medida es más conveniente para los textos.

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

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 a nuestro directorio raíz.

TIPOGRAFÍA LOCAL

CSS

```
@font-face {  
  font-family: "Mystery Quest";  
  src: url("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.

TIPOGRAFÍA WEB

Habíamos visto que usando “*font-family*”, es posible agregar algunas limitadas fuentes, pero... podemos usar muchísimas opciones de fuentes con “**Google Fonts**”.

TIPOGRAFÍA WEB

CSS

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

HTML

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

Ver [Google Fonts](#)

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



LOS NOMBRES DE LAS CLASES E IDS

No es posible crear nombres separados por espacios.

“*kebab-case*”, es otra metodología de nombrado permite que se puedan leer de forma más simple un conjunto de palabras separandolas con un guion.

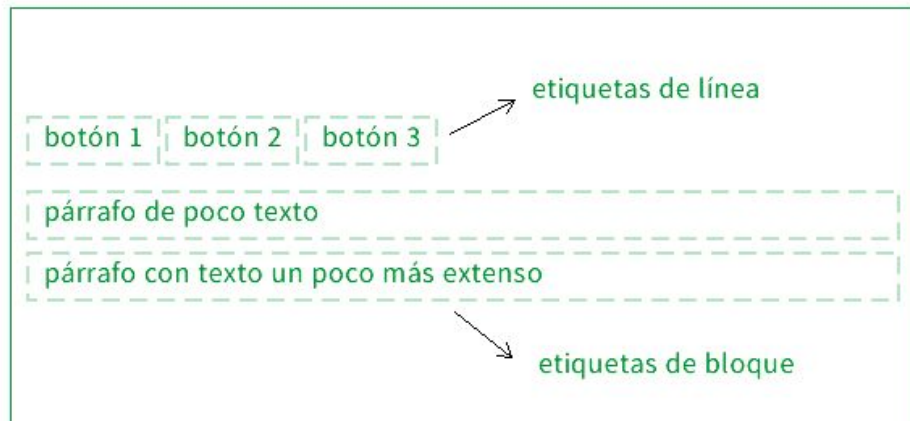
clase-de-maquetacion

con-borde

productos-mas-vendidos

PROPIEDADES DE LA CAJA

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



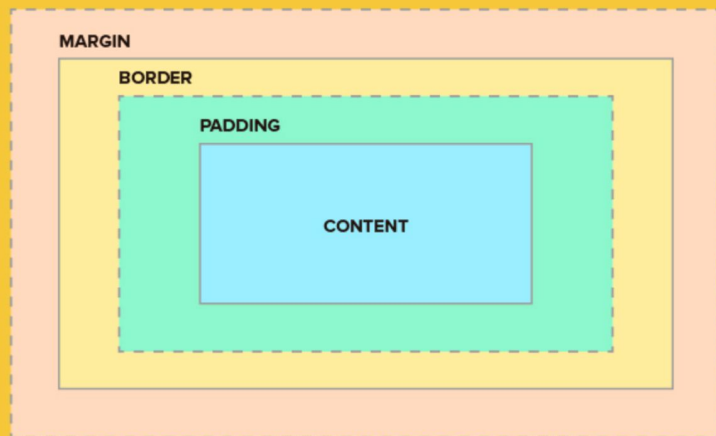
- En los elementos de **línea**, se verá uno al lado del otro.
- En cambio en los de **bloque**, uno debajo del otro.

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

Box Model Css



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.

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.

Alto

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

 No admite valores negativos.

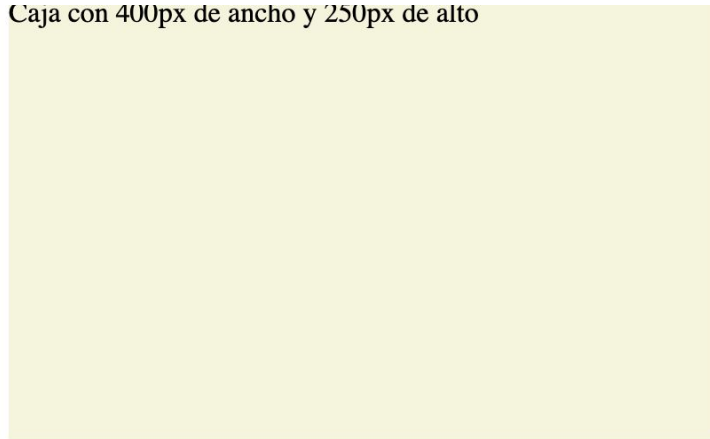
ALTO Y ANCHO

CSS

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

Se ve así

Caja con 400px de ancho y 250px de alto



OVERFLOW

Propiedad: **overflow**

Tiene 4 valores posibles:

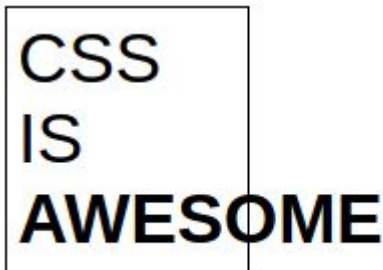
- **Visible:** valor por defecto. El excedente es visible.
- **Hidden:** el excedente no se muestra (lo corta) → **recomendado.**
- **Scroll:** genera una barra de scroll en los dos ejes (x/y) del elemento, aunque no se necesite.
- **Auto:** genera el scroll solo en el eje necesario.

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

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

CSS

HTML

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



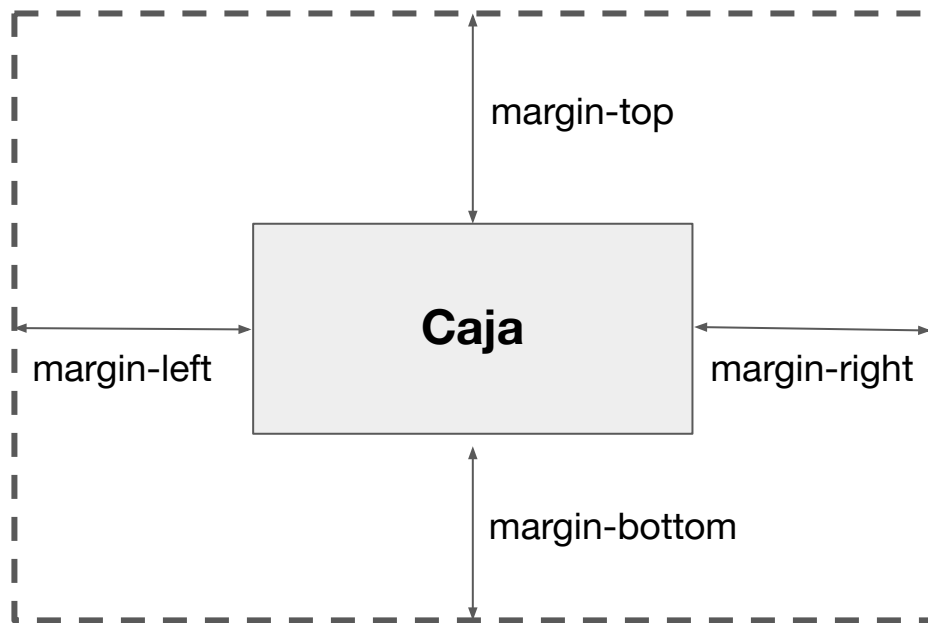
```
div {  
  /* propiedades decorativas */  
  border: solid 1px black;  
  padding: 5px;  
  display: inline-block;  
  font-size: 32px;  
  font-family: Arial;  
  
  /* propiedades que hacen el "problema" */  
  width: 100px;  
  height: 110px;  
  
  /* solucion */  
  overflow: hidden;  
}
```


ESPACIO EXTERIOR

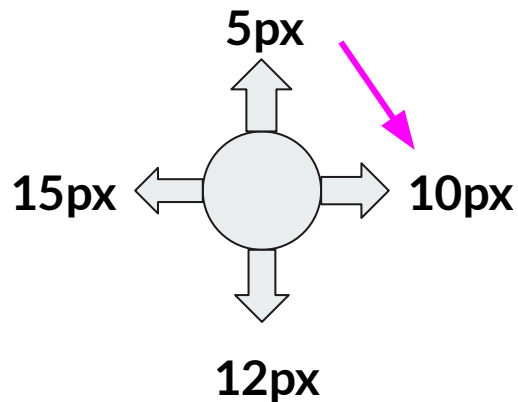
Margin (márgenes)

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



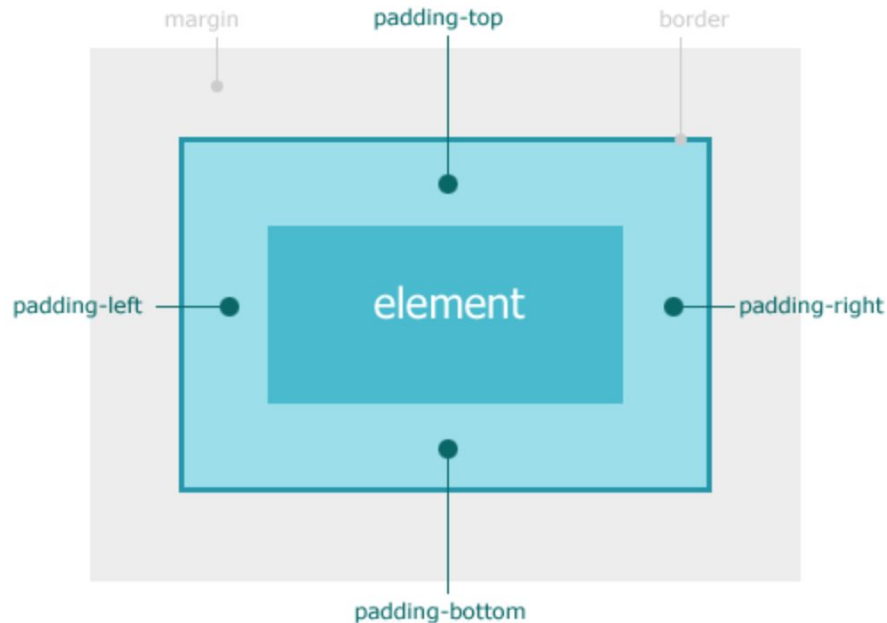
```
div {  
    margin-top: 5px;  
    margin-right: 10px;  
    margin-bottom: 12px;  
    margin-left: 15px;  
}  
  
/* forma abreviada pone en top, right, bottom, left */  
div {  
    margin: 5px 10px 12px 15px;  
}
```

ESPACIO INTERIOR

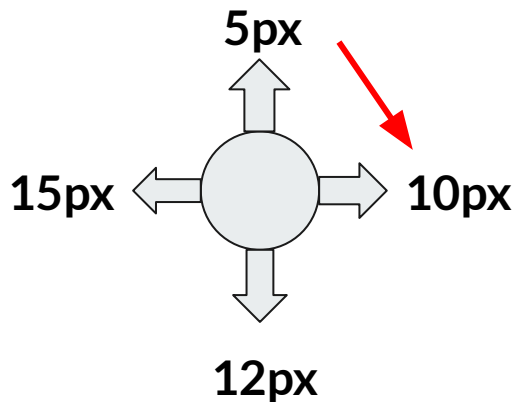
Padding (relleno)

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;  
}
```

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.

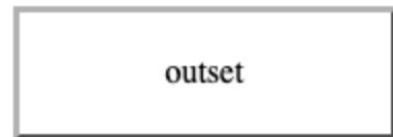
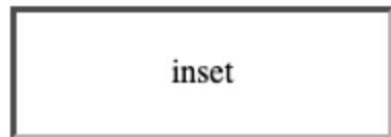
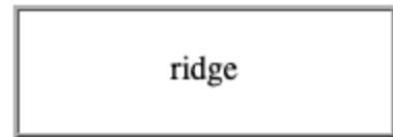
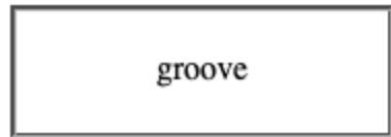
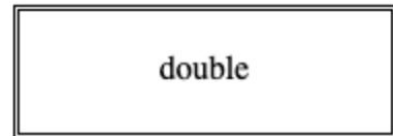
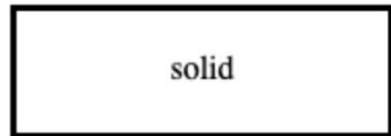
Puedes definir los cuatro lados (forma abreviada “`border`”) o sólo aquellos que necesites.



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

CSS

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

Se ve así



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.

TIPOS DE ELEMENTOS

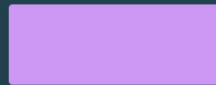
Representación gráfica

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

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 ejemplo

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;  
}
```

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 adentro.

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.

Ellas 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***.

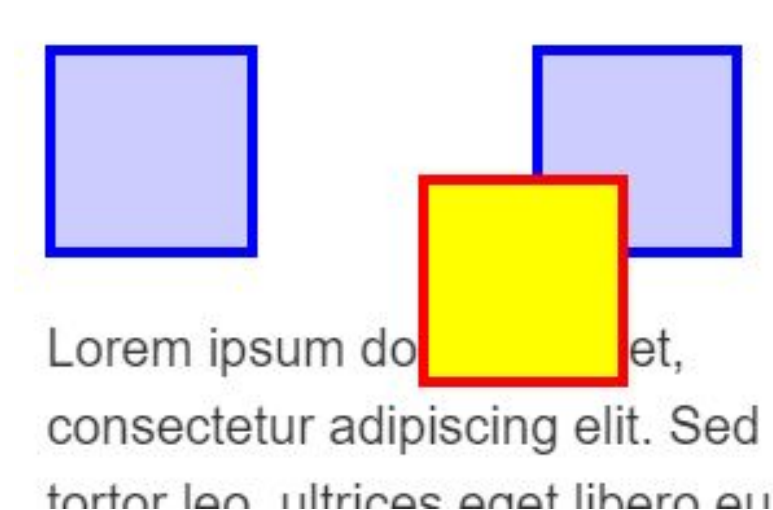
El desplazamiento no afecta la posición de ningún otro elemento, provocando que se pueda superponer sobre otro.

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.

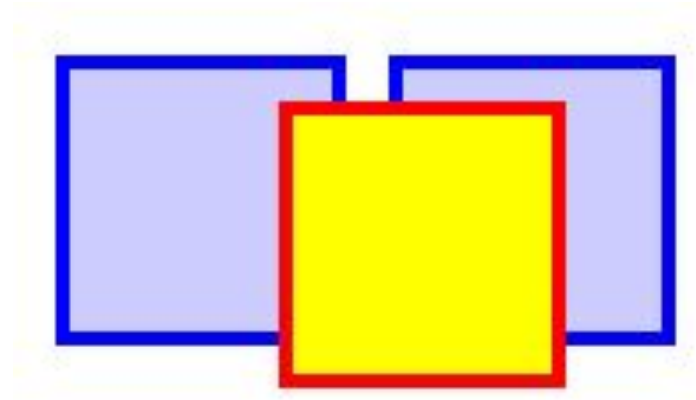
Es posicionado relativo a su padre, siempre y cuando su padre tenga “position:relative”. De lo contrario, se ubica relativo al body. Se recomienda establecer un ancho y alto (width, height).

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.