¿Qué es CSS? <https://upgradehub.instructure.com/courses/343/pages/css-introduccion?module_item_id=7858>

En diseño y desarrollo web se conoce como **CSS** al lenguaje utilizado para crear **Hojas de Estilo en Cascada,** que son la parte de nuestra web encargada de dar estilos, color y forma a todos los elementos que nos encontremos.

Si tuviésemos que comparar CSS y HTML, podríamos decir que el HTML es el esqueleto que da forma a todas las webs que crearemos de ahora en adelante, y CSS es todo aquello que hace bonito y da el componente visual a estas.

Propiedades básicas

* Color {color: RED; } ;
* Tamaño de fuente {Font-size: 30px ; } .
* Decoraciones del texto {text-decoration: underline; } .
* Transformar el texto {text-transform: capitalize; } .
* Tipografía: {font-family: Times New Roman; } .
* Color de fondo: {background-color: BLUE;}
* Opacidad del fondo: {opacity-color: 1; } . (1 es máxima opacidad; 0 mínima)

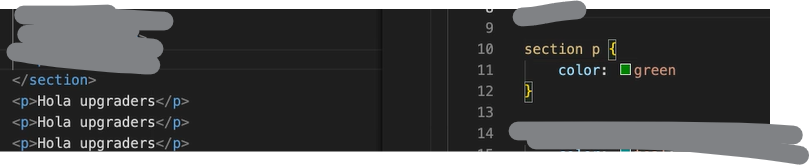
Para poder ligar el CSS a HTML tendría que definir el campo sobre el que se quiere aplicar los estilos y describir los cambios dentro de las llaves { } . Por ejemplo: h1{ color: red; }

2. Aquí dice dónde encontrar y ligar HTML con CSS (p

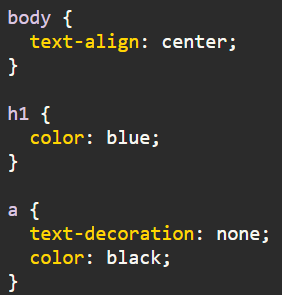
Selectores en CSS

En todo archivo CSS tendremos que definir a qué elementos aplicaremos los atributos de estilos que definimos, y eso lo hacemos mediante tres tipos de selectores:

* Selectores de etiqueta (h1, h2, h3, p, button, ...) (apenas usado)
* Selectores de clases (.titulo-azul) (utilizar SIEMPRE, el más usado)
* Selectores de id (#titulo) (NO USAR NUNCA, para no tener q hacer mucho más código) Funciona igual q clase
* Selectores múltiples (para usarlo en varios elementos separados por coma)
* Selectores por relación (separados por un espacio entre el elemento y la llave) NO USAR, ya que el código lee de izq a dcha “header h2 {}) primero se lo aplicaría a h2 y luego buscaría header h2.

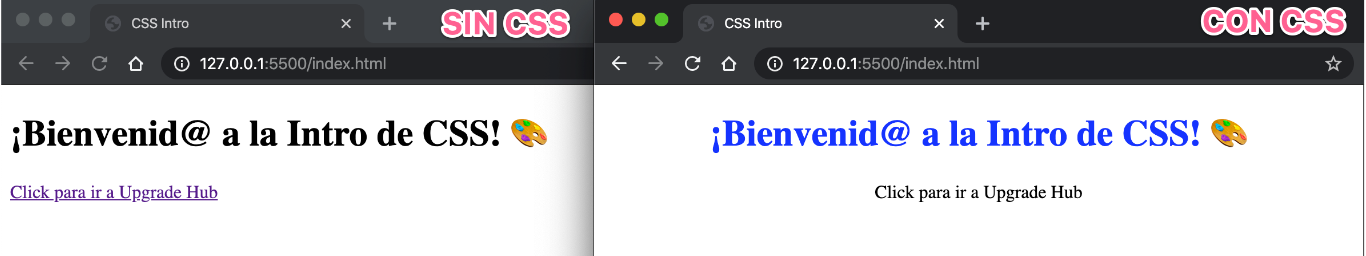


**Selectores de etiqueta Todos los elementos compartirían mismas características**

Este tipo de selectores serán aplicados a todos los elementos HTML que compartan dicha etiqueta.

Con estos estilos hemos definido que el **<body>** de nuestra web tendrá todo el texto centrado, los elementos <**h1>** serán azules, y los links **<a>** serán de color negro y no tendrán el subrayado típico de todos los links por defecto en HTML.

Este tipo de selectores tiene un principal problema, y es que al afectar a todos los elementos del mismo tipo que hay en nuestra web, no podríamos tener varios títulos con colores diferentes en la misma web si son del tipo **<h1>**.



**Selectores de clase Puede personalizarse para cada elemento**

Los selectores de clase son aquellos que se definen usando las clases que utilizamos en nuestros elementos HTML y nos permiten dar propiedades únicas a cada elemento. Para referenciarlos, usaremos un **punto** delante del nombre de la clase (CSS). Los elementos pueden tener tantas clases como se desee. Ejemplo gráfico:

|  |
| --- |
| **HTML** **CSS**  Resultado visible |

### **Selectores de id Predominan y anulan el resto en el archivo entero**

Por último, vamos con los selectores **más importantes,** pero no son importantes porque sí, lo son porque su uso pisa a los estilos de cualquier clase que se haya definido tanto antes como más tarde en nuestro código.

A parte de esto, hay que considerar también que los ids deben ser únicos, y por tanto **nunca repetirse.**

Vamos a editar el HTML y a añadir un id a nuestro último **<h3>** con el atributo **id="title-purple".** Y vamos a crear los estilos para la id con su selector utilizando el prefijo **#** para indicar que es una id:

|  |
| --- |
| **HTML** **CSS**  Resultado visible |

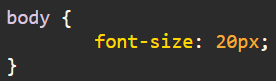
La especificidad es la importancia que tiene al aplicar un estilo en el navegador, y se “recuentan” o se sabe cuál es más importante en base a puntos (suma puntos en función del tipo de elemento q sea)

Estilos aplicados a textos

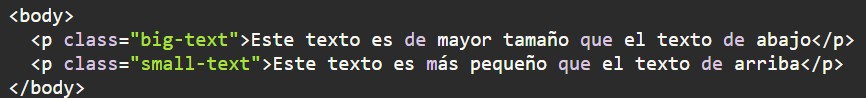
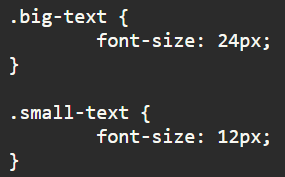
En esta sección vamos a ver los estilos que podemos aplicar a los textos de nuestra web usando CSS y los atributos siguientes:

* **font-size**: Cambia el tamaño de la fuente
* **font-weight:** Cambia el peso de la fuente
* **font-family:** Cambia la familia tipográfica del texto
* **color:** Cambia el color del texto
* **text-decoration:** Añade subrayado o tachado al texto
* **text-transform:** Transforma el texto a mayúsculas, minúsculas, capitalizado...

**Font-size**

Para cambiar el tamaño de la fuente se puede hacer que aplique a todo un contenedor, o especificar sobre un elemento en concreto.

P.Ej: Se quiere cambiar sobre todo el body, en CSS habría que introducir 🡪

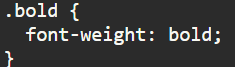
P.Ej: Se quiere cambiar sobre un elemento específico 🡪 **CSS** 🡪

**HTML**

**Font-weight**

Para cambiar el peso de la fuente para que destaque más o menos, aplicaremos el comando “font-weight”, modificando su **valor** entre el **100** (más ligero) y **900** (más pesado), o con valores no numéricos como:

**Bold**, **bolder**, **lighter**, **normal**. 400 es el más usado.

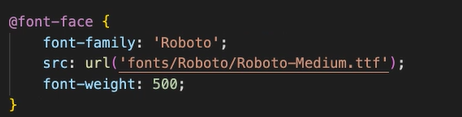
** HTML CSS**

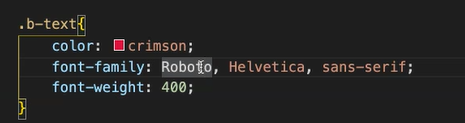


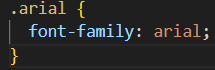
**Font-family** [**https://fonts.google.com/**](https://fonts.google.com/) **para descargar nuevas fuentes**

<https://www.w3schools.com/css/css_font.asp> **para saber cuáles registra Google**

Para cambiar la tipografía del texto.

 SRC es para ubicar dónde tienes guardada la font (si no es una estándar)

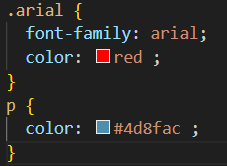
**** Si aparecen varias fonts registradas, lee de izq a dcha

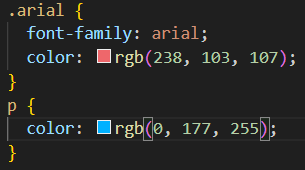
**HTML CSS**

**Color** [**https://coolors.co/**](https://coolors.co/)

Se pueden identificar los colores mediante RGB o con códigos hexadecimales.

Ejemplo con código hexadecimal:

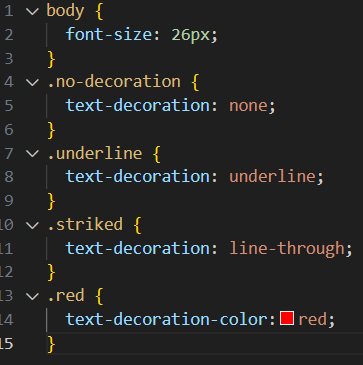
**HTML CSS**

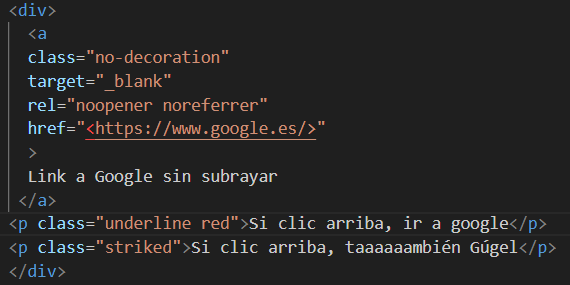
**Ejemplo RGB**

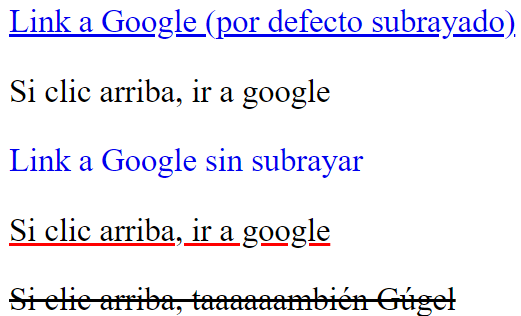
**HTML CSS**



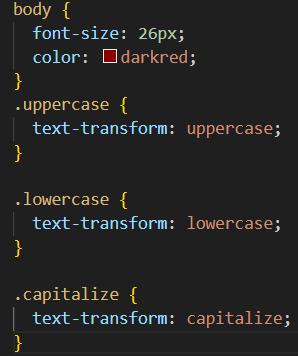
**Text-decoration**

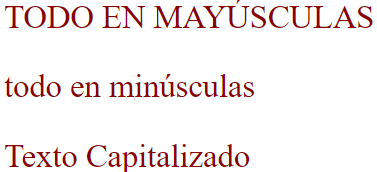
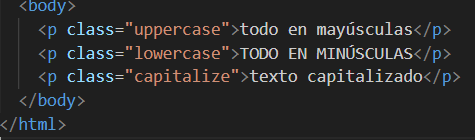
Cambia el estilo de “decoración” de un texto: subrayado, tachado… y a éste mismo se le pueden aplicar modificaciones como el color de la línea, tamaño, etc.

**HTML** **CSS 🡪**

**Resultado**

**Text-transform**

Con este comando podemos aplicar valores para convertir el texto a mayúsculas, minúsculas, capitalizarlo, etc.

**HTML CSS Resultado**

Estilos aplicados a estructura

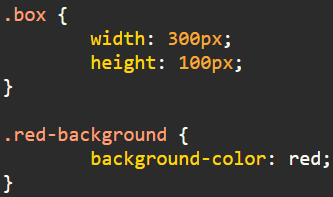
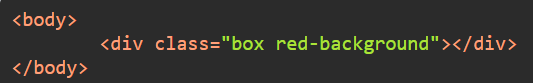
La estructura o **layout** de nuestra web nos ayudará a posicionar elementos donde creamos que sean necesarios, de forma que consigamos que la barra de navegación esté siempre arriba y el footer esté siempre abajo.

**Atributos Width y Height**

Algunos elementos del HTML, como los divs, no serán visibles a pesar de tener color de fondo si no tienen contenido de bloque (textos entre otros). Para hacerlos visibles cuando están vacíos (imagina crear una caja roja en la pantalla) tenemos que darle un ancho y un alto. Eso lo conseguiremos con las propiedades width y height.

|  |
| --- |
| Para que un elemento ocupe el ancho 100% de la pantalla (sea desde el dispositivo que sea) se representa en CSS con {width: 100vw} (ViewWidth)  Para que un elemento ocupe el alto de 100% de la pantalla se representa con {height: 100vh} ViewHeight  Se puede aplicar sobre elementos concretos, sobre todo el body, sobre lo que se desee….. |

Vamos a crear una caja de 300px de ancho y 100px de alto. Para ello, añadiremos un **div** sin contenido con las clases **red-background** y **box** a nuestro

******HTML CSS** 🡪 **Resultado**

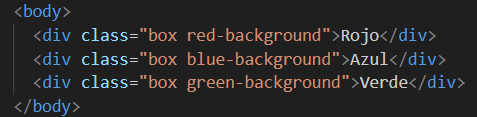
**Atributos display:** [**https://upgradehub.instructure.com/courses/343/pages/css-conceptos-basicos-display?module\_item\_id=7863**](https://upgradehub.instructure.com/courses/343/pages/css-conceptos-basicos-display?module_item_id=7863)

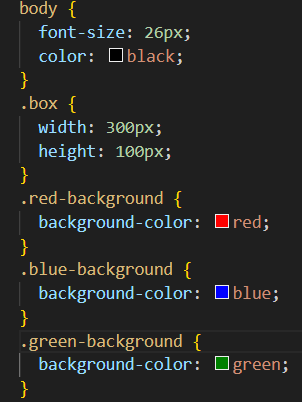
Todos los elementos de nuestra web van a tener un tipo de **display** o forma de mostrarse distinta por defecto.

* Inline. Valor por defecto. Expone un elemento como elemento en línea. No admite width, height ni margin
* Block. Expone un elemento como elemento de bloque. Este elemento ocupará todo el ancho posible.
* Inline-block. Su comportamiento es una mezcla de los dos anteriores, se muestran en la misma línea (respetando el flujo) todos los elementos y además, acepta las propiedades width, height y margin.
* None. Lo oculta de la web. Por ejemplo, las cookies.
* Flex. Expone un elemento como un elemento de bloque de tipo flexible, generalmente nos servirá para colocar elementos en una misma fila o columna.

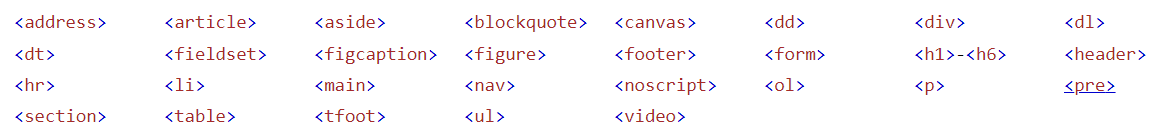
En el caso de los **div** tendremos elementos de tipo **display**: **block**, y para los **span** tendremos elementos del tipo **display**: **inline**.

Los **elementos** **block** ocuparán por defecto el espacio de ancho de pantalla que tengan disponible, no permitiendo a otros elementos estar a su lado. Esta opción es la que sale por defecto al crear **box**.

**Ejemplo en block HTML CSS Resultado**

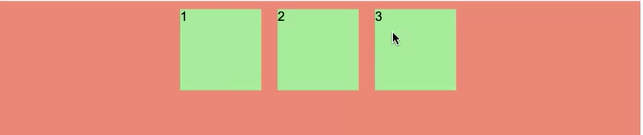
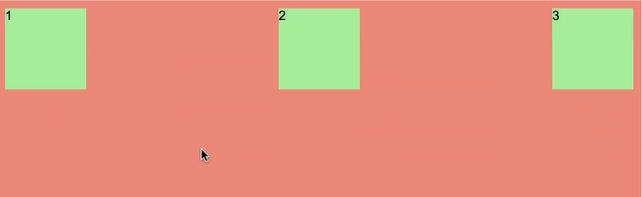


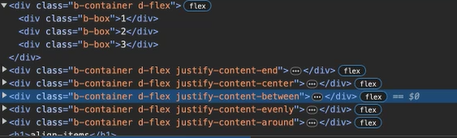
Tipos de elementos block-level



Los **elementos flex** es muy útil, especialmente en divs. Al declarar flex en el DIV PADRE, a los DIVS HIJOS directo les afecta poniéndolo. Básicamente, es como INLINE pero pudiendo modificar el width y el heigh.

Al poner flex, se habilita “justify content” para saber cuán justificado quieres el contenido del div a modo columna, horizontal.

(dcha/end, centro/center , between para dejar , izq)



Para que se recoloquen los elementos en función del tamaño del dispositivo (móvil, web) o la cantidad de elementos que añadas (p. ej imágenes)

Cuando se le añade display flex, para que no se nos salgan los elementos (ya que el máximo del dispositivo es el 100%), hay que añadirle la fórmula “calc” para que reste la medida que queremos de los márgenes y no dé más de un 100%.

Un DIV PADRE con Display flex del 100% de vista, hay que añadirle a los hijos el width calc que reste la parte proporcional

width: calc(25% - 20px);

height: 100px;

background-color: lightgreen;

justify-content: center;

align-items: center;

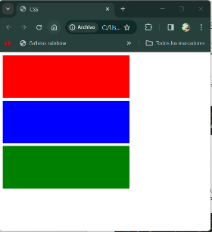
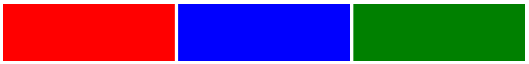
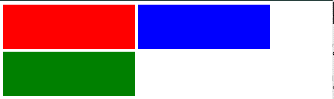
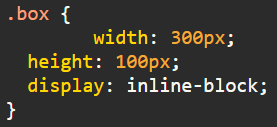
margin: 10px;

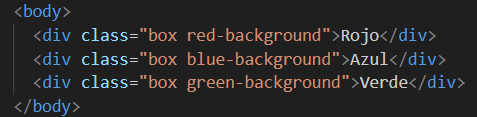
También está “align content” justifica los divs de dentro a modo fila (de manera vertical)

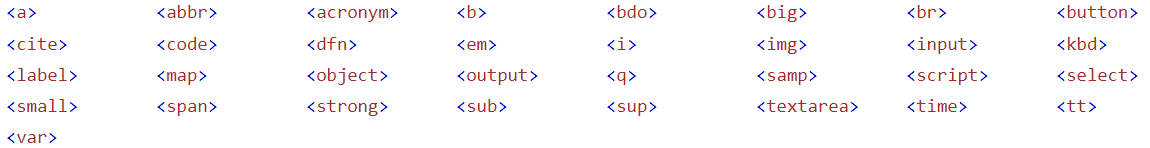
**Un elemento inline no puede contener un block-level**

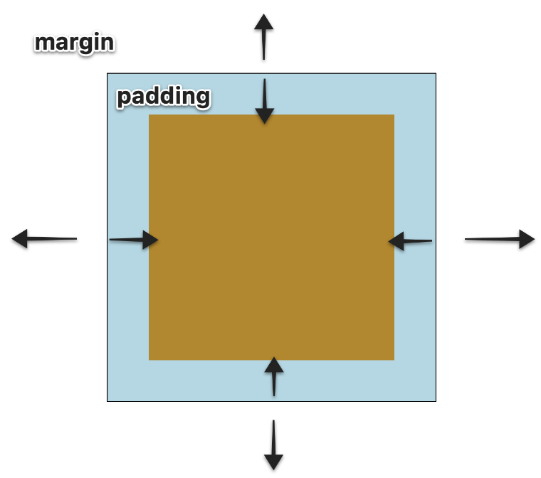
**HA USADO HEADER ARRIBA, PERO HA MENCIONADO QUE SE PUEDE PONER MÁS ADELANTE**

Los **elementos** **inline** se pegarán a sus elementos adyacentes intentanto ocupar el menor espacio posible. Pudiendo encogerse y adaptarse según hagamos estrecha la ventana de Google.

******Ejemplo inline HTML CSS Resultado**

****

Tipos de elementos inline-level

**Propiedades de espaciado: padding y margin**

Si consideramos cada elemento de la web como una caja, ¿cómo veríamos un <div> dentro de otro <div>?

Suponemos un <div> de background-color: lightblue que contiene a otro <div> de menor tamaño y con background-color: darkgoldenrod. Hemos centrado al div interior, pero no usando ninguno de los métodos anteriores, es la propiedad padding la que decide cómo se ve el contenido de dentro. Y si tuviésemos elementos cercanos al div superior, estarían empujados por la propiedad margin, razón por la cual está centrado en nuestro <body>.

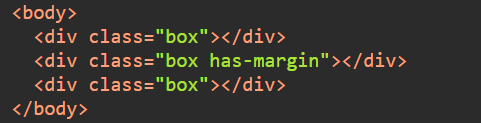
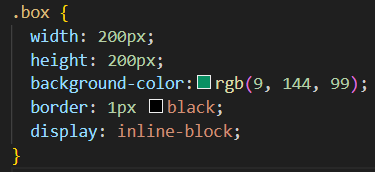
**Propiedad margin**

La propiedad margin es el espacio que es empujado por un elemento desde su borde hacia afuera, empujando otros elementos circundantes. Es decir, dado un <div> con margin, los <div> que lo rodeen estarían separados de este primero tanto como margin hubiese.

**Si queremos que TODA la web tenga margen ‘0’ y/o el padding ‘0’ (recomendable para poder aplicarle luego el que queramos) hay que hacer en css:**

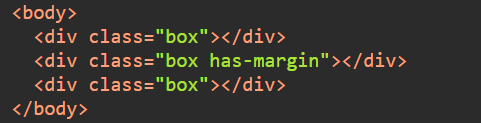
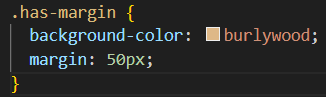
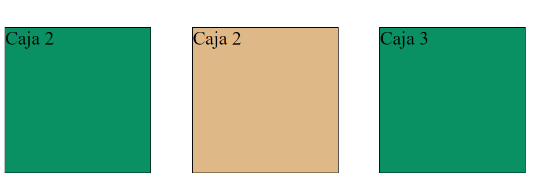
|  |
| --- |
| **\*{**  **margin: 0;**  **margin: 0;}** |

**Ejemplo gráfico de la función margin**

**HTML CSS Resultado**

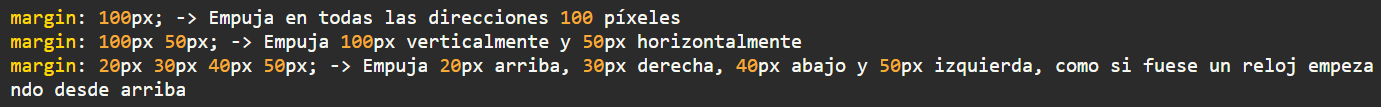


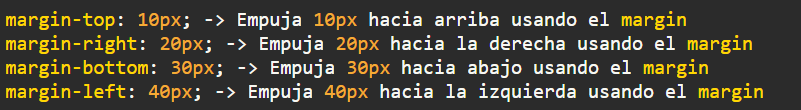
Al añadirle la propiedad margin y cambiarle el color a la del centro para que destaque quedaría:

**HTML CSS**

Si queremos que solo haya "empuje" con el margin en los laterales, vamos a cambiar el CSS para que afecte solamente a **margin-right** y **margin-left**.

Podemos hacerlo de dos formas, ya que la propiedad margin puede definirse con varios valores:

Opción 1

Opción 2

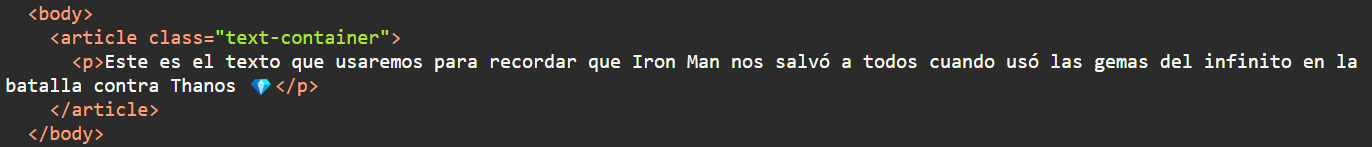
También se puede centrar un elemento dentro de su contenedor de manera absoluta con valor 0 auto;

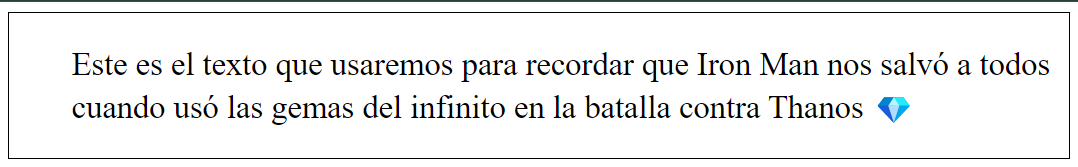


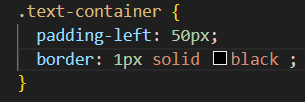
**Propiedad padding**

La propiedad padding es el espacio que es empujado por un elemento desde su borde hacia dentro.

Ejemplo gráfico de la función padding

**HTML**

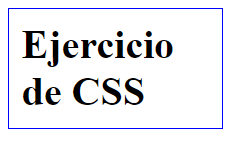
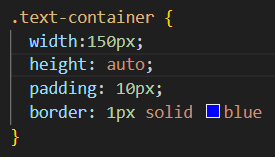
** CSS Resultado**



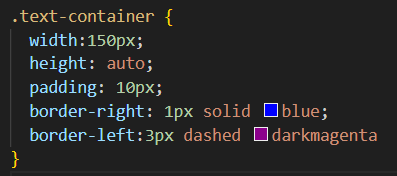
**Propiedad border**

Es la propiedad que delimita un espacio; lo que es un borde. Puede especificarse al igual que margin y padding:

**De manera uniforme** **CSS Resultado**



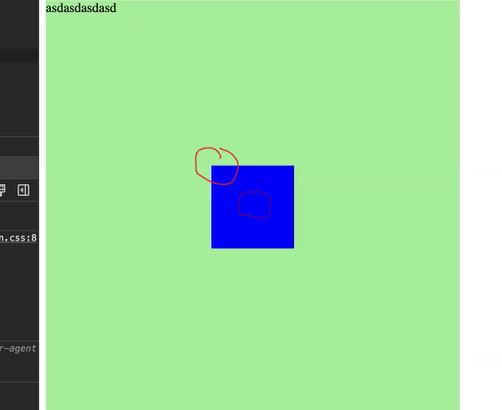
**O concretando sobre el lado que queramos CSS Resultado**



Position CSS

La propiedad position especifica que método de posicionamiento es usado para un elemento. Existen cuatro tipos de posicionamiento:

* **Static:** Posicionamiento por defecto. El elemento será posicionado acorde al flujo normal de la web. Propiedades como top, right, bottom y left no tendrán efecto.
* **Relative:** El elemento se posiciona acorde al flujo normal de la web. Propiedades como top, right, bottom y left tendrán efecto e iniciarán las coordenadas 0,0 en el punto de partida del elemento. Generalmente se utiliza para las “caja padre” y después poder maquetar sus hijos.
* **Absolute:** El elemento se posiciona relativamente al primer padre con el valor relative. Propiedades como top, right, bottom y left tendrán efecto e iniciarán las coordenadas 0,0 en el punto de partida del primer padre con el valor relative. Funciona como una capa en PS, para superponer capas/elementos. Generalmente se utiliza para las “cajas hijos” y poder superponerle lo q sea.
* **Fixed:** El elemento se posiciona relativamente a los extremos de la pantalla y quedará fijo aunque hagamos scroll. Propiedades como top, right, bottom y left tendrán efecto e iniciarán las coordenadas 0,0 en el borde superior izquierdo del viewport.



Para poder colocar una caja hijo dentro de una caja padre, al colocarle <position: absolut>, centrará la caja en base a la esquina superior de la misma, es decir, centrará su vértice izq sobre la caja padre. Para poder utilizar el centro de la caja como referencia, en vez del vértice, y así poder colocarla centrada. Habría q añadir

1. position: absolute;
2. left: 50%;
3. top: 50%;
4. transform: translate(-50%, -50%);

Cuando una imagen está dentro de una caja padre con un alto y ancho delimitado se le mete a css:

width: 100%; width: 25%;

height: 100%; Ó padding: 10px;

object-fit: cover; margin: 0;

box-sizing: border-box;

el box-sizing

(más contenido en canva)

Selectores avanzados NO SE USAN. USAR CLASES

Flex

Box. Los más usados son: flex-wrap, flex-direction, justify-content, align-items.

CSS Conceptos Básicos - Background

La propiedad background nos sirve para cambiar el fondo de un elemento. Esto nos permitirá realizar efectos muy chulos si conocemos todas las propiedades y sabemos usarlas correctamente.

Os dejamos una lista de las propiedades más importantes:

* \*\*background\*\*: Establece todas las propiedades de background en una sola línea, es lo que se conoce como un short-hand
* background-image: Establece la imagen de fondo de un elemento
* background-size: Especifica el tamaño de la imagen de fondo
  + Background-size: cover. Cubre el tamaño de toda la caja q contiene una imagen
* background-position: Establece la posición inicial de una imagen de fondo
* background-repeat: Establece cómo se repetirá una imagen de fondo
* background-color: Establece el color de fondo de un elemento (difícil eh?)
* background-attachment: Establece si una imagen de fondo es fija o se desplaza con el resto de la página
  + también conocido como parallax.

Metodología BEM

Bloques (caja padre)

Elementos (hijos)

