



CSS Básico 1.

¿Qué es CSS y para qué sirve?

¿Por qué CSS3?

¿Cómo funciona CSS?

Las 3 maneras de escribir código CSS.

Selectores en CSS.

Cascada, Especificidad y Herencia.

Box-Model en CSS.

▼ ¿Qué es CSS y para qué sirve?

CSS (Cascading Style Sheets por sus siglas en inglés) es un lenguaje de estilos, que es **usado para personalizar visualmente a nuestro HTML**. Definiendo que el HTML será toda la información de nuestra página, textos, encabezados, enlaces, listas, imágenes, eso es el HTML.

El CSS permite modificar los estilos de nuestro HTML, ya sea colores, fondos, tamaños de texto, forma de las imágenes, tamaños de elementos, posicionamiento, animaciones, y todos los estilos de nuestro HTML.

Página de Netflix con CSS.



Página de Netflix sin CSS.



CSS es un lenguaje de estilos en cascada, que nos ayudará a darle estilos y cambiar la apariencia visual de nuestro HTML.

▼ ¿Por qué CSS3?

CSS fue lanzado en 1996, por lo que hasta el día de hoy que es 2023 han lanzado muchísimos módulos y mejoras a CSS pasando de CSS hasta CSS3, actualmente estamos en la versión 3, llega un punto donde ya no habrá un CSS4, sino que ahora se decidió trabajar por módulos.

De cualquier manera no te preocupes, en este reto vamos a ver CSS3 con sus módulos más recientes.

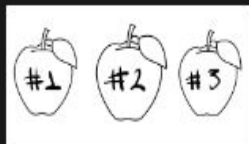
▼ ¿Cómo funciona CSS?

Imaginemos que tenemos tres manzanas y queremos cambiar la apariencia de una de esas manzanas, la forma en la que sería con CSS sería **primero seleccionar la manzana a modificar**, después **definir que estilo queremos darle**, en mi caso quiero darle color y por último decidir **que valor tendrá el estilo de la manzana** yo quiero un color rojo.

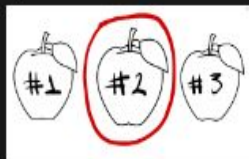
Bajo esa analogía funciona CSS, para mí CSS tiene 3 pilares en los que se basa casi todo el lenguaje. (Selectores, Propiedades y Valores)

A continuación te lo explico con manzanas:

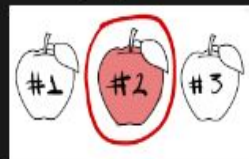
Tenemos 3 manzanas, de las cuáles queremos seleccionar una manzana de todas ellas y darle un color.



Queremos seleccionar a la manzana número 2 (Esto sería un selector, que define a que elemento queremos darle estilos)



Ahora queremos darle un color que sea rojo. (El color sería la propiedad y el valor rojo, define que valor tiene la propiedad.



Selectores

Los selectores definen a que elemento o elementos vamos a modificar.

Propiedades.

La propiedad define que estilo queremos cambiar o definir del elemento a seleccionar, por ejemplo el color, el tamaño, el fondo o cualquier otro estilo permitido en CSS.

(Un selector puede tener muchas propiedades a cambiar o definir)

Valores.

Los valores van ligados directamente con la propiedad, no puede haber uno sin el otro, y el valor determina el valor de el estilo a cambiar, por ejemplo si queremos definir un color, el valor sería rojo, azul, verde, si queremos cambiar un tamaño el valor sería el tamaño a cambiar.

Cada propiedad tiene un valor permitido, si queremos cambiar el ancho de un elemento no podemos darle el valor rojo.

Primer código CSS.

- En CSS tenemos distintos tipos de selectores, pero empecemos con el más común, que es poner el nombre tal cuál de la etiqueta que queremos modificar.

- Después abriremos y cerraremos llaves, esto indicará que dentro de estas llaves, irán todos los estilos que queremos modificar del selector.
- Pondremos una propiedad a definir, y un valor para la propiedad, después de cada pareja de valor y propiedad, pondremos un punto y coma, esto define que podemos escribir otra regla CSS debajo.

```
h2{  
  color: red;  
  border: 1px solid #000;  
}
```

Los selectores cambiarán todos los elementos que coincidan con los mismos, en este ejemplo modificará a todos los elementos que tengan una etiqueta `<h2>`.

🧠 Momento de pasar al código CSS. (Manos a la obra)

▼ Las 3 maneras de escribir código CSS.

Hay 3 formas de agregar CSS a nuestro documento HTML, a continuación veremos las 3, pero solamente una es la recomendada.

Estilos en línea.

Directamente definimos los estilos en la etiqueta que queremos modificar, en este ejemplo le dimos un color rojo, y un tamaño de fuente de 40px.

```
<h2 style="color:red; font-size: 40px;">¡Bienvenido al RETO!</h2>
```

En este ejemplo no hay selector, porque definimos los estilos en la etiqueta a modificar, aunque sí hay propiedades y valores, separadas por un punto y coma.

El inconveniente con esta manera es que si yo quiero darle los estilos a otra etiqueta `<h2>` tendría que agregar los estilos en la otra etiqueta `<h2>` (Además genera un problema de especificidad, se verá más adelante)

Etiqueta Style.

En HTML hay una etiqueta llamada `<style>` en la cuál nosotros podemos incrustar código CSS directamente en nuestro HTML, ahora si tenemos la estructura completa de CSS, teniendo un selector, propiedad y valor.

En este ejemplo, a todas las etiquetas `<h2>` se les aplicarán los estilos dentro del selector.

```

CSS
Copy

<style>

  h2{
    color: red;
    font-size: 40px;
  }

</style>

<h2>¡Bienvenido al RETO!</h2>
<h2>¡Empezamos!</h2>
```

El único inconveniente de esta manera, es que si tenemos otro documento HTML y queremos darle los mismos estilos, tendríamos que copiar la etiqueta y sus valores, para después pegarlos en el otro documento donde queremos agregar los estilos.

Creando un archivo CSS.

La última manera y más recomendada es crear un archivo CSS, para eso lo vas a crear con el nombre que desees, yo te recomiendo el nombre `style`, posteriormente un punto y la extensión `css`, de tal manera que tu archivo quedaría así: `style.css`.

Para poder referenciar tu archivo CSS, irás a la etiqueta `<head>` de tu documento HTML, y ahí usarás la etiqueta `<link>` indicando la ruta donde se encuentra tu archivo CSS. Se vería de esta manera:


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

Ahora puedes pegar éste código, dentro de tu archivo CSS.

```
h2{  
  color: red;  
  font-size: 40px;  
}
```

¡Y listo! Has creado tu primer archivo CSS y tus primeras líneas de código.

▼ Selectores en CSS.

Para poder cambiarle estilos a los elementos en CSS, necesitamos seleccionarlos. Para eso son los selectores.

Los selectores es la primera parte de una regla CSS, definiendo que elementos del HTML serán modificados, todos los elementos que coincidan con el selector, se verán afectados por los estilos del mismo. (Si el selector no encuentra ninguna coincidencia entonces no aplicará ningún estilo).



Todos los elementos que coincidan con el selector, se verán afectados.

A continuación voy a enumerarte los selectores que debes conocer en CSS:

- 🎯 [Selectores Básicos.](#)
- 🎯 [Selectores Combinadores.](#)

▼ Cascada, Especificidad y Herencia.

Antes de comenzar a darle magia y vida a nuestros proyectos, debemos conocer como funciona CSS, porque para que nuestro conocimiento sea fuerte y eficaz, debemos tener unas bases solidas.

Trabaja con este HTML:

```
<section class="section">
  <h1 class="title" id="title">RETO CSS en 30 días.</h1>
  <p class="paragraph">Este es un reto diseñado para dominar CSS en 30 días desde un
nivel básico a un nivel avanzado, al finalizar el reto serás capaz de implementar CS
S en cualquier proyecto.</p>
</section>

<p class="paragraph"></p>
```

▼ Cascada.

CSS en español significa hojas de estilo en cascada, esto es porque los estilos se manejan en cascada, en pocas palabras la cascada define que lo estilos se aplican de arriba hacia abajo, por lo que si dos o más estilos se repiten para el mismo elemento, entonces predomina el último estilo en ser declarado.

Problema.

Si tenemos 2 selectores que cambien los estilos de un mismo elemento, ¿Qué selector aplicará los estilos a ese elemento?

```
.title{  
  color: crimson;  
}  
  
.title{  
  color: steelblue;  
}  
  
.title{  
  color: cornflowerblue;  
}
```

Respuesta:

Se aplicará el último selector en ser definido, esto es por la cascada, que siempre define el último estilo en ser declarado. Por ende se aplicará el color `cornflowerblue`.

Respuesta:

Se aplicará el último selector en ser definido, esto es por la cascada, que siempre define el último estilo en ser declarado. Por ende se aplicará el color `cornflowerblue`.

Para resumir, la cascada es la manera en la que CSS determina que estilos se aplicarán cuándo hay más de un estilo que modifica al mismo elemento, en ese caso definirá a los últimos estilos en ser declarados.

▼ Especificidad.

Si bien la cascada es una de las maneras de definir que selectores con sus propiedades se aplican, la más importante en CSS es la especificidad.

La especificidad le da a cada selector un valor, de esta manera el selector que predomina es el que tiene un valor mayor a todos los demás que modifican al mismo elemento y estilos.

Problema.

Tenemos 3 selectores distintos, los 3 selectores modifican al mismo elemento, ¿Qué selector tomará CSS para aplicar los estilos?

```

.title{
  color: crimson;
}

#title{
  color: steelblue;
}

h1{
  color: cornflowerblue;
}

```

Solución:

Tomará los estilos de el selector de ID, porque es el selector que tiene más valor, y la especificidad toma el selector con mayor valor.

TABLA DE ESPECIFICIDAD.

Especificidad 100	Especificidad 10	Especificidad 1	Especificidad 0
Selector de ID	Selector de clase	Selector de tipo/etiqueta	Selector Universal
	Selector de atributos	Pseudoelementos	
	Pseudoclases		

- 🧠 En VSCODE podemos visualizar la especificidad de cada selector.

Consideraciones:

- Cuándo usas más de un selector para una regla, entonces se suma su especificidad (Ten cuidado con esto, incluso en responsive).
- Cuándo dos selectores con la misma especificidad modifican a un mismo elemento y propiedad, entonces en ese caso se aplica la cascada.
- Los estilos en línea y el valor `!important` rompen la especificidad, aunque `!important` tiene más valor que los elementos en línea.

La razón del selector Universal.

El selector universal tiene especificidad 0, **porque esta pensado para cambiar los estilos por defecto del navegador**, cuándo cambiemos los estilos de cualquier elemento, aunque sea un selector con un valor de especificidad de 1, le ganará al selector con especificidad de 0.

Si al principio definimos que todos los elementos tengan un `margin: 0;` con el selector universal, no importa que se encuentre en la última parte del código CSS, cualquier selector podrá cambiar los estilos del selector universal.

Consideraciones:

- A los elementos que no les cambiemos los estilos con otro selector, se quedarán con los estilos del selector universal.

▼ Herencia.

En CSS los elementos padres heredan algunos estilos a sus hijos por defecto, **no todos los estilos se heredan por defecto**. En pocas palabras la herencia es la capacidad de los padres de heredar estilos a sus hijos (Si yo defino a sus hijos con otros estilos, la herencia se anula)

Ejemplos de Herencia en CSS:

- Un ejemplo es cuando a `body` le damos un font-family, casi todos sus hijos heredarán esa fuente.
- Otro ejemplo es cuando a un contenedor le definimos un color, sus hijos también toman el color.

Consideraciones:

- Si un elemento hereda un color de su padre, pero después a ese elemento le doy un color distinto, entonces la herencia se anula.
- No todas las propiedades son heredables.
- Podemos obligar a una propiedad a heredar con el valor `inherit` que nos permite heredar el valor del padre, en la propiedad que estemos utilizando.

Ejercicios para comprender la herencia:

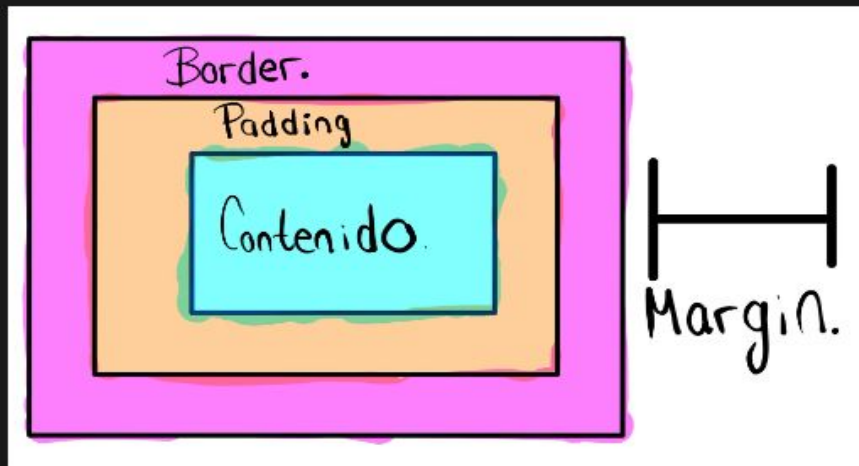
- Darle a la clase section, un color `brown` y un `text-align:center;`
- Darle a body una fuente `cursive`.
- Hereda un borde de section para el elemento `.title`.

▼ Box-Model en CSS.


Usa este HTML:


```
<div class="caja">  
  Soy un texto dentro de una caja.  
  
    
</div>
```


Para CSS todos los elementos son considerados cajas, y estas tienen propiedades que podremos manipular para modificar el tamaño, posicionamiento y darle un diseño más personalizado a nuestro HTML. (Esto se conoce como Box-model o modelo de cajas, y se divide en capas, como en Shrek)




Capas del Box-Model:


 **Capa del contenido.**

 **Capa del padding.**

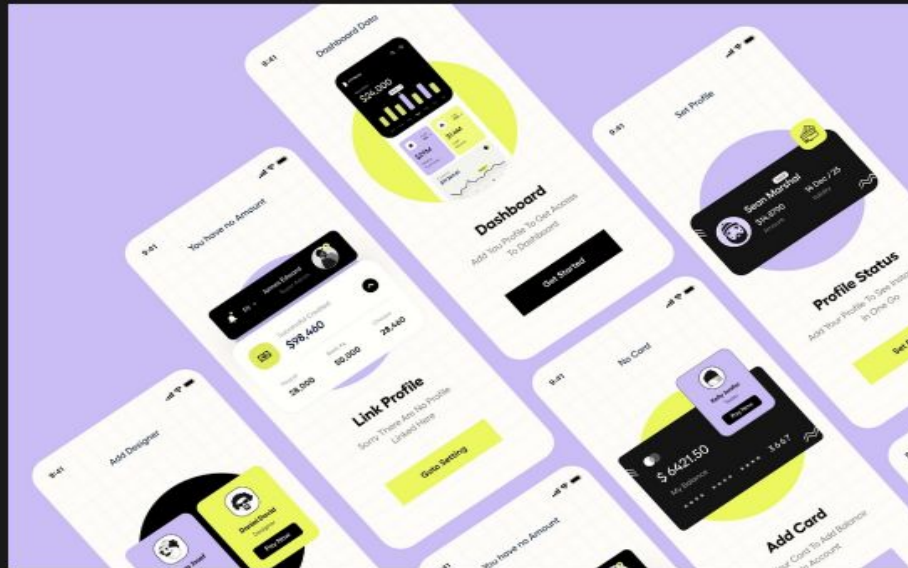
 **Capa del borde.**

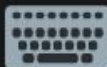
 **Capa del margen.**

 **Box-sizing.**

 **Tipos de elementos.**

Ejercicio Final:





Capa del contenido.

La primera capa definida por el contenido del elemento, texto o imágenes.

Aunque también tenemos propiedades para cambiar las dimensiones del elemento, que por defecto en el Box-model afectan la capa del contenido.

Width

Da una anchura al área de contenido en CSS. Por defecto el width es automático y se da por el contenido de nuestro elemento.

```
width: 500px;  
width: 400px;
```

Max-width

Define un ancho máximo a un elemento, esto nos puede ayudar cuando estamos escribiendo en un elemento pero queremos que no mida más de cierto limite. **(No importa que en el width demos un valor mas grande que en el max-width, el max-width dará el tamaño máximo)**

Max-width

Define un ancho máximo a un elemento, esto nos puede ayudar cuando estamos escribiendo en un elemento pero queremos que no mida más de cierto limite. **(No importa que en el width demos un valor mas grande que en el max-width, el max-width dará el tamaño máximo)**

```
max-width: 500px;  
width: 600px;
```

Min-width

Define un ancho mínimo para el elemento, esto hará que el elemento no mida menos que ese ancho, aunque no tenga contenido dentro. **(No importa que en el width demos un valor mas pequeño que en el min-width, el min-width dará el tamaño mínimo)**

```
min-width: 300px;  
width: 200px;
```

Height

Nos permite dar altura al área de contenido en CSS. Por defecto el height es automático y se da por el contenido de nuestro elemento.

CSS

 Copy

```
width: 500px;  
height: 500px;
```

Problema Overflow.

Cuándo tenemos un width y height fijos, y más contenido del que pueden abarcar estas 2 propiedades, entonces se desborda. Para eso tenemos la propiedad `overflow`.

```
overflow: hidden | visible;
```


Min-height


Define una altura mínima, esto hará que el elemento tenga una altura mínima y que no pueda tener una altura inferior a ella. **(No importa que en el height demos un valor mas pequeño que en el min-height, el min-height dará el tamaño mínimo)**

```
min-height: 500px;  
height: 400px;
```

Max-height

Define una altura máxima a un elemento. **(No importa que en el height demos un valor mas grande que en el max-height, el max-height dará el tamaño máximo)**

```
max-width: 500px;  
width: 600px;
```



Volver al menu



Capa del padding.

El padding o capa de relleno, es la segunda capa en CSS se encuentra entre el borde del elemento y su contenido, nos ayuda a crear un espacio entre el borde del elemento y su contenido, de esta manera dando un diseño más armonioso. **(Agregar padding puede incrementar el alto o ancho de nuestro elemento, ya que es una capa que tiene dimensiones)**

Hay 4 propiedades para modificar al Padding, cada propiedad esta hecha para cada dirección del elemento, arriba, abajo, izquierda y derecha. Vamos a comenzar en el orden de las manecillas del reloj.

padding-top

Define un relleno en la parte de arriba o superior de nuestro elemento.

```
padding-top: 20px;
```

padding-right

Define un relleno en la parte derecha de nuestro elemento.

```
padding-right: 40px;
```

padding-bottom

Define un relleno en la parte de abajo o inferior de nuestro elemento.

```
padding-bottom: 60px;
```

padding-left

Define un relleno en la parte de izquierda de nuestro elemento.

```
padding-left: 80px;
```

Padding


En CSS hay propiedades que abrevian a muchas propiedades, a esto se le conoce como un shorthand.

En la propiedad padding podemos abreviar a las 4 subpropiedades juntas, veamos como tener el mismo resultado. La propiedad padding puede recibir máximo 4 valor, mínimo 1.

CSS

 Copy

```
/* Usando 4 valores, el orden es en las manecillas del reloj */  
padding: top right bottom left;  
  
/* Usando 3 valores, el primer valor se aplica a arriba,  
el segundo se aplica a la izquierda y derecha y el tercero se aplica a abajo */  
padding: top right-left bottom;  
  
/* Usando 2 valores, el primer valor se aplica a arriba-abajo, y el segundo se aplica  
a izquierda-derecha */  
padding: top-bottom right-left;  
  
/* Usando 1 valor, el valor se aplica a todos los lados de igual manera */  
padding: top-right-bottom-left;
```

 Volver al menu



Capa del borde.

El borde la última capa que modifica el tamaño del elemento, define la línea del borde del elemento agregando diferentes estilos para el borde.

El borde en CSS tiene 3 características:


- Color
- Ancho
- Estilo


Para que un borde funcione debemos declarar al menos el estilo del borde (el más usado es


`solid`).

Propiedades borde

```
border-color: purple;  
/*Define el color del borde*/  
  
border-width: 20px;  
/*Define el color del borde*/  
  
border-style: solid;  
/*Define el estilo del borde*/
```

 [Border-width](#)


 [Border-style](#)

 [Border-color](#)

Border

La propiedad `border` es el shorthand de las propiedades, width, style y color, no importa el orden en que pongamos los bordes, pero yo prefiero poner el width primero. (Si quieres modificar más de un lado, se deben usar las propiedades `border-width`, `border-style` y `border-color`)

```
border:red solid 30px ;  
border: solid 30px red ;  
border: 30px solid red ;
```

 Volver al menu



Capa del margen.

La capa del margen establece un margen en las 4 direcciones del elemento, es la única capa que no afecta el tamaño del elemento porque se encarga de crear márgenes entre elementos.

```
margin: 50px;
```

`margin` es un shorthand que abrevia a 4 propiedades que se encargan de definir los márgenes por cada dirección del elemento, (usa la misma lógica que en el padding sólo que en vez de agregar relleno, esta agrega márgenes).

Trabajando con las propiedades de margin individualmente.

```
margin-top: 50px;  
margin-right: 30px;  
margin-bottom: 70px;  
margin-left: 10px;
```

Shorthand Margin.

```
margin: 50px 30px 70px 10px;  
/* Esto es lo mismo que el ejemplo anterior, recordando que el primer valor es para top,  
el segundo para right, el tercero bottom y el cuarto left*/  
  
margin: 50px 30px 70px;  
/* Esto agregará un margin arriba de 50px, uno hacia los lados de 30px y uno hacia  
abajo de 70px*/  
  
margin: 50px 30px;  
/* Esto agregará un margin arriba y abajo de 50px y uno hacia los lados de 30px*/  
  
margin: 50px;  
/* Esto agregará margin de 50px hacia todos los lados*/
```



Márgenes automáticos



Colapso de márgenes



Márgenes automáticos

En CSS podemos usar el valor, `auto`. El valor `auto` solamente funciona de manera horizontal hasta el momento (puede cambiar en flexbox).

`margin-left`

Si a un `margin-left` le damos el valor `auto` esto hará que se vaya a la derecha.

`margin-right`

Si a un `margin-right` le damos el valor `auto` esto hará que se vaya a la izquierda.

Consideraciones:

- Si usamos el valor `margin-right`, y `margin-left`, en `auto`, centrará horizontalmente al elemento.(Podemos resumir esto en `margin: 0 auto;`)
- Sólo se podrá centrar con márgenes si se cumplen 2 condiciones.
 - Que el elemento tenga `display:block`.
 - Que tenga un `width` definido que ocupe menos del 100% del contenedor



Colapso de márgenes

Es un comportamiento que sucede cuando dos márgenes se combinan y se vuelven un solo margen, (solo sucede con los márgenes verticales).

Lo que sucede en el colapso de márgenes es que se fusionan los 2 márgenes que se encuentran, formando uno solo, siempre predomina el tamaño más grande.

Colapso de márgenes de hermanos adyacentes

Sucede cuando dos hermanos adyacentes, fusionan su `margin` verticalmente, en vez de que se fusionen los márgenes, prevalecerá el más grande.

```
<main class="main">
  <h1 class="main__title">Hola mundo</h1>
</main>

<footer class="footer">
  <h2 class="footer_title">Footer</h2>
</footer>
```

```
CSS
*{
  margin: 0;
}

body{
  font-family: Arial;
}

.main{
  background-color: steelblue;
  color: white;
  margin-bottom: 30px;
}

.footer{
  margin-top: 50px;
  background-color: slateblue;
  color: white;
}
```

Como vemos el margen no es 80px, sino 50px que es el margen con mayor tamaño.

Colapso de márgenes de elementos hijo

Este es el caso más alarmante, porque cuando tenemos un elemento hijo que esta en la primera posición y queremos darle un `margin-top` tendremos un problema, y es que el `margin-top` de el hijo se fusionará con el margin-top del elemento padre.

HTML ▾

 Copy

```
<main class="main">
  <h1 class="main__title">Hola mundo</h1>
</main>

<footer class="footer">
  <h2 class="footer_title">Footer</h2>
</footer>
```

```
.main{
  background-color: steelblue;
  color: white;
}

.main__title{
  margin-top: 40px;
}

.footer{
  background-color: slateblue;
  color: white;
}
```

Esto pasa debido a que se mezclan los márgenes porque en elemento padre no hay ninguna capa que intervenga con el margen del elemento hijo, entonces si queremos evitar esto debemos poner una capa en el elemento padre, las capas que podemos usar será el `padding` o `border` mayores a 0, en mi caso pongo un `0.1px padding`.


```
.main{
  background-color: steelblue;
  color: white;
  padding: .1px;
}

.main__title{
  margin-top: 40px;
}

.footer{
  background-color: slateblue;
  color: white;
}
```

Este comportamiento igual pasa si usamos `margin-bottom`, es la misma solución.

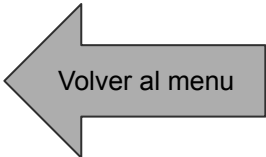
```
.main{
  background-color: steelblue;
  color: white;
}

.main__title{
  margin-top: 40px;
  margin-bottom: 40px;
}

.footer{
  background-color: slateblue;
  color: white;
}
```

Consideraciones:

- Solo afecta a los elementos bloque.



Volver al menu



Box-sizing.

Al momento de agregar las primeras tres capas del `box-model` el tamaño del elemento cambia, esto es debido al `box-sizing` que define el comportamiento de las cajas respecto al tamaño, en lo personal este comportamiento por defecto no me gusta.

El valor por defecto para todos los elementos es `content box`, este valor no toma la propiedad `width` y `height` como tamaño final, sino solo como el tamaño de la capa de contenido, a la cual se le sumarán el `padding` y `border`.

Valor border-box

En este valor la propiedad `width` y `height` son los valores definitivos, donde para calcular el contenido primero se tomará el valor del `padding`, y `border`, el sobrante de este será el tamaño del contenido.

```
.element{  
  box-sizing: border-box;  
}
```



Se acostumbra definir con el selector universal, que todos los elementos tengan un `box-sizing: border-box;`

Volver al menu



Tipos de elementos.

En CSS hay 2 tipos de cajas o elementos por defecto, las cajas en bloque y las cajas en línea.

Cajas en bloque (Display:block)

Podemos saber que una caja es en bloque por las siguientes características por defecto:

- Genera un salto de línea, debido a que toma el 100% de su contenedor.
- Tomará el 100% de su contenedor
- Se pueden usar todas las propiedades del box-model
- Ejemplos de elementos bloque:
 - divs
 - section
 - h1
 - p

https://developer.mozilla.org/es/docs/Web/HTML/Block-level_elements



(Recuerda que solo podemos centrar un elemento con márgenes automáticos si tiene es un elemento bloque y tiene un width definido)

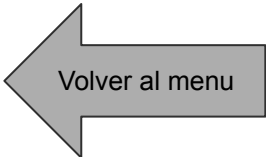
Cajas en línea (Display:inline)

- No hay saltos de línea, se ponen uno a lado de otro elemento, hasta que se acabe el espacio disponible se genera el salto de línea.
- Ocupan solo el espacio de su contenido
- No se aplican la propiedad width y height
- Tenemos problemas al usar margin, padding y border de manera vertical.
- Ejemplos:
 - Etiqueta <a>
 - Etiqueta

https://developer.mozilla.org/es/docs/Web/HTML/Inline_elements

Hibrido Display:inline-block;

Si queremos que los elementos se pongan uno a lado del otro, que ocupen solo el espacio de su contenido, pero poder usar las propiedades del box-model sin errores, entonces podemos usar el valor `display: inline-block`, que permite que un elemento en línea pueda usar las propiedades del box-model sin errores.



Volver al menu