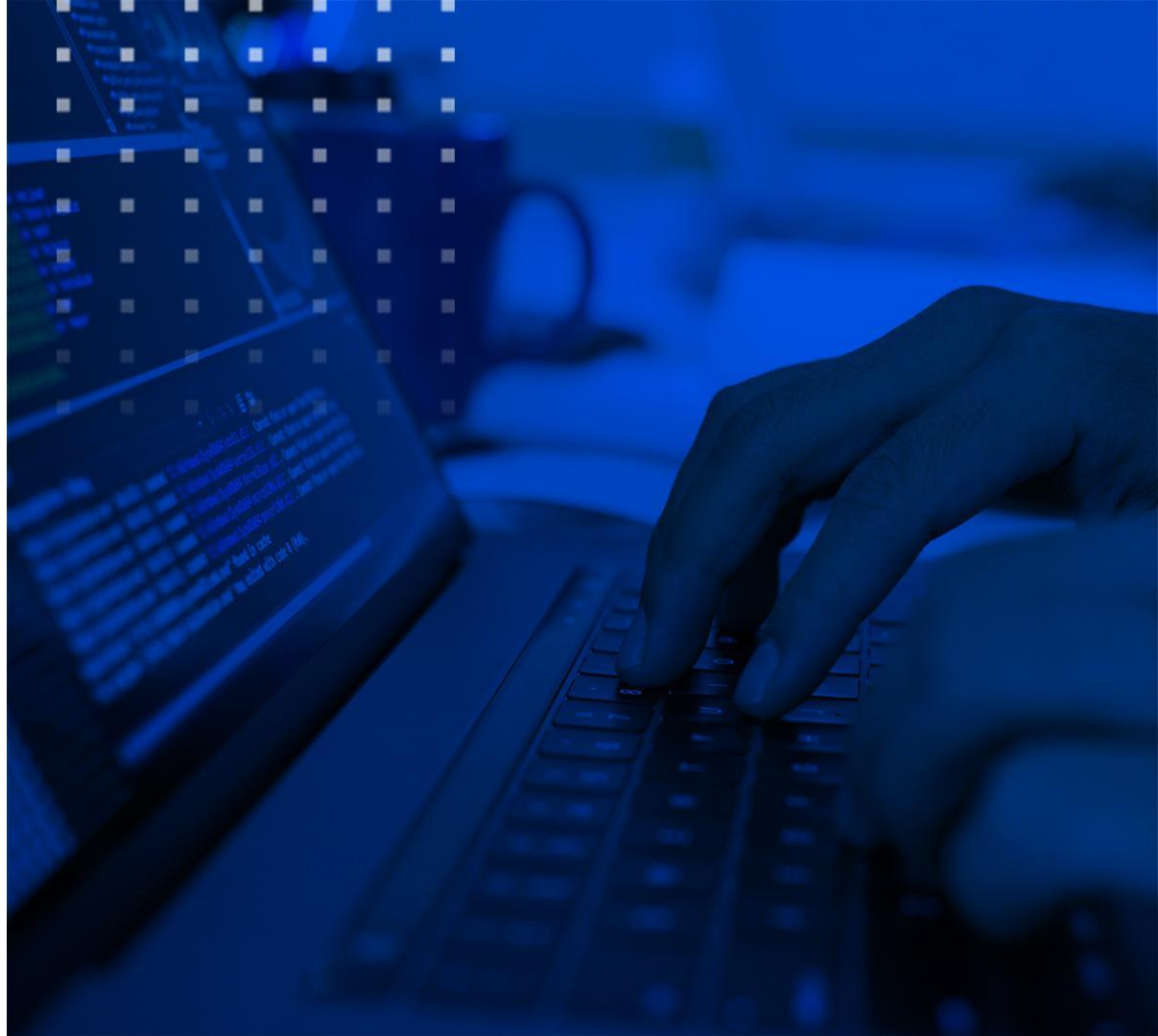


# CURSO

FULL STACK DEVELOPER  
NIVEL INICIAL

UNIDAD 2

Front-End: Estilos con CSS



# Full Stack Developer Inicial

Front-End: Estilos con CSS

# Objetivos del módulo...

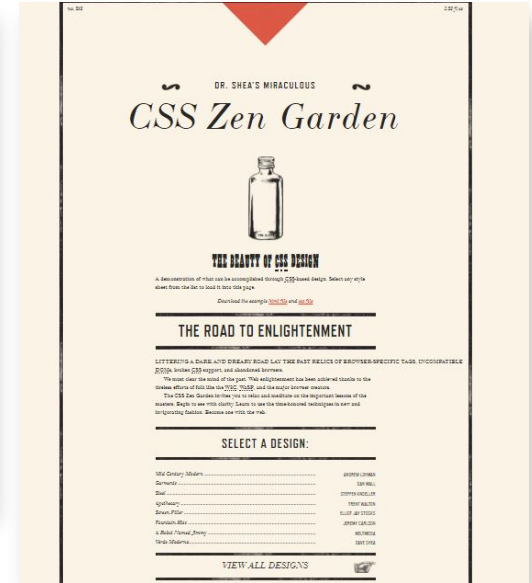
- Comprender la sintaxis de CSS.
- Incluir CSS en un proyecto.
- Conocer el uso de medidas, fondos y fuentes en CSS.
- Aplicar las propiedades en un proyecto

# BASES DE CSS

# PREMISAS

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.

CSS bien implementado permite cambiar **todo el diseño** de un sitio web, sin modificar el HTML. Las siguientes dos imágenes corresponden al mismo código HTML pero distinto CSS:



# SINTAXIS CSS

# SINTAXIS

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

Ejemplo

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



# PADRE E HIJOS

# 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, con
elit. Temporibus cum magnam eos a c
="#"> possimus </a> iure, eius nemo
debitis!
  </p>
</article>
</section>
```

# PADRE E HIJOS

Esto te habilita a agregar propiedades específicas 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 **HIJO**



Selector **PADRE**



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

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.

## HTML

```
<section>
  <article>
    <h2>Título</h2>
    <p> Lorem ipsum dolor sit amet,
    consectetur adipisicing elit.
    Temporibus cum magnam eos a commodi
    sequi possimus iure, eius nemo
    voluptates fuga debitis!
    </p>
  </article>
</section>
```

## CSS

```
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

# INSERTAR CSS EN EL HTML

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" />
```



1

# INSERTAR CSS EN EL HTML

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

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



2

# INSERTAR CSS EN EL HTML

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>
```



3



# CLASS

# 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" />
```

# ATRIBUTO ID

# 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

# COMPARACIÓN

	¿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 Y CASCADA

# 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*.

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

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

# CASCADA

El navegador lee de arriba hacia abajo (forma de cascada) ¿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>
```

# ESPECIFICIDAD

En este gráfico se resume cuán importante es cada selector:



Estilo aplicado  
a la **Etiqueta**.



Estilo aplicado  
a la **Class**.



Estilo aplicado  
al **ID**.



Estilo aplicado  
**Inline**.

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

recapitulemos...

# Selectores - Tipo

- Son los más básicos
- Funciona con los elementos sin ningún atributo especial
- Hay que tratar de usarlos, cuando es posible, porque son más fáciles de manejar

## HTML

```
<p>...</p>
```

## CSS

```
p {...}
```

recapitulemos...

# Selectores - Clase

- Permite ponerle el mismo estilo a una lista de elementos, agregándole el atributo class
- Las clases se escriben en CSS poniendo un punto “.” delante del nombre de la clase
- Está permitido usar la misma clase en múltiples elementos en la misma página

## HTML

```
<p class="clase">...</p>
```

## CSS

```
.clase {...}
```

recapitulemos...

# Selectores - Id

Son parecidos a los selectores de clases, pero son utilizados solo para UN ÚNICO elemento por página

- Usan un atributo id en el elemento HTML
- Se escriben con un # delante del id en el css

## HTML

```
<p id="identif">...</p>
```

## CSS

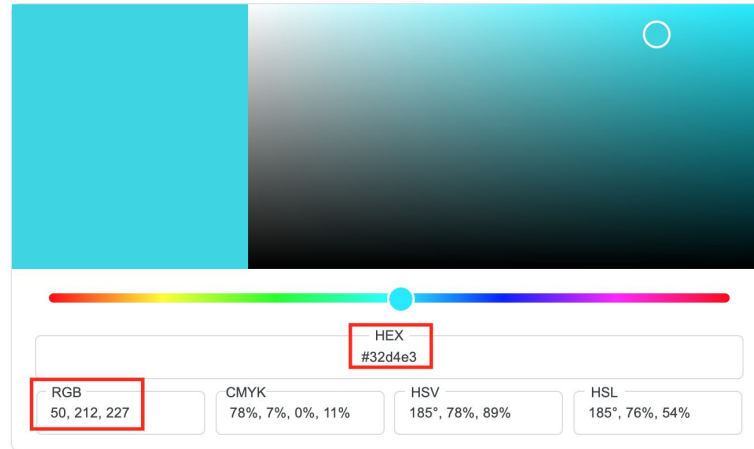
```
#identif {...}
```

Se usan muy poco, para elementos específicos que siempre van a ser únicos

# PRIMERAS PROPIEDADES BÁSICAS

# PROPIEDAD: COLOR

Desde Google, puedes buscar “[color picker](#)” (alternativa [w3schools](#)).





Desde Visual Studio Code, simplemente te “paras” sobre el color. Por ejemplo, escribe “red” y haz la prueba:



# 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.

# AGREGANDO CSS A NUESTRO HTML

Comienza a utilizar CSS en tu proyecto.

## AGREGANDO CSS A NUESTRO HTML

>> **Consigna:** crea un archivo CSS, y vincula al HTML con que estás trabajando. Asigna color a títulos, párrafos y listas, mediante clases y etiquetas, utilizando los conceptos explicados en clase.

# SELECCIÓN DE HTML MEDIANTE CSS

3 formas

Por etiqueta



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

Por clase  
(anteponiendo el “.”)



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

Por ID  
(anteponiendo el “#”)



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

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.) Así se ve en la sección <head> de tu documento HTML:*

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

# 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 las palabras compuestas:

conBorde

productosMasVendidos

Otra convención muy usada recomienda usar guiones:

con-borde productos-mas-vendidos



# PROPIEDADES CSS



# RESETEO CSS

Los reset CSS contienen en su código fuente definiciones para propiedades que los diseñadores necesitan unificar.

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. Para subsanar esa diferencia se suele declarar la siguiente línea:

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

# RESETEO CSS

Esa línea con el selector universal de CSS representado por un asterisco, indica que todos los elementos carecerán de márgenes predeterminados.

Los diseñadores se verán obligados a declarar los márgenes para cada elemento, sin dejar ese aspecto librado a los valores por defecto de cada navegador, y minimizando las diferencias visuales entre los mismos.

# ESTILO PARA LISTA

# 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.

# ESTILOS PARA TEXTO

# FONT-STYLE

## CSS

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

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

Se ve así

Texto normal

*Texto en italica*

# FONT-WEIGHT

## CSS

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

Se ve así

**Texto en negrita**

Texto normal

# FONT-SIZE

## CSS

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

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

## Se ve así

Texto en 20 px

Texto en 200%

Valores posibles: medida en píxeles | porcentaje | rem (1 rem = 16px)



# FONT-FAMILY

## CSS

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

## Se ve así

**Familia Impact**

Familia Comic

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.

**Nota:** te recomendamos visitar el sitio <https://www.cssfontstack.com/>, para conocer más acerca de cómo funciona cada fuente, en los distintos sistemas operativos.

# FONT-FAMILY



Las **tipografías Serif** son aquellas que llevan remates, detalles adicionales en los bordes de las letras. El ejemplo por excelencia de Serif es la **Times New Roman**. Son muy usadas en los periódicos impresos, puesto que los detalles de las letras ayudan a seguir la lectura.

Las **tipografías Sans Serif** carecen de estos detalles y también son denominadas de palo seco. Algunas de las más conocidas son la **Arial** o la **Calibri**. Se utilizan mucho en entornos digitales, puesto que los detalles son difíciles de plasmar en píxeles.

**Nota:** este texto está escrito en Arial que es una tipografía Sans Serif.

# TEXT-ALIGN

## CSS

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

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

Se ve así

texto

texto

Valores posibles: left | right | center | justify

# LINE-HEIGHT

## CSS

```
.interlineado {  
  line-height: 1.6;  
}
```

## Se ve así

texto  
ejemplo

Valores posibles: none | <número> | <longitud> | <porcentaje>

# TEXT-DECORATION

## CSS

```
.subrayado {  
  text-decoration: none;  
}
```

```
.tachado {  
  text-decoration: line-through;  
}
```

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

## Se ve así

Enlace

Parrafo

# ESTILOS PARA FONDOS

# BACKGROUND-COLOR

CSS

Se ve así

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

Parrafo fondo amarillo

Valores posibles: [color]



# BACKGROUND-IMAGE

## CSS

```
.catsandstars {  
  background-image:  
  url("https://mdn.mozillademos.org/files/1199  
1/startransparent.gif"),  
  url("https://mdn.mozillademos.org/files/7693  
/catfront.png");  
}
```

Valores posibles: url | none

## Se ve así

Párrafos con gatos  
y estrellas.



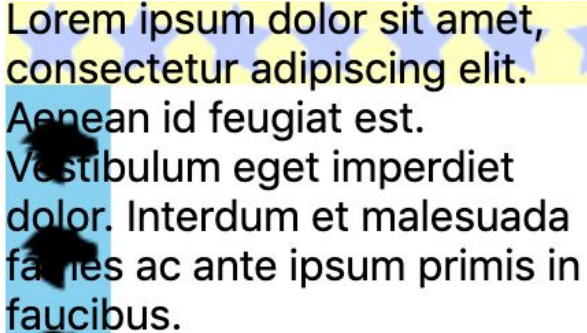
Párrafo sin fondo.

# BACKGROUND-REPEAT

## CSS

```
.ejemplo {  
  background-image:  
url(https://mdn.mozillademos.org/files/12005/starsolid.gif),  
url(https://developer.cdn.mozilla.net/media/redesign/img/  
favicon32.png);  
  background-repeat: repeat-x,  
                    repeat-y;  
}
```

## Se ve así



Lorem ipsum dolor sit amet,  
consectetur adipiscing elit.  
Aenean id feugiat est.  
Vestibulum eget imperdiet  
dolor. Interdum et malesuada  
fames ac ante ipsum primis in  
faucibus.

Valores posibles: repeat | repeat-x | repeat-y | no-repeat | space | round ([ver ejemplos](#))

# BACKGROUND-POSITION

## CSS

```
.ejemplo {  
  background-image:  
url("https://mdn.mozillademos.org/files/12005/s  
tarsolid.gif");  
  background-repeat: no-repeat;  
  background-position: right center;  
}
```

## Se ve así

Lorem ipsum dolor sit amet,  
consectetur adipiscing elit.  
Aenean id feugiat est.  
Vestibulum eget imperdiet  
dolor. Interdum et malesuada  
fames ac ante ipsum primis in  
faucibus.



Valores posibles: posicionX posicionY ([ver ejemplos](#))

# BACKGROUND-SIZE

## CSS

```
.ejemplo {  
  background-image:  
    url("https://mdn.mozillademos.org/files/12  
005/starsolid.gif");  
  background-repeat: no-repeat;  
  background-size: cover;  
}
```

Valores posibles: [ancho] | [alto] | cover | contain ([ver ejemplos](#))

## Se ve así

Lorem ipsum dolor sit amet,  
consectetur adipiscing elit.  
Aenean id feugiat est.  
Vestibulum eget imperdiet  
dolor. Interdum et malesuada  
fames ac ante ipsum primis in  
faucibus.

# UNIDADES DE MEDIDAS

# 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 adaptables (responsive).

# UNIDADES DE MEDIDAS

El problema con los píxeles.

The image shows a screenshot of the Santander website. The main header features the Santander logo and a navigation bar with links: do, Tarjetas, Cuentas, Seguros, AAdvantage, and SuperClub. Below the navigation bar is a large banner with the text "Hola :) ¿en qué te podemos". A configuration menu is open on the right side of the page, titled "Configuración". The menu has a search icon and a "Diseño" section. Under "Diseño", there are three settings: "Temas" (with a link to "Abrir Chrome Web Store"), "Mostrar el botón de la Página principal" (set to "Inhabilitada"), and "Mostrar barra de favoritos". The "Mostrar barra de favoritos" setting is highlighted with a red box, and its value is "Muy pequeño". Below this, there is a "Personalizar fuentes" option.

	A	B	C	D	E	F	G	H	I
1	07	45	77	85	23	23	24	23	24
2	94	78	83	68	75	75	44	75	44
3	75	09	93	16	16	16	51	16	51
4	02	10	05	00	00	00	00	00	00

# UNIDADES DE MEDIDAS

¿Encuentras las diferencias? Exacto, no cambió nada

The screenshot shows the Santander website interface. The header includes the Santander logo and navigation links: Tarjetas, Cuentas, Seguros, AAdvantage, and SuperC. The main content area displays a greeting 'Hola :) ¿en qué te podemos' and a table of numbers. A Chrome DevTools configuration menu is open on the right, showing the 'Diseño' (Design) tab. The 'Tamaño de fuente' (Font size) option is highlighted with a red box and set to 'Muy grande' (Very large).

A	B	C	D	E	F	G	H	I
07	45	77	85	23	23	24	23	24
94	78	83	68	75	75	44	75	44
75	09	93	16	16	16	51	16	51
63	40	65	39	39	39	92	39	92
20	75	13	38	38	38	58	38	58
85	54	38	45	45	45	76	45	76



# UNIDADES DE MEDIDAS

Ejemplo que tiene la solución:

## ELECCIONES EN URUGUAY

### Las encuestadoras dan una leve ventaja

Luis Lacalle Pou (Partido Nacional) es el favorito, mientras que el resto de los candidatos dar la sorpresa



Temas

[Abrir Chrome Web Store](#)

Mostrar el botón de la Página principal

Inhabilitada

Mostrar barra de favoritos

Tamaño de fuente

Muy pequeño

Personalizar fuentes

# UNIDADES DE MEDIDAS

¿Ahora sí está diferente, no?

## ELECCIONES EN URUGUAY

### Las encuestadoras da Pou

Luis Lacalle Pou (Partido Nacional) es el favorito. Amplio, espera dar la sorpresa



#### Diseño

Temas

Abrir Chrome Web Store

Mostrar el botón de la Página principal

Inhabilitada

Mostrar barra de favoritos

Tamaño de fuente

Muy grande

# 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 */  
}
```

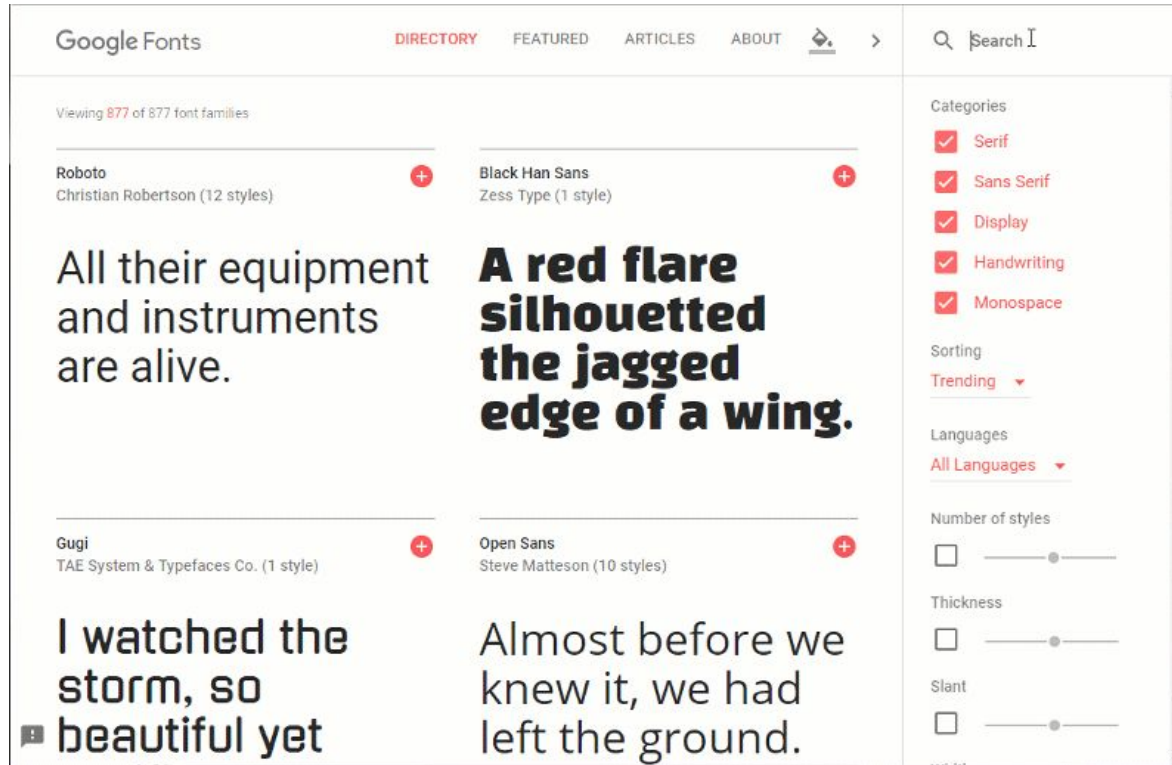
Texto simulando  
20px

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

# TIPOGRAFÍA

# 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;  
}
```

Ver [Google Fonts](#)

## HTML

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

# ¡BONUS!

EMBED

CUSTOMIZE

Load Time: Moderate

## Roboto

☒ thin 100

☐ *thin 100 Italic*

☒ light 300

☐ *light 300 Italic*

☒ regular 400

☐ *regular 400 Italic*

☒ medium 500

☐ *medium 500 Italic*

☒ bold 700



# ¡BONUS!

## CSS

```
h1{  
  font-family: 'Roboto', sans-serif;  
  font-weight: 100;  
}
```

## HTML

```
<link  
href="https://fonts.googleapis.com/css?famil  
y=Roboto:100,300,400,500,700&display=sw  
ap" rel="stylesheet">
```

Ver [Google Fonts](https://fonts.google.com/)

# ¡BONUS!

Se ve así porque *100* es lo más delgado posible:

Titulo

## ASIGNANDO ESTILOS

### >> Consigna:

Incluir algunas de las propiedades vistas:

- **Texto:** font-style, font-weight, font-size, font-family, text-align, line-height y text-decoration.
- **Listas:** list-style-type, list-style-position y/o list-style.
- **Fondos:** background-color, background-image, background-repeat, background-position y/o background-size.
- Tipografía de [Google Fonts](#).

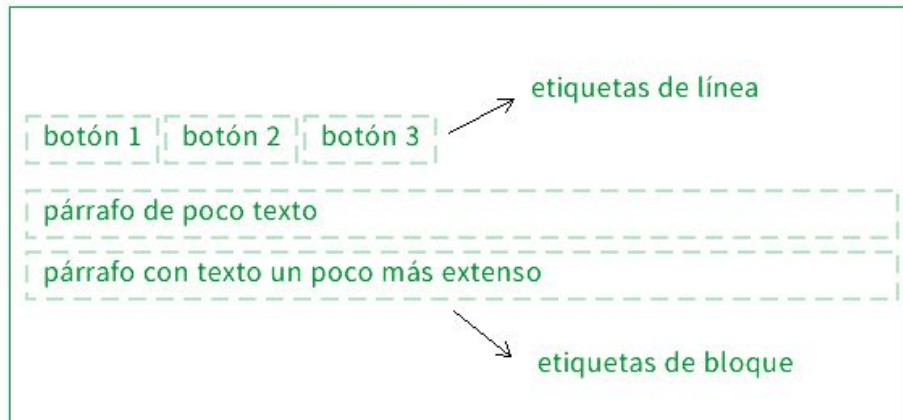
# BOX MODELING

# BOX MODEL

Ese concepto de que “todo es una caja”, da lugar a algo llamado **box model**. Sin importar si son de línea o de bloque, todas las etiquetas tienen propiedades en común.

# PROPIEDADES DE LA CAJA

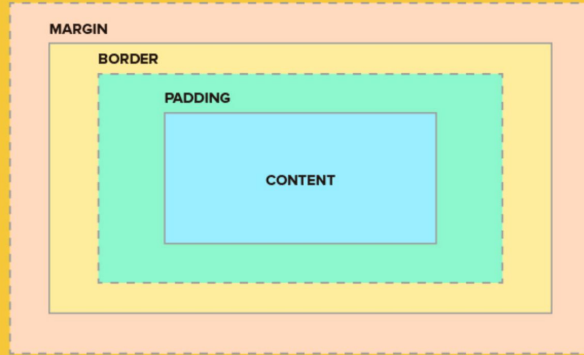
Todos los elementos del HTML son cajas. Un `<strong>`, un `<h2>` y demás, son rectangulares:



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

# PROPIEDADES EN COMÚN

## Box Model Css



**CONTENT:** el espacio para el texto o imagen.

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

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

**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 el ancho de la caja de los elementos.

## *Alto*

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

**width** y **height** no admiten valores negativos, y aquellos en porcentaje se calculan a partir de la altura de su elemento padre.



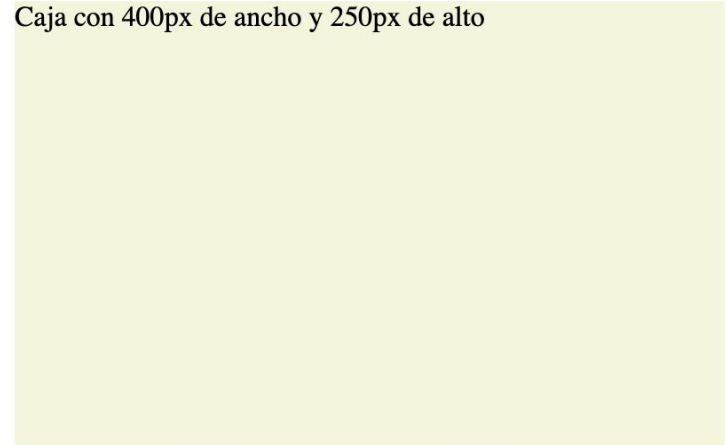
# ALTO Y ANCHO

## 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.](#)

# ALGO MÁS PARA ACLARAR

Cuando un elemento tiene un alto o ancho fijos, cualquier contenido que exceda la caja, desbordará. 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 IS AWESOME

## CSS

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

# 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.

# SOLUCIÓN

## HTML

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

CSS  
IS  
AWESOME

## CSS

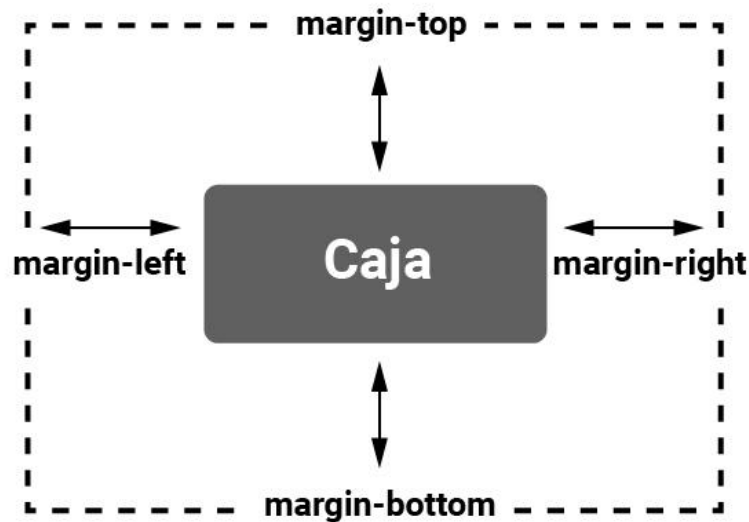
```
div {  
  /* propiedades decorativas */  
  border: solid 1px black;  
  padding: 5px;  
  display: inline-block;  
  font-size: 32px;  
  font-family: Arial;  
  /* propiedades que hacen el "problema" */  
  width: 100px;  
  height: 110px;  
  /* solucion */  
  overflow: hidden;  
}
```

# ESPACIO EXTERIOR

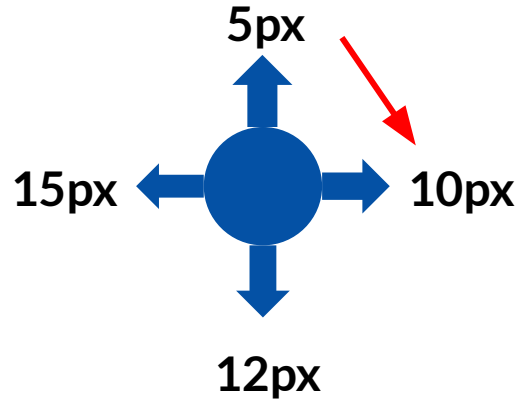
## *Margin (márgenes)*

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;  
}
```

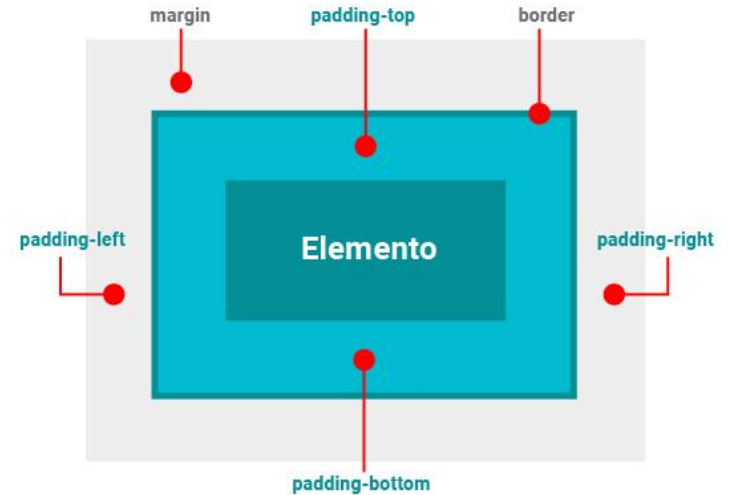
[Más información sobre Margin](#) | Nota: se pueden resumir los 4 lados poniendo solo “margin: valor”

# ESPACIO INTERIOR

Padding (relleno)

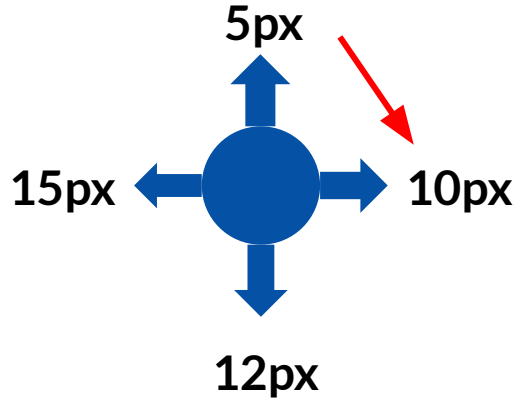
Las propiedades *padding-top*, *padding-right*, *padding-bottom* y *padding-left*, se utilizan para definir los espacios internos de cada uno de los lados del elemento, por separado.

Puedes definir los 4 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

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 4 lados (forma abreviada “*border*”) o sólo aquellos que necesites.

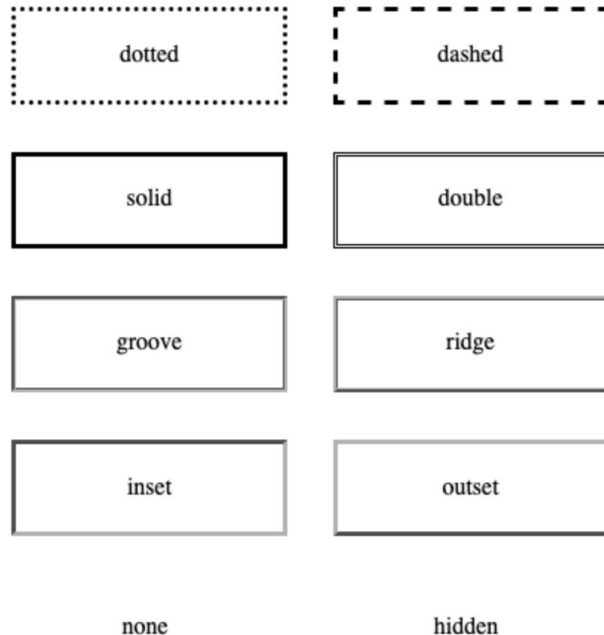


# BORDES

## Nota

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

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



# 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



Valores comunes: estilo grosor color| [Ejemplos y más información](#)

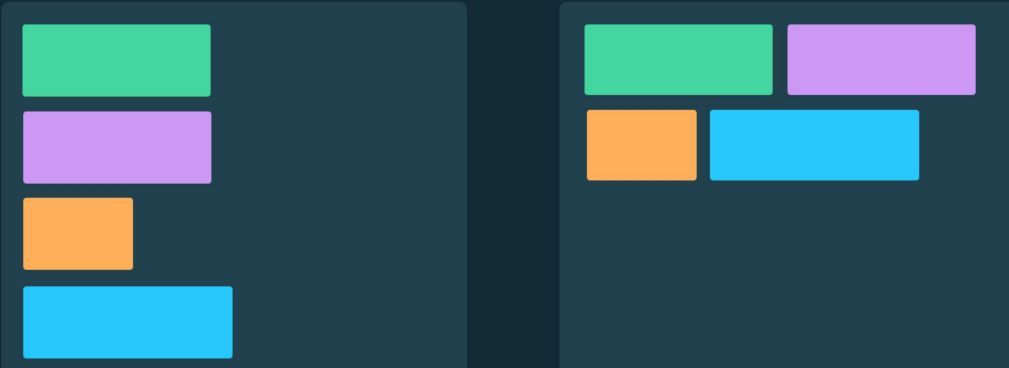
# DISPLAY

# 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

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 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 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;  
}
```

# MENÚ CON DISPLAY

## HTML

```
<ul>
  <li><a href="#home"
    class="activo">
    Home</a></li>
  <li><a href="#nosotros">
    Nosotros</a></li>
  <li><a href="#contacto">
    Contacto</a></li>
</ul>
```

## CSS

```
ul {
  list-style-type: none;
  overflow: hidden;
  background-color: #333;}

li {
  float: left; }

li a {
  display: inline-block;

.activo {
  background-color: blue;}
```

# BOX MODEL

Incluye box model en tu proyecto.

# BOX MODEL

>> **Consigna:** agrega al CSS de tu Proyecto.

- Márgenes.
- Rellenos.
- Bordes.
- Menú.



- Imágenes de relleno | ***placekitten***
- Imágenes de relleno | ***placedog.net***
- The CSS Box Model | ***CSS-TRICKS***

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



**UTN.BA**  
UNIVERSIDAD TECNOLÓGICA NACIONAL  
FACULTAD REGIONAL BUENOS AIRES