# Curso CSS – Apuntes

***Tabla de contenido***

[Curso de CSS – Apuntes 1](#_heading=h.gjdgxs)

[Inicio 3](#_heading=h.30j0zll)

[1.](#_heading=h.1fob9te) Formas de añadir CSS en HTML 3

[2.](#_heading=h.3znysh7) Estructura de CSS 4

[3.](#_heading=h.2et92p0) Tipos de selectores 4

[4.](#_heading=h.tyjcwt) Especificidad 7

[5.](#_heading=h.3dy6vkm) Metodología BEM 8

[6.](#_heading=h.1t3h5sf) Unidades 11

[7.](#_heading=h.4d34og8) Propiedades de texto 13

[8.](#_heading=h.2s8eyo1) Normalize 15

[9.](#_heading=h.17dp8vu) Cajas 16

[10.](#_heading=h.3rdcrjn) Padding 17

[11.](#_heading=h.26in1rg) Width y Height 17

[12.](#_heading=h.lnxbz9) Margin 18

[13.](#_heading=h.35nkun2) Bordes 19

[14.](#_heading=h.1ksv4uv) Box model 20

[15.](#_heading=h.44sinio) Outline 20

[16.](#_heading=h.2jxsxqh) Z index 21

[17.](#_heading=h.z337ya) Position 21

[18.](#_heading=h.3j2qqm3) Display 22

[19.](#_heading=h.1y810tw) Overflow 23

[20.](#_heading=h.4i7ojhp) Float 23

[21.](#_heading=h.2xcytpi) Pseudoelementos 24

[22.](#_heading=h.1ci93xb) Pseudo-clases 25

[23.](#_heading=h.3whwml4) Object-Fit - imagenes 26

[24.](#_heading=h.2bn6wsx) Object-position 27

[25.](#_heading=h.qsh70q) Cursor 27

[26.](#_heading=h.3as4poj) Colores 27

[27.](#_heading=h.1pxezwc) Conceptos básicos de responsive 28

[28.](#_heading=h.49x2ik5) Conceptos de FlexBox 28

[29.](#_heading=h.2p2csry) Flex direction 29

[30.](#_heading=h.147n2zr) Flex wrap 30

[31.](#_heading=h.3o7alnk) Alineación en flex 30

[32.](#_heading=h.23ckvvd) Trabajando con flex-items 31

[33.](#_heading=h.ihv636) Grid - Conceptos y propiedades 33

[34.](#_heading=h.32hioqz) Grid implícito y grid explícito 36

[35.](#_heading=h.1hmsyys) Grid dinámico 36

[36.](#_heading=h.41mghml) Alineación en Grid 37

[37.](#_heading=h.2grqrue) Grid Areas 38

[38.](#_heading=h.vx1227) Nombrar Grid-line 40

[39.](#_heading=h.3fwokq0) Grid shorthand 40

[40.](#_heading=h.1v1yuxt) Responsive Design – Tema full 41

[41.](#_heading=h.4f1mdlm) Transition 42

[42.](#_heading=h.2u6wntf) Animaciones 42

[43.](#_heading=h.19c6y18) Cubic bezier 44

[44.](#_heading=h.3tbugp1) Transform 44

[45.](#_heading=h.28h4qwu) Background 45

[46.](#_heading=h.nmf14n) Variables 46

[47.](#_heading=h.37m2jsg) Filter 47

[48.](#_heading=h.1mrcu09) Otras propiedades 48

[-](#_heading=h.46r0co2) Sigue aprendiendo - 48

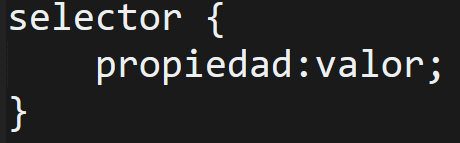
# Inicio

## Formas de añadir CSS en HTML

Hay 4 formas. (Ordenadas de mejor a peor, en cuanto a las formas más óptimas)

* Link, usando rel como “hoja de estilo”, y href es donde está alojado  
  
* Línea, es decir que se usa como un atributo más  
  
* Etiqueta Style  
  
* Import (Ya no se usa)  
  

## Estructura de CSS

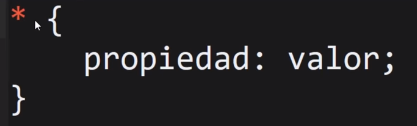
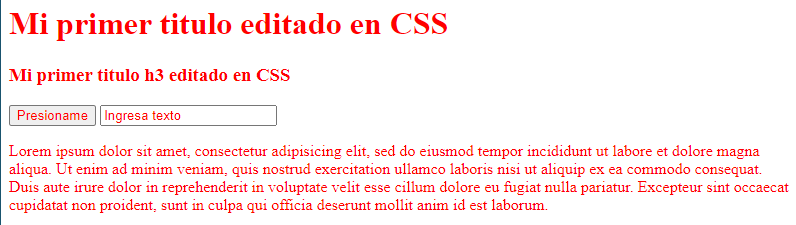
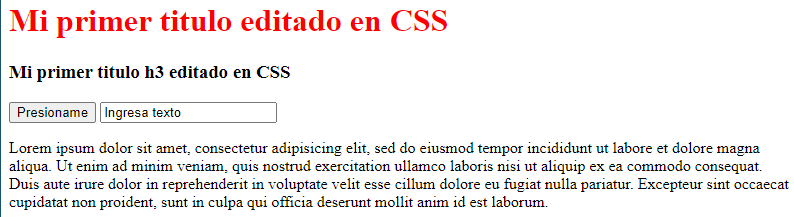
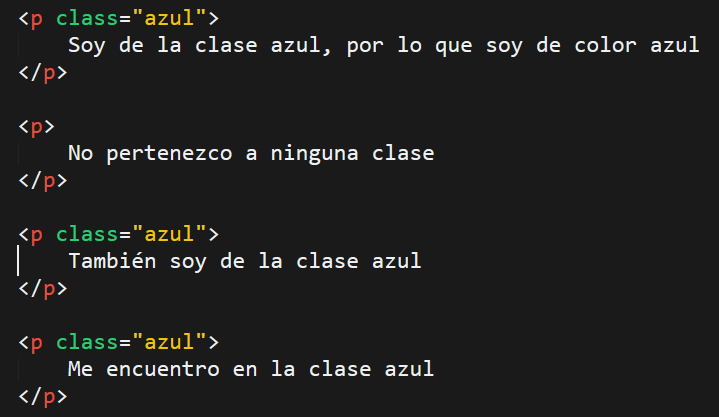
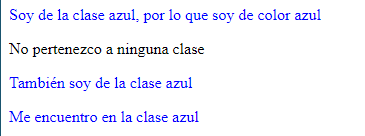
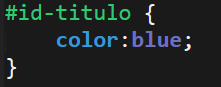
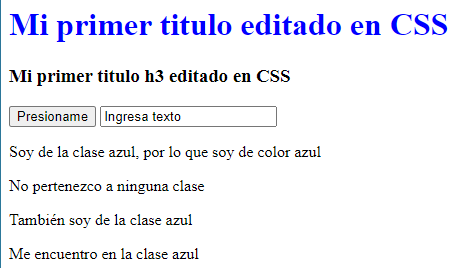
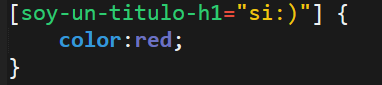
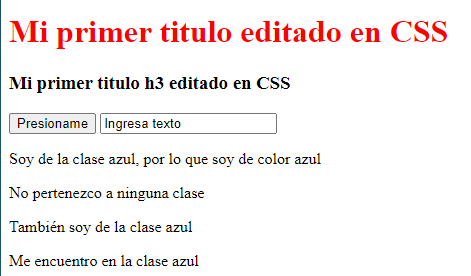
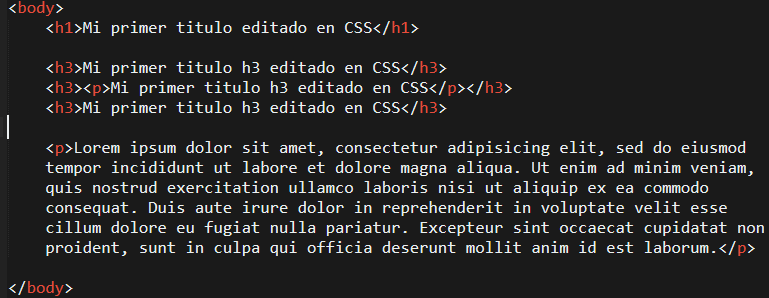
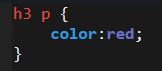
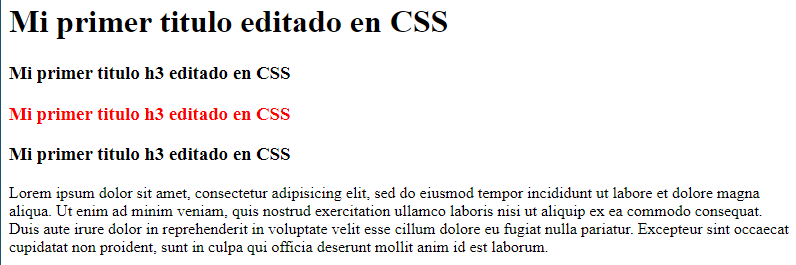
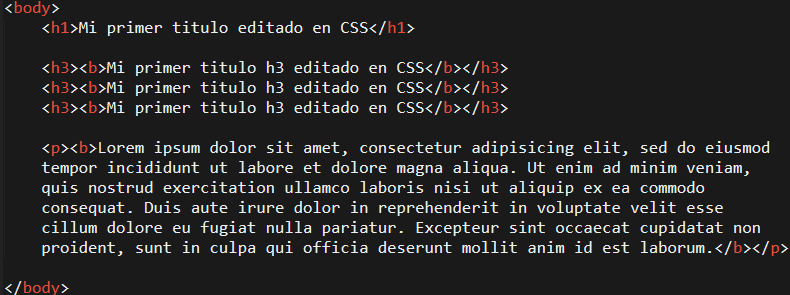
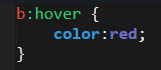
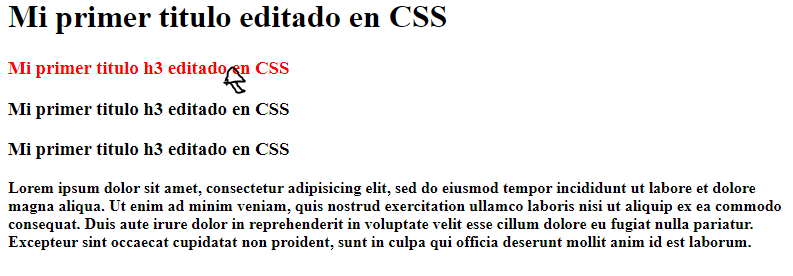
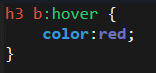


Donde:

* Selector: Es lo que se desea cambiar
* Propiedad: Qué se le va a cambiar, color, tamaño, posición, bordes, etc.
* Valor: Cómo se va a cambiar

## Tipos de selectores

Primero que todo, aclarar que el valor siempre tiene que terminar con punto y coma “ ; ”

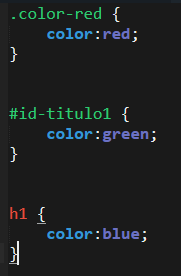
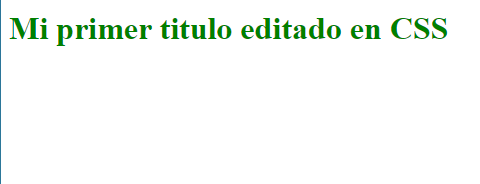
* Universal: Selecciona todos los elementos, por lo que, si la propiedad color tiene valor rojo, todo estará en rojo  
     
  
* De tipo: Selecciona directamente con el nombre de los elementos  
    
  
* Clases: Selecciona a las etiquetas por clases que se le añaden con el atributo class. Se debe colocar un punto y después el nombre de la clase  
     
  
* ID: funciona como las clases, solamente que este ID debe ser único, solamente para un elemento.  
    
   
* Por atributo: Se crea un atributo dentro de la etiqueta HTML y se definen sus propiedades en CSS  
    
  
* Descendiente: Aquellos que están dentro de una etiqueta, por ejemplo, un párrafo dentro de un <h#>. Se debe poner de mayor prioridad a menor prioridad, de izquierda a derecha.  
     
    
  Se puede combinar con los de tipo, atributos, clases, ID y pseudo-clases
* Pseudo-clases: Se trata de los eventos donde se genera una selección, hay varias, pero en este caso se usará “hover”. Por ejemplo, al pasar el mouse por un título.  
      
  Y combinado sería que solo los b que están en h3 tengan el evento hover  
  

Hay muchas más formas de seleccionar, pero se verán más adelante.

## Especificidad

Es la prioridad con la que se definen los atributos, aquí una tabla  
[](#_heading=h.gjdgxs)

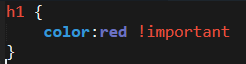
Por lo que si se tiene este código html junto con este de css

Le da preferencia al color verde ya que es un identificador, y no importa que después diga que los h1 deben ser azules o que antes diga que las clases deben ser rojas.

Los siguientes que tienen más prioridad son los estilos en línea, es decir que se establecen en la etiqueta, por ejemplo:  
  
  

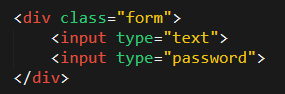

El de mayor prioridad es el important, aunque no es recomendable usarlo se usa de la siguiente manera:

A pesar de que el h1 tenga un style que le defina el color azul, si le agrego el ¡important así sea a un elemento, se tomará como mayor prioridad.  
  
 

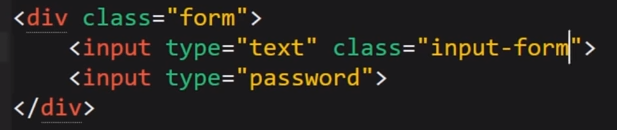
Aquí cambiaría la estructura del código por el hecho de que no se colocaría el punto y coma después del color (valor).

## Metodología BEM

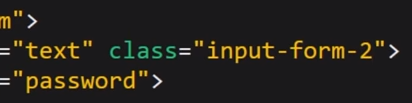
Es una metodología para evitar conflictos a la hora de crear muchas clases. El programador se dará cuenta de qué se está seleccionando a partir de las clases.



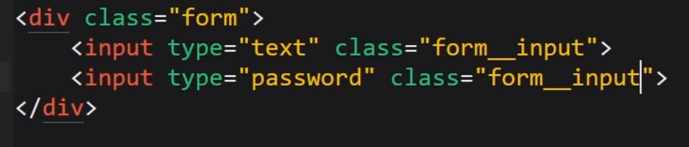
Por ejemplo, aquí se tiene un div con la clase form. Lo que no se debe hacer es esto:



Ya que en la página web, pueda haber más de un formulario, poner un “2” tampoco lo solucionaría



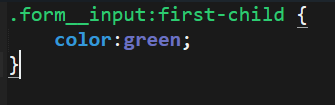
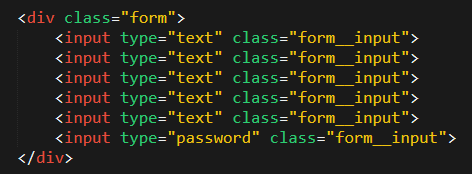
¿Entonces cuál sería la solución? Poner el nombre de la clase del contenedor (div), dos guiones bajos y luego el nombre del elemento. Esta es la forma más óptima para crear clases



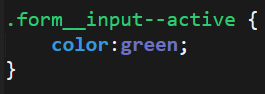
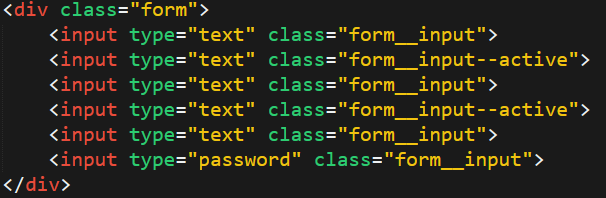
Se podría mejorar siendo más específicos y definiendo de qué es el formulario.



Ahora ¿Si tengo varios inputs y solo quiero cambiar el primero? Lo que debo hacer es escribir el siguiente código en CSS.

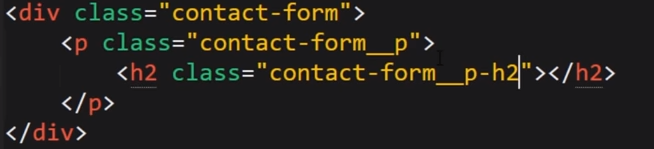


También se puede poner doble guión y una palabra que lo caracterice (active, por ejemplo), para aplicarle la propiedad solo a ciertos inputs dentro del form:

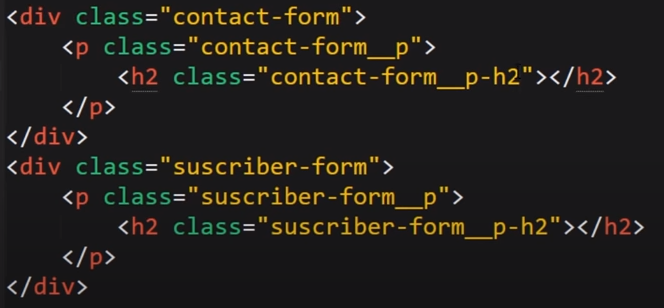




En el caso de tener varias etiquetas, la hija principal, en este caso p, llevaría los dos guiones bajos y después en el h2, se añadiría un guión simple.



Tener en cuenta que esto se hace por secciones, si se tiene otro div, se cambia la sección, aun así, la estructura es la misma.

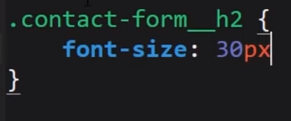


## Unidades

En CSS hay dos unidades de medidas.

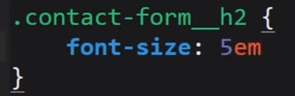
* Las relativas: Dependen de algo para establecer su valor, son variables. Las medidas relativas suelen depender de la caja contenedora (Divs)
* Las fijas: Ya están establecidas, por ejemplo, pixeles, milímetros, centímetros, etc.

Propiedades:

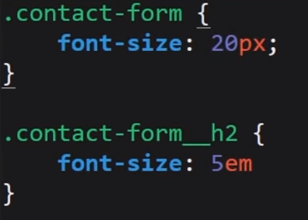


Cambia el tamaño de la letra, en este caso a 30px. Aunque lo mejor sería trabajar con la unidad em, ya que es relativa respecto al Div, y esto va a ayudar cuando la página web se abra en dispositivos móviles y tenga resoluciones diferentes de pantalla.

¿Cómo se hace esto? Se llama Responsive design y es trabajando con medidas relativas. Aquí se usará REM



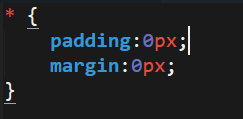
¿Cuánto es 1em? Naturalmente depende del navegador, pero suele ser 16px, por lo que la medida real sería 80px. Pero se puede heredar y el programador darle la equivalencia que desea.

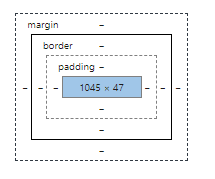
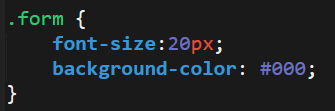


Si el contenedor tiene 20px, entonces 1em también serán 20px y 5em = 100px.

Como medida relativa, también se tiene viewport height and width. Lo que hacen es acomodarse al tamaño de la pantalla. Por ejemplo, 100vw (viewport width) será el largo total de la pantalla y 100vh el ancho total de la pantalla.

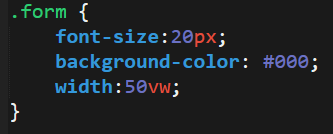
Para comprobar esto, se quitará el padding y margin dejando el texto pegado al borde, además de ponerle un fondo negro a modo de prueba.

Sucede que cubre toda la pantalla a lo largo, pero esto se puede cambiar con el viewport  

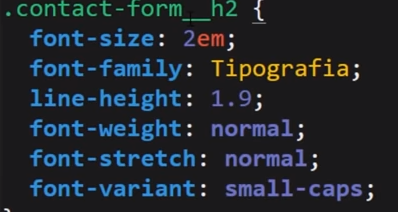

Se añadirá el viewport width al 50vw. Dejando así solo el 50% de la pantalla.

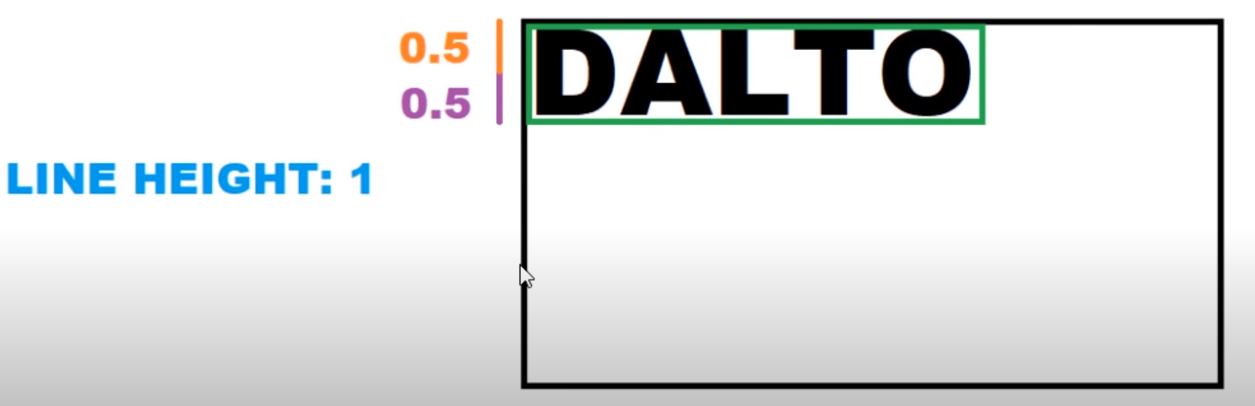
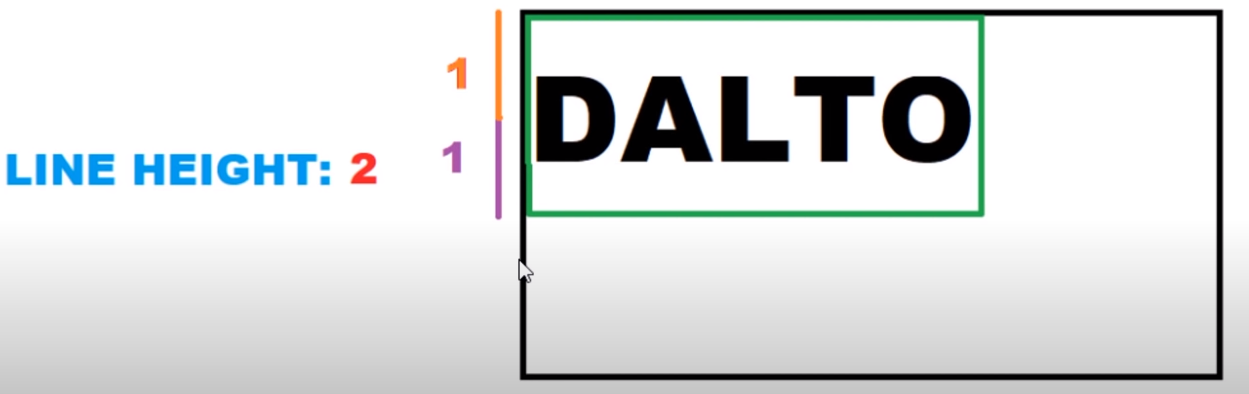
 Con viewport height también se podría realizar.



## Propiedades de texto

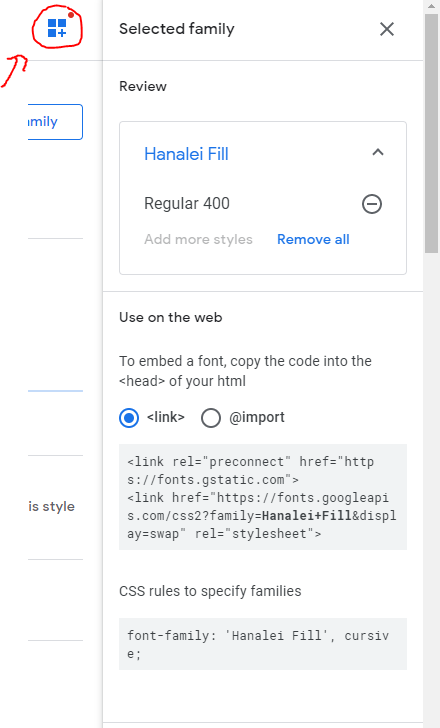
Algunas de las más usadas:

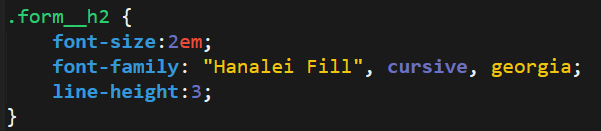
 Donde

* Font-size: El tamaño de la letra (Puede trabajarse con unidades relativas)
* Font-family: Tipografía (Arial, Times New Roma, Georgia, etc.)
* Line-height: Lo que ocupa la letra, se divide del centro dependiendo del valor asignado. No quiere decir que sea más grande la letra, solo ocupará mayor espacio.  
   
* Font-weight: El grosor de la tipografía.
* Color: Color del texto

En cuanto las fuentes, se pueden importar de varios sitios web, el mas popular siendo [Browse Fonts - Google Fonts](https://fonts.google.com/), Una vez allí se escoge la que más guste y se selecciona.  
  


Después clic en los cuadros con +. Se copia el codigo en head y el otro en las propiedades del texto



Tiene varias tipografías porque en caso de no encontrar la primera, pasa a la segunda y de no ser encontrada tampoco, a la tercera.

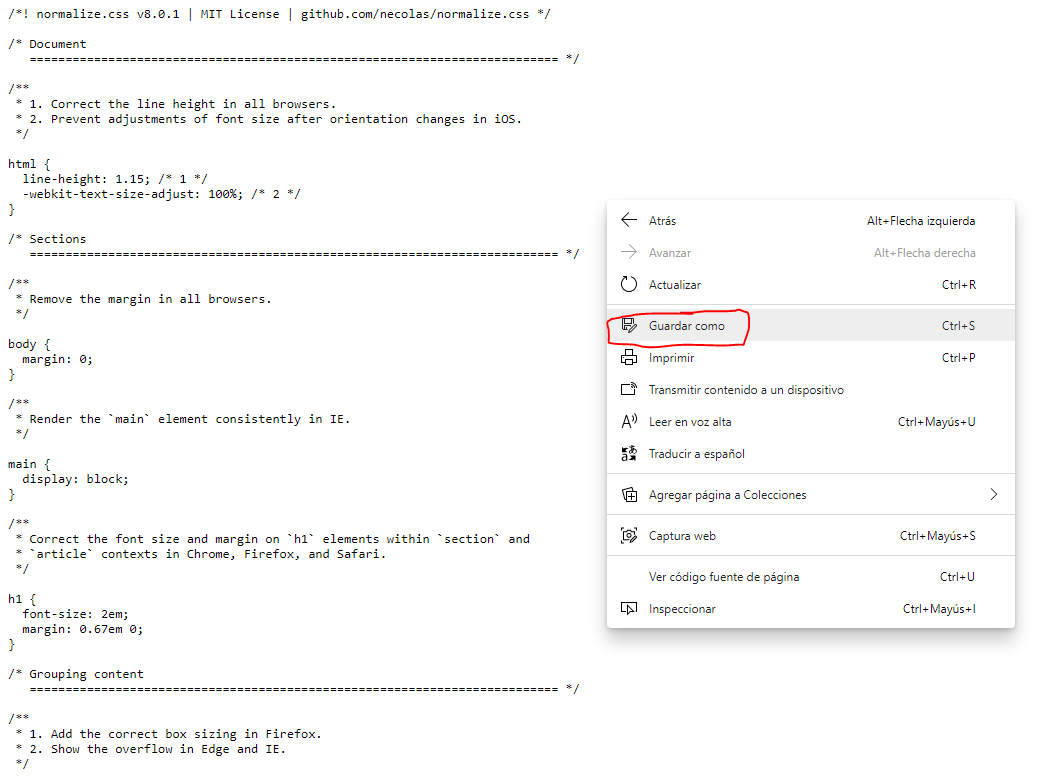
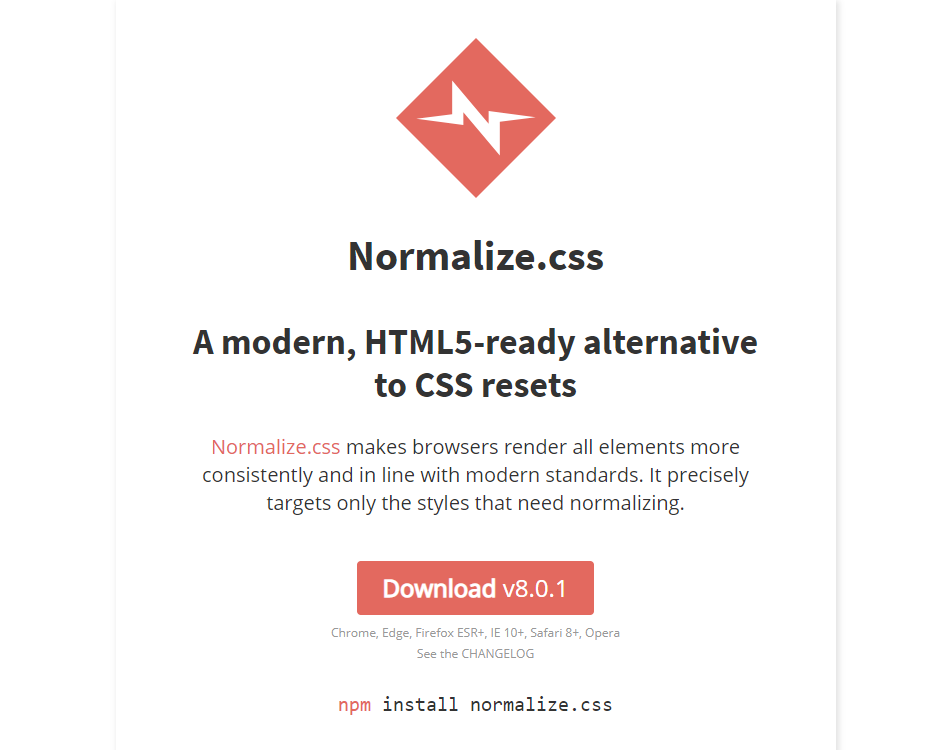


## Normalize

Usualmente los navegadores tienen propiedades por defecto, como el tamaño, color, margen, etc. Pero pueden llegar a generar problemas ya que no todos los navegadores utilizan los mismos valores por defecto y mucho menos en dispositivos móviles.

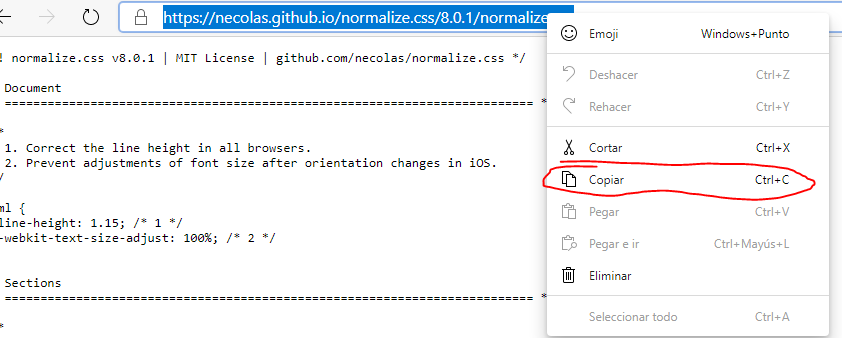
¿Cómo se cambia eso? Se hace de 2 formas, descargando un archivo o con el enlace al archivo (Se recomienda descargarlo)

[Normalize.css: Make browsers render all elements more consistently. (necolas.github.io)](https://necolas.github.io/normalize.css/)



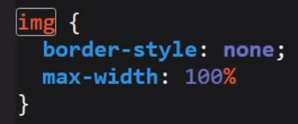
Cuando se oprime Download aparecerá todo el código, lo que se debe hacer es clic derecho > guardar como > y el formato debe ser .css > guardarlo en la misma carpeta del index.html

Para incluirlo, desde el head se inserta.  

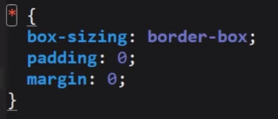

Otra forma es copiar y pegar el link  




Abrimos el normalize.css y con ctrl + F se busca img para ponerle max-width: 100%, así las imágenes en los móviles no den problemas y los ajuste



Además de ajustar el borde de las cajas y sus propiedades de margin y padding (Se entenderá mejor en la teoría de cajas)

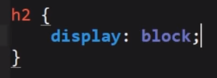
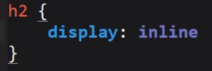
 no se coloca unidad porque cualquier unidad en 0 es nada.

## Cajas

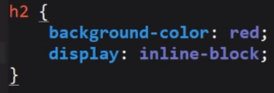
Hay dos tipos de cajas en html:

* Bloque: Siempre se ajustan al ancho del contenedor (h1, h2, h3, h4, h5, h6, p, etc.)
* Línea: Su ancho se acomoda a su ocupación real en pantalla (b, i, strike, small, etc.)  
  No se le pueden poner las siguientes propiedades: height, width

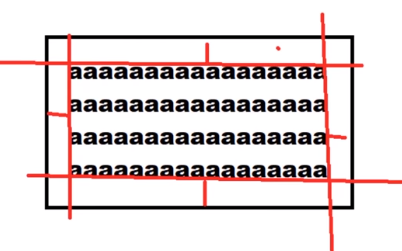
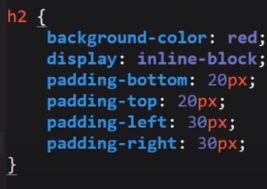
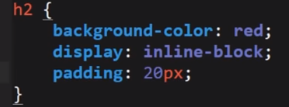
Se puede cambiar su comportamiento entre sí, modificando la propiedad “display”

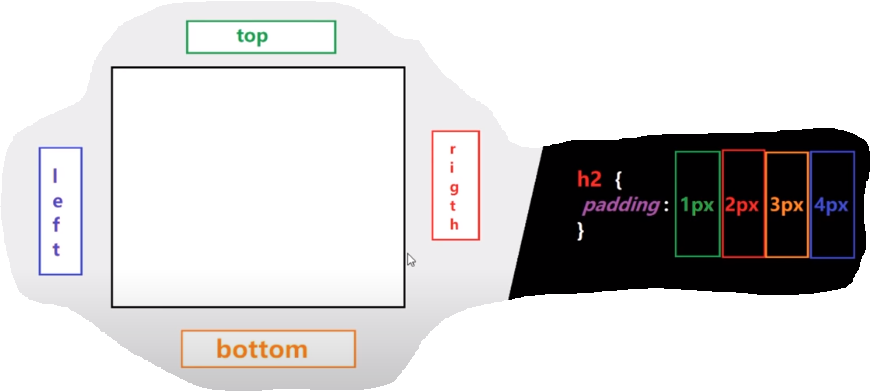
 ---------> 

Propiedades de cajas:

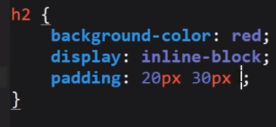
* background-color: Color de fondo de la caja  
  
* display: cambia las propiedades de la caja, dando valores de cajas lineales, de bloques o de ambas. (inline, block, inline-block)  
    
  

## Padding

Es la distancia que existe entre el texto y la caja. Además, se puede ajustar cada uno de sus lados.  
  A todos los lados 🡪 



Si se hace esto:

 Quiere decir que top - botton tendrán 20px y left – right tendrán 30px

El orden para definir las medidas es: Top, Right, Botton, Left. Si se hace esto:  


Y en caso de no tener el cuarto número es: Top, Right, Botton y Left tendría la misma medida de Right



## Width y Height

Definen lo height = alto y width = ancho

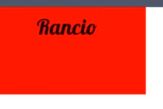


en este caso sería un cuadrado



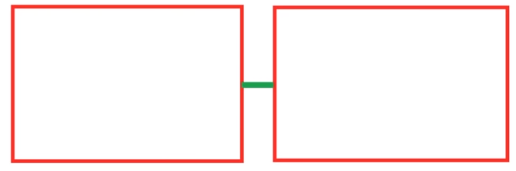
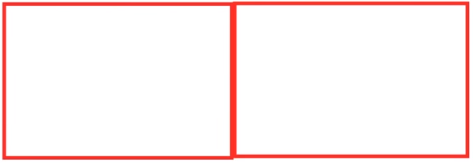
Pero ¿Por qué el texto está por fuera?

Esto se debe a la propiedad que se ha puesto en normalize.css  
 Esto para que no afecte a otras propiedades, ya que si no se tuviera y fuera así: (por defecto) 

 🡨 La caja ahora no es un cuadrado de 100x100px, ya que el padding es de 50px a los lados, al ser dos lados son 100px, así que ahora la cada tiene 200px a los lados. Además de 120px de arriba y abajo. Pero eso es una deformación, así que es mejor dejarlo con border-box para mantener el cuadrado.

El texto se puede configurar con text-aling: center; aunque hay formas mejores.

## Margin

A diferencia del padding que es la distancia del texto al borde de la caja, el margin es la distancia que hay entre caja a caja. Si hay margin de 0 las cajas estarán pegadas.  
 

También tienen las propiedades Top, Right, Botton, Left.

Al colocar un margin de 10px a todos los lados sucede esto 

 Se ve como la distancia entre cajas no cuadra con la distancia del tope superior. ¿Entonces cómo se puede poner uniforme?

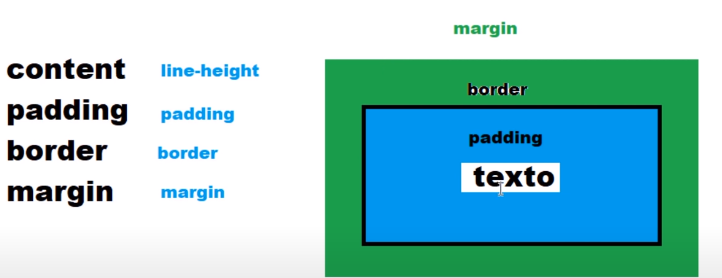
Esta es una solución  así hay 20px arriba y abajo, además de 10px que juntando las dos cajas darían también 20px a los lados.

## Bordes

Están las propiedades:

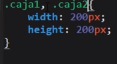
* border-radius: valor; esto para redondear los bordes. Si el valor es 50% se redondea completamente   
   
* border: Tiene varias cortaciones (border-width, border-color, border-style, etc) pero se pueden simplificar de la siguiente manera:  
    
  border: tamaño estilo color  
    
  Tamaños: em, px, mm, cm, etc.
* Estilos de bordes  
  Solid: dashed double groove, inset, ouset dependen del color  
       
* box-shadow: ejex ejey desenfoque borde(0) color (Se hace mas intenso con comas)  
    
  
* Text-shadow: igual que box-shadow pero con texto y sin borde(0) (se puede hacer más intenso separando con comas)   
  
* Transform: rotate(gradosdeg)   
  

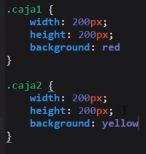
## Box moddel

El modelo en el que se trabajan las cajas, y sus jerarquías:  


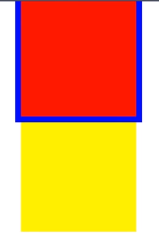
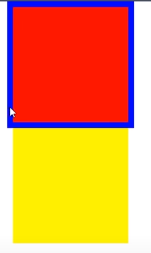
## Outline

Es como un border, la diferencia es que no ocupa un espacio real en el DOM (document object model), el DOM son todos los objetos que hay en la página, al poner el Outline no afecta a las posiciones, tamaños ni a nada, solo resalta.  
Una forma de no escribir dos veces el código para dos clases que tienen exactamente las mismas propiedades es:

, pero en caso de no ser exactas se hacen por separado



El Outline, en si consiste en colocar un borde sin aumentar el tamaño ni quitarle campo a otros elementos, por ejemplo, aquí están las dos cajas de 200x200px, pero al añadirle el borde a la caja roja (Y sin cambiar la propiedad de box-sizing) ocuparía ya no 200x200px sino que se le suman las dimensiones que tengan los bordes a cada lado.

Pero si en vez de un borde se le da un Outline se tendrán las dimensiones 200x200 y conservando el borde  


## Z index

Es la forma en la que se va a colocar un elemento por encima de otro. En el eje Z los ordena, el valor más alto lo mostrará por encima del más bajo. Se marca en código como z-index: valor

El z index funciona siempre y cuando la caja esté posicionada, además se recomienda dejar valores de 10 en 10 entre cada z index.

Tiene complejidad para las cajas contenedoras e hijas.

## Position

Posicionar los elementos, es decir, añadirle nuevas propiedades de posición afectando el flujo HTML (el orden de los elementos). Se marca en código como position: valor; y las posibilidades son:

* Static: El valor por defecto, indica que no está posicionado.
* Relative: El espacio asignado por el HTML se sigue conservando a pesar de poder moverse en el entorno sin afectar a los demás objetos. Las propiedades más importantes son Top y Left.
* Absolute: No existe un espacio reservado para la caja, además si se le aplica a un Div, este se ajustará al contenido, cosa que no pasa con un relative.
* Fixed: No ocupa espacio reservado y se apega a la pantalla sin importar el scroll.
* Sticky: Se conserva el espacio reservado y además se desplaza como el fixed a partir del punto donde se desee.

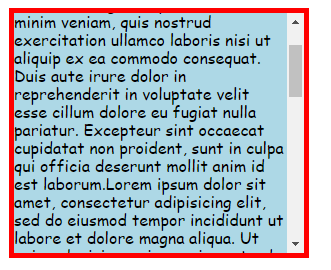
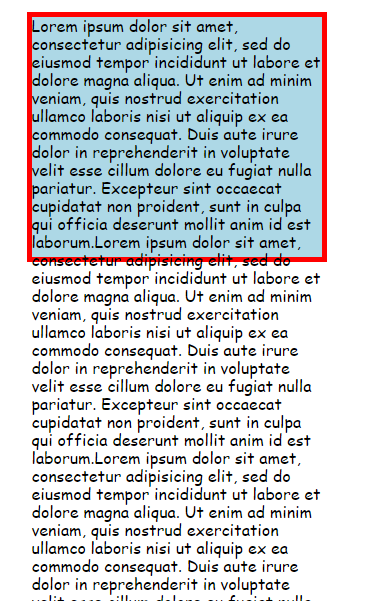
## Display

Es una propiedad que modifica el comportamiento de las cajas.

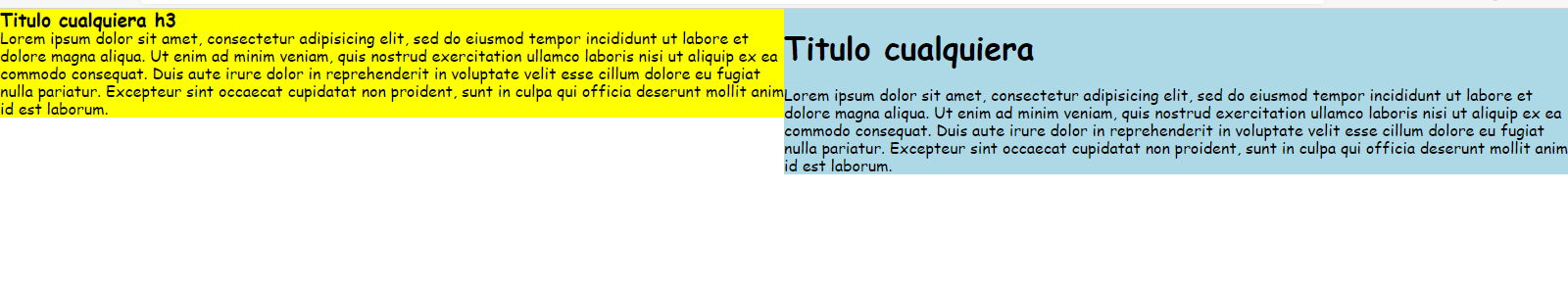
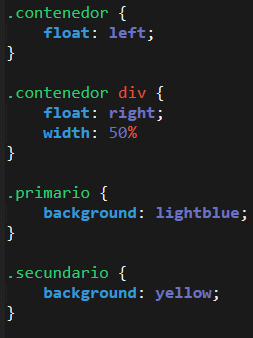
* Block: Comportamiento de bloque
* Inline: Comportamiento de línea, generalmente para texto (adaptable al contenido)
* Inline block: Comportamiento de bloque, pero modificando el height y width, además de colocar bloques juntos.
* Table: Comportamiento de tabla.
* Inline table: Comportamiento de tabla en línea.
* List item: Comportamiento de lista <li>.
* Flex:
* Grid:
* Inline Flex:
* Inline Grid:

## Overflow

Sirve para no sobre pasar texto en cajas Div. El valor por defecto deja que el texto se salga, el valor “auto” sirve para crear un scroll en la caja. También está hidden que es para ocultar el scroll (Con sus variantes Overflow-x / y)  
  
Por defecto Auto



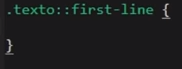
## Float

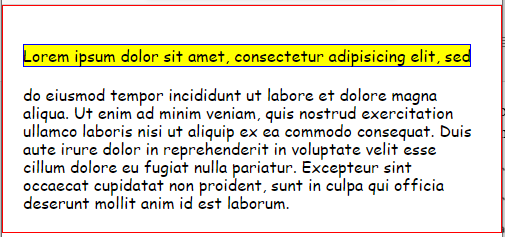
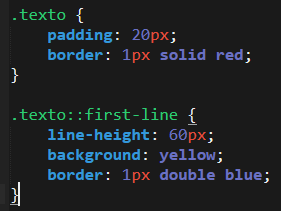
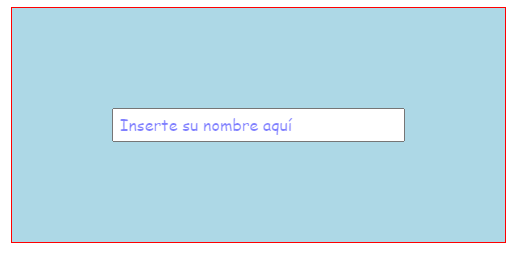
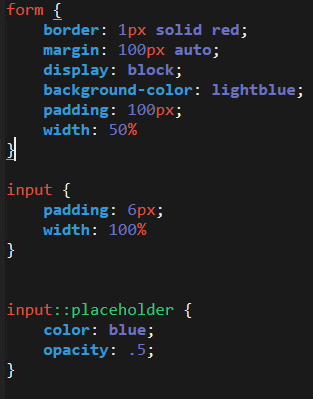
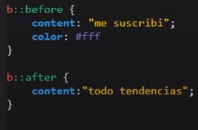
Organizar las cajas a los lados de la pantalla, se utilizaba mucho antes, ahora hay formas más óptimas de hacerlo.  


## Pseudoelementos

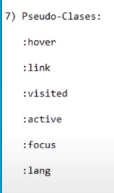
No forman parte del DOM, pero si se pueden apreciar cambios visuales (Parecido al outline) algunos de ellos son:



Para usar un pseudoelemento se coloca dos doble punto seguido de la clase.  


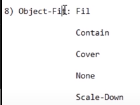
* First-line: sirve para cambiar las propiedades de la primera línea.  
  
* First-letter: Cambia las propiedades de la primera letra.
* Placeholder: Útil en los input aportando información que no afecta al DOM.  
    
  
* Selection: Sirve para cambiar las propiedades de selección, como el color.
* Before y After: Son hijos del elemento al cual se les aplica, si o si necesitan la propiedad content y además son elementos en línea.  
  

## Pseudo-clases

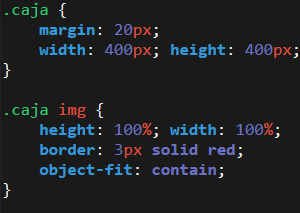
Se trabaja como un pseudoelemento, pero en este caso lo que hace es escuchar un evento.  


* Hover: identifica cuando el mouse está encima y ejecuta la acción que se defina en propiedades. Funciona en bloques y líneas.
* Link: Define las propiedades de un link antes de ser visitado (antes de oprimir el link)
* Visited: Cambia sus propiedades (usualmente el color) cuando el link fue visitado (<a href=””>)
* Active: Funciona cuando el elemento es presionado.
* Focus: Cambia sus propiedades cuando el elemento está seleccionado (puede servir en inputs de formuario)  
   🡪 
* Lang: Casi no es usado, su función es modificar lo que esté escrito en otro idioma especificado (en, es, ru, etc.)  
    

## Object-Fit - imagenes

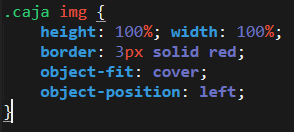
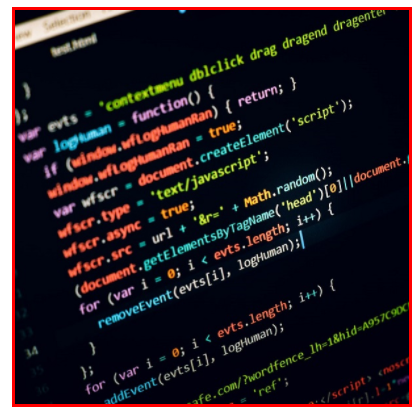


Es una propiedad que se usa en imágenes, cambia sus dimensiones, tamaños, etc.

* Contain: El valor que viene por defecto, se adapta al contenido de la caja, dejando espacios en blanco.  
   
* Cover: Amplía o reduce la resolución de la imagen, en caso de ampliar recorta lo sobrante para que encaje al contenido de la caja.  
  
* None: Pone la resolución original de la imagen en el contenedor.
* Scale-down: Se queda con la resolución más pequeña entre Contain y none.

## Object-position



 Al estar en Cover se ampliará, pero se junta con object-position Left, lo que quiere decir que se mostrará la parte izquierda.  
 🡪 

Lo mismo pasa con Right, Top y Botton. También se pueden mover al gusto con unidades.  
  


## Cursor

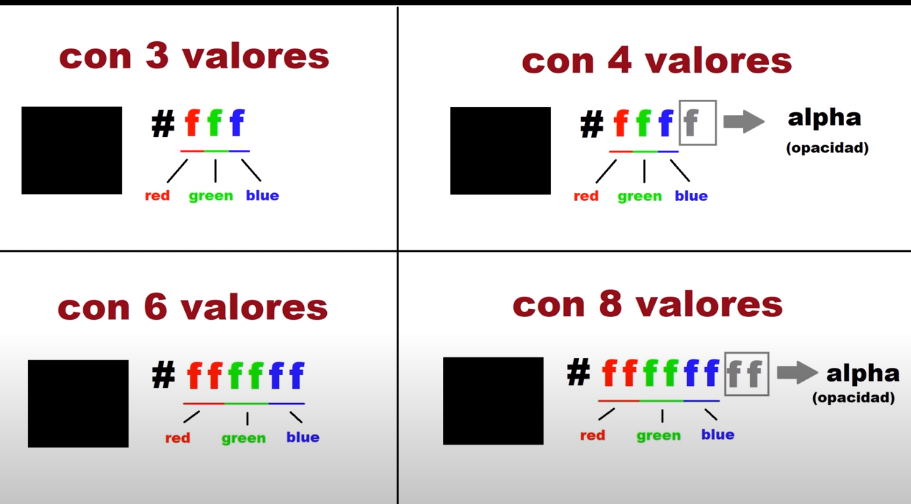
Propiedad que permite cambiar el cursor, se puede poner en cajas y pseudo-clases.

Lista de cursores: [Tryit Editor v3.6 (w3schools.com)](https://www.w3schools.com/cssref/tryit.asp?filename=trycss_cursor)

## Colores

Se pueden trabajar los colores por nombre (red, blue, yellow, pink, etc.), por hexadecimal (#f00 = rojo). Y por RGB.

Para trabajar con RGB se debe saber que su significado es rojo, verde y azul. Además, sus valores van del 0 al 255, y se juntan entre sí para dar un color de origen.  
, los valores van en orden primero rojo, luego verde y por último azul. Con RGBa lo que se crea es un cuarto parámetro que define la opacidad del fondo.

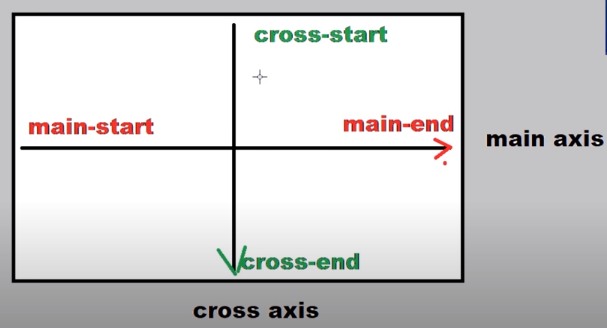
En hexadecimal se tienen 3, 4, 6 u 8 valores, variando de las necesidades.  


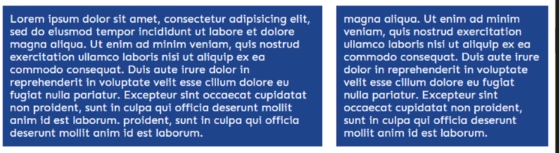
## Conceptos básicos de responsive

Cuando se trabaja con responsive design se busca adaptar el contenido a diferentes resoluciones. Se recomienda adaptar el concepto de movile first, es decir, crear la página web a móviles y después a tablets y computadoras.

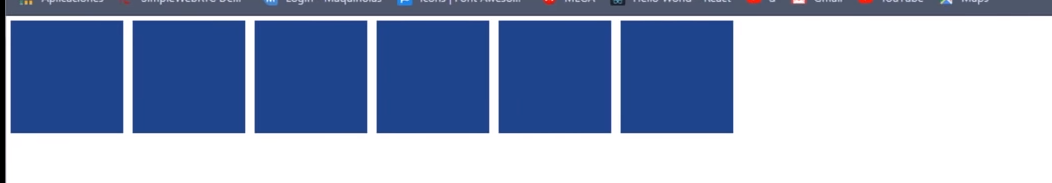
Existe una particularidad especial de CSS para trabajar con responsive design  
 La cual indica que, si la resolución es menor a 800px, hará lo que se le indique (Comportamientos en bloque, textos más pequeños, entre otros.) Además, es importante tener metadatos como meta viewport que ayudarán a conseguir un mejor efecto.

## Conceptos de FlexBox

Los cambios se reflejan dentro del contenedor, solamente los hijos directos son flexitems. Estos son los elementos por defecto:  


El contenido se ajusta al contenedor. Esto pasa cuando se tiene un display flex en la caja que las contiene. Además, no hay problemas de altura.  


Si hay varios contenedores flex, se ajustarán a la resolución. Hasta el punto definido

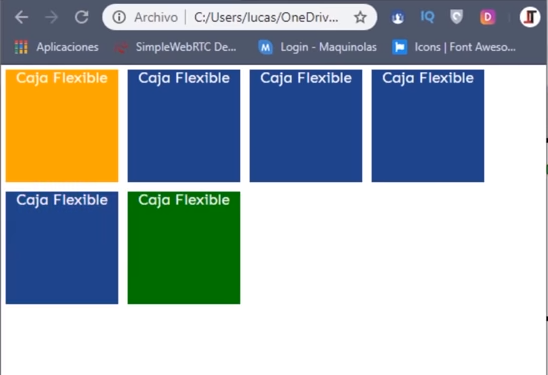


## Flex direction

Cambiando la dirección del main axis se logra girar de horizontal a vertical. La propiedad es flex-direction, se aplica al contenedor para afectar a los hijos. Valores:

* Row: Valor por defecto, comportamiento como filas (horizontal).
* Row-reverse: Se comporta como un espejo horizontal cambiando el main-start y main-end
* Column: Se comportan como columnas, de arriba hacia abajo.   
  
* Column-reverse: Es un espejo del column cambiando el cross-start y el cross-end

## Flex wrap

Si el contenedor se vuelve cada vez más pequeño, ahora las cajas no se apretarán ni se ajustarán, sino que se desplazarán hacia abajo. . Y con wrap-reverse, las cajas van hacia arriba.  


El nowrap es el valor por defecto, no sucede nada.

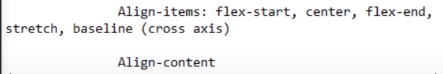
Para cambiar flex wrap y flex direction se usa flex-flow .

## Alineación en flex

.

La propiedad justify-content tiene varios valores.

* Center: Es para centrar el contenido.
* Space-arround: Reparte el contenido, se le da un margin auto.
* Space-between: Se alejan las cajas al máximo unas de otras.
* Space-evenly: Reparte el contenido con igualdad de espacios entre una y otra caja.

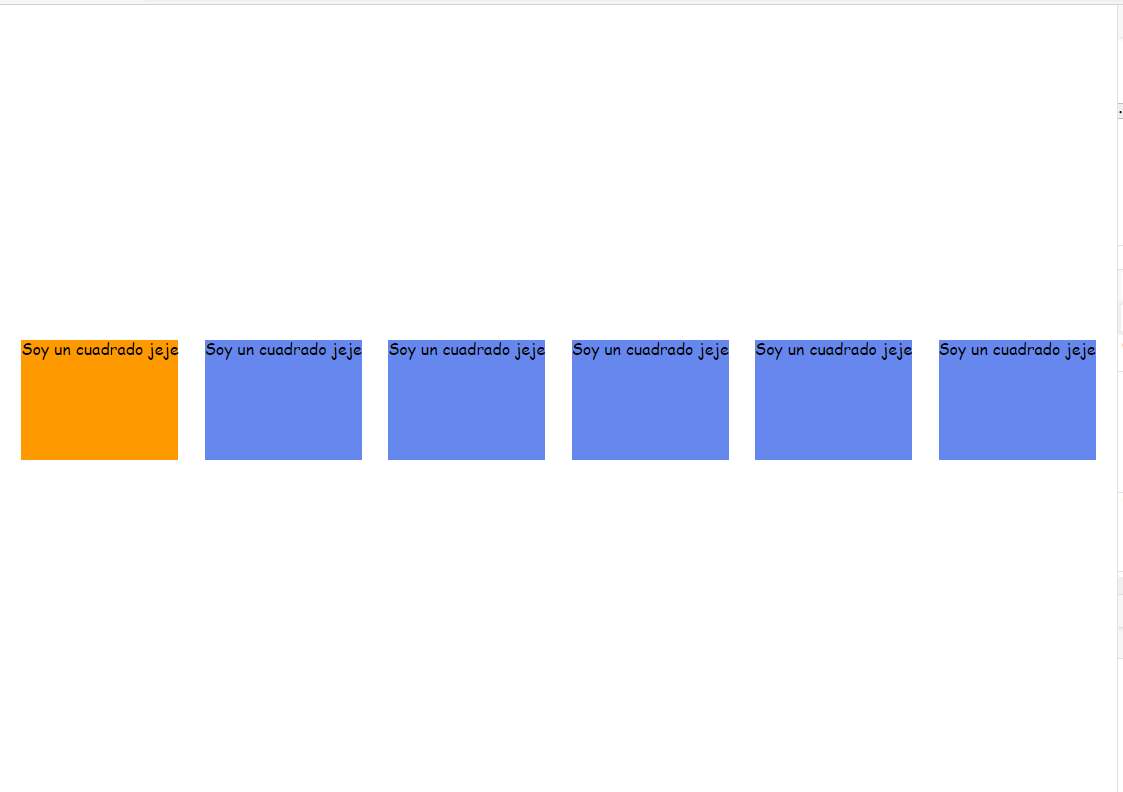
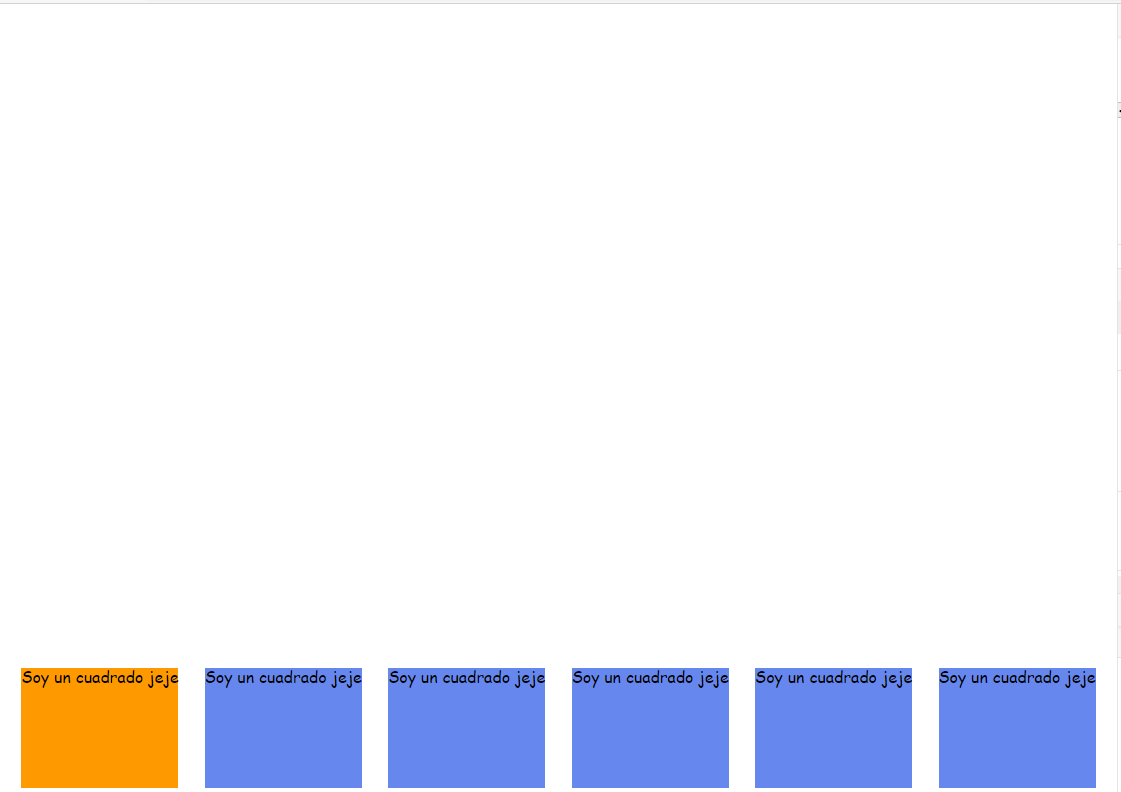


* Aling-items: Para alinear en el cross-axis cuando solo hay una línea.
* Aling-content: Para alinear en el cross-axis teniendo varias líneas.

Valores:

* Stretch: El valor por defecto.
* Flex-start: Posiciona las cajas en la parte superior del contenedor.
* Center: Posiciona las cajas en el centro del contenedor.
* Baseline: Se suele usar junto con el wrap-reverse y el Aling-content ya que es útil para ajustar las cajas al final del contenedor.
* Flex-end: Posiciona las cajas al final del contenedor.

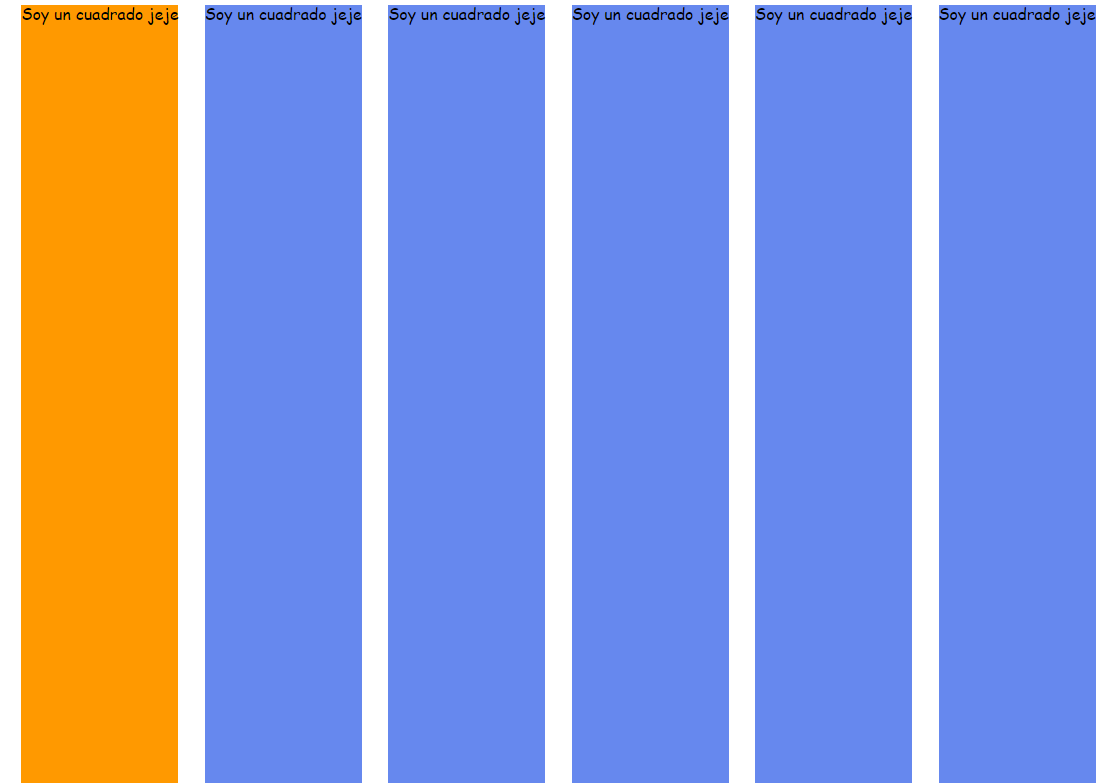
Flex-start Center flex-end

Diferencia entre flex-start y stretch:

Si los hijos de los contenedores no tuvieran height:

Flex-start Stretch

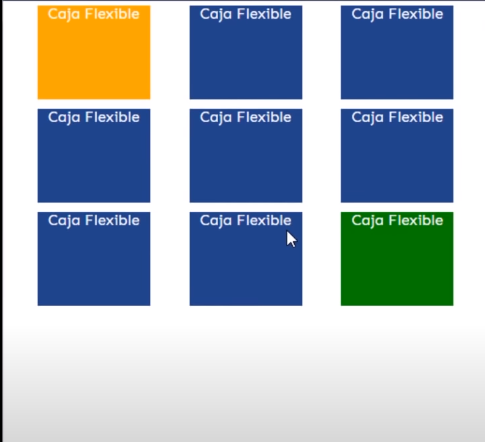
 

En flex-start se ajusta al contenido, mientras que en stretch se estira a lo ancho del contenedor.

Si hay varias líneas ya no se usa flex-items, sino Aling-content porque:

(con el wrap activado)

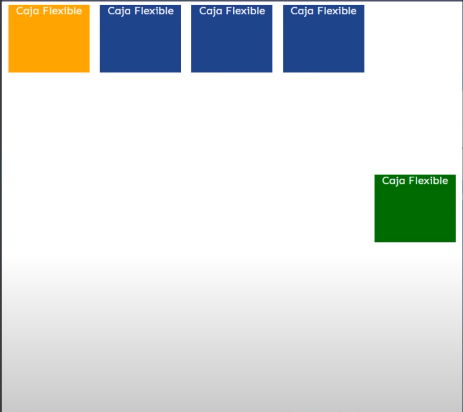
Aling-items: flex-start; Aling-content: flex-start;

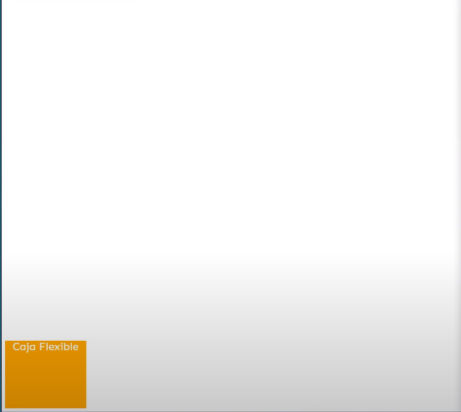
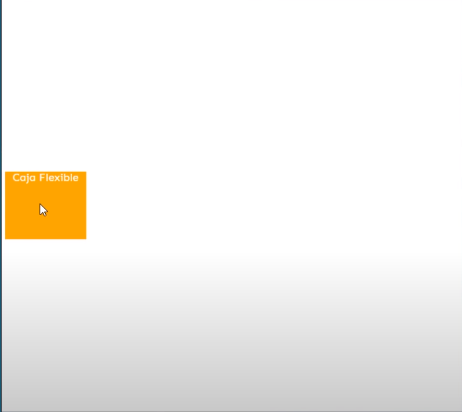
 

## Trabajando con flex-items

Se les da las propiedades a los items y solamente cuando el contenedor es flex. Estas propiedades son:

* Aling-self
* Margin
* Flex-grow
* Flex-shrink
* Flex-basis
* Flex (Abreviación)
* Order

Aling-self: Funciona en el cross axis (flex-start/center/flex-end)  


Margin: Funciona diferente en contenedores flex, lo que sucede es que va al lado contrario al que se le mencione en el margin auto en Top/Left/Right/Botton.  
   

Flex-grow: Reparte el espacio sobrante a las cajas flexibles que haya en el contenedor. Haciéndolas flexibles a lo ancho expandible (Width). Se puede definir el mínimo de contracción con min-width.

También se le puede dar flex-grow a una única caja, la que se llevará todo el espacio sobrante. Si se le quiere dar flex-grow a más de una caja hay que saber que el valor que se le dé define si se queda con más o menos parte sobrante.

Flex-basis: Ajusta al ancho como un width, pero este tiene una mayor preferencia (Especificidad)

Flex-shrink: A la caja que se le dé, esta cederá más espacio a las demás que tengan flex-grow o puede ceder menos que las demás con números menos que 1 (.9 .8 .7 .6 .5 etc..). Colocando 0 simplemente no cederá espacio.

Flex: Una propiedad que agrupa flex-grow, flex-shrink y flex-basis. Esta propiedad necesita al menos un parámetro.

Order: Como un z-index, pero no en el eje z sino en el main axis. Mayor preferencia al menor número.

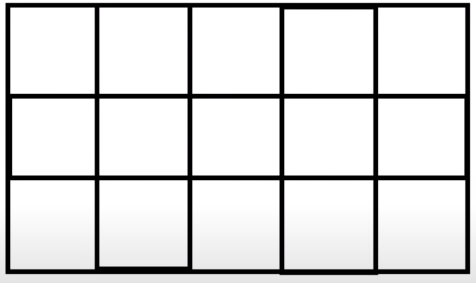
Color de fondo en degradado: , Donde:

El primer parámetro es el que define de dónde saldrá el degradado.

El segundo parámetro es el color con el que comenzará.

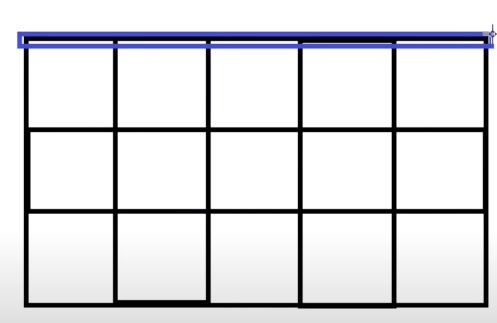
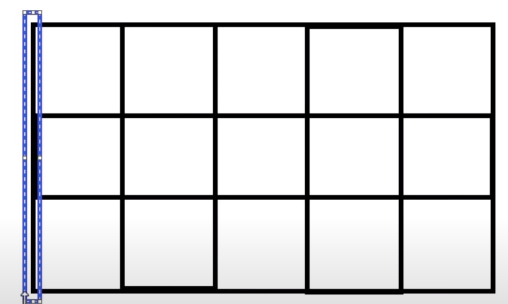
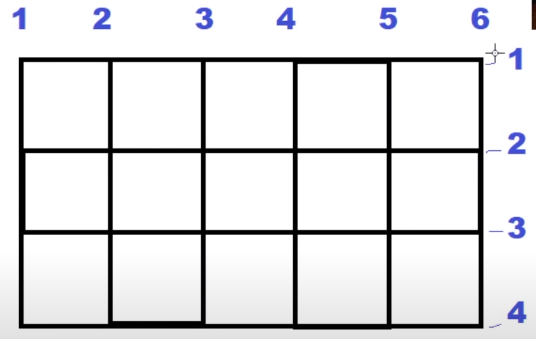
El tercer parámetro es el color con el que terminará.

## Grid - Conceptos y propiedades

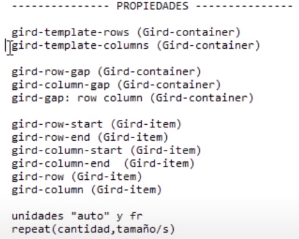
Grid es un estilo, el cual puede ser definido como un valor de la propiedad display. Se trabaja a través de grillas que se ven más o menos así:  


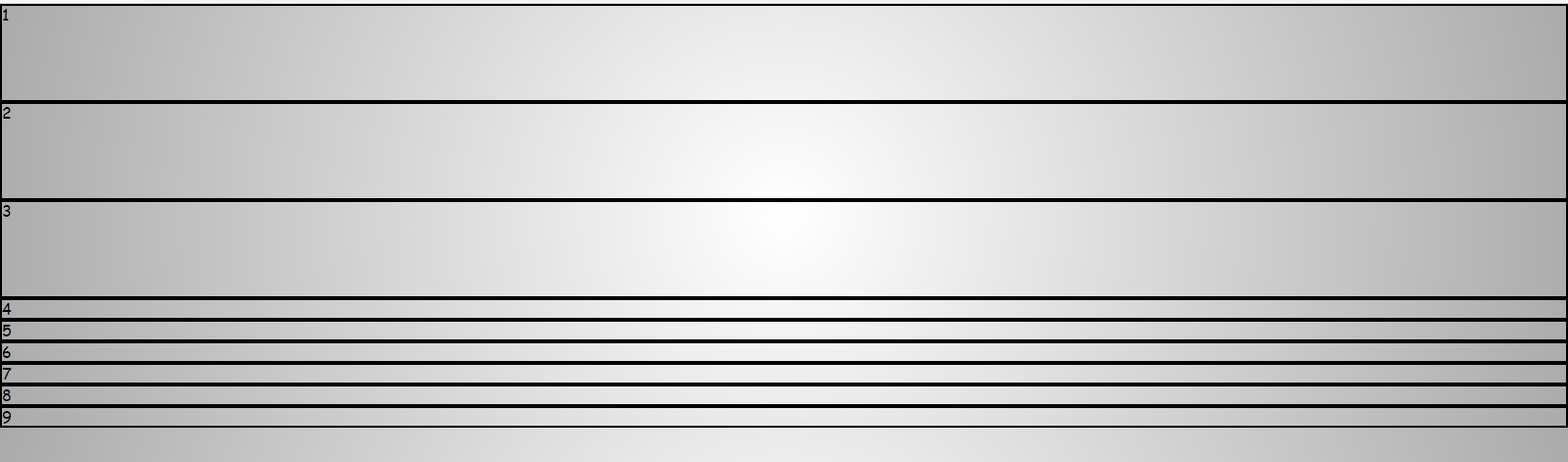
Funciona a través de celdas, columnas, áreas, tracks, rows, etc.

Propiedades:

* Grid container: El contenedor completo, todos los cuadros de la imagen. Se comporta como un bloque, pero cambia la estructura.
* Grid item: Todos los elementos del grid container son grid items (Puede haber excepciones). Son los hijos directos del grid container.
* Grid Cell: Cada celda, es decir cada cuadro de la imagen. Tienen cuatro lados y son paralelos.
* Grid tracks (Column y Row): Columnas y filas. Para saber cuantos grid tracks hay en total se suman los cuadros verticales y horizontales, en el caso de la imagen hay 8.
* Grid area: Áreas que se seleccionan y son consecutivas, es decir que no pueden estar en diagonal. Además, se definen por el programador.
* Grid line (Column Line y Row Line): Son las líneas que conforman el grid container, la primera Row Line y Column Line serían:  
    

Propiedades adicionales:

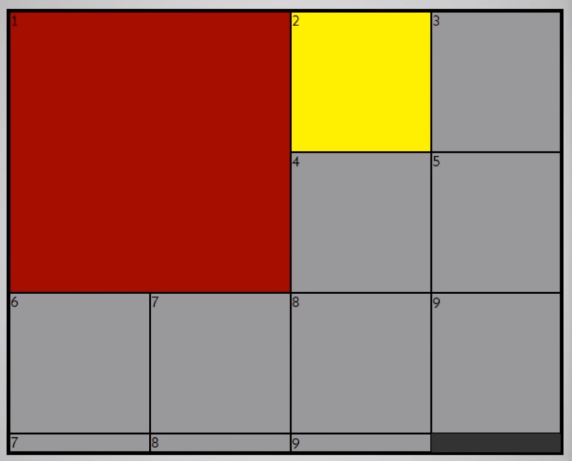
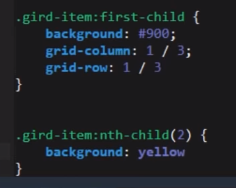


* grid-template-rows/column: Definir las filas o columnas del grid. Se definen 3 rows y el resto son grid implícito.  
  

En cuanto las unidades, la unidad fr se usa parecido a flex-grow en las cajas flex. Es decir que el espacio sobrante se lo añade a la columna o fila que se le ponga.

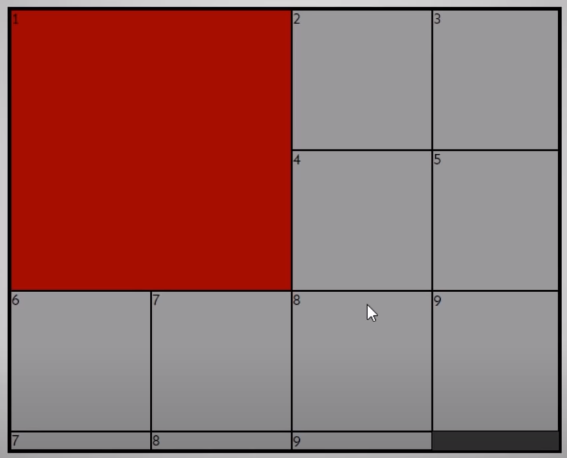
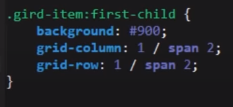
* Grid-row/column/ninguna-gap: Es como un margin, es la distancia que habrá entre una celda y otra.

Para grid items:

* Grid-row/column-start
* Grid-row/column-end
* Las dos anteriores se pueden abreviar con grid-row/column: Se utiliza para que las celtas ocupen más tamaño  
   .

Para seleccionar el primer elemento es first-child y con los otros sería nth-child(#).

Otra forma de realizar la selección de espacio es con , pero sería escribir más código.

Con span se puede decir, arranca desde la n fila/columna y ocupa de aquí en delante n filas/columnas.   
 

Repeat(): Se usa para resumir la creación de filas y columnas. En vez de escribir esto:

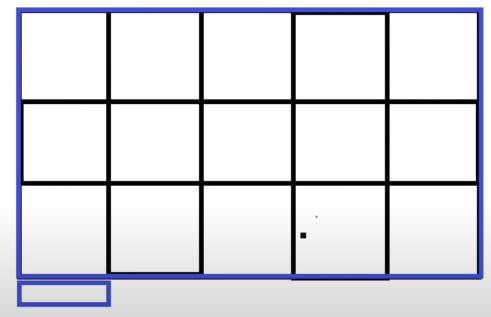


Se escribiría esto:

 El primer parámetro indica las veces que se debe repetir lo que está en el segundo parámetro.

## Grid implicito y grid explicito

Si se tiene una grilla con 3 filas y 5 columnas sería así:

 Y esa celda sobrante vendría siendo el grid implícito.

 Las celdas con grid implícito suelen provocar celdas que no se llenan.

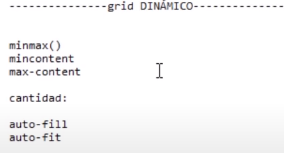
Por lo que se tienen propiedades con las que se puede trabajar el grid implícito.  

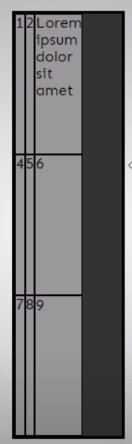
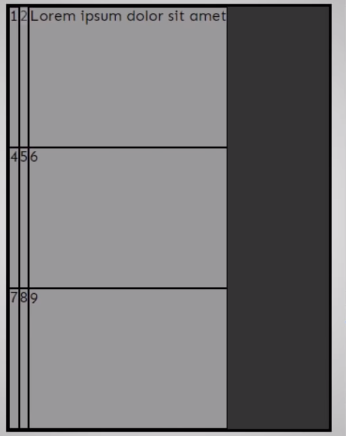

* Dense: Valor del grid-auto-flow, lo que hace es rellenar un espacio en blanco con la celda más cercana.
* Column: También se puede usar en grid-auto-flow para que lo implícito se comporte como columna.

## Grid dinámico

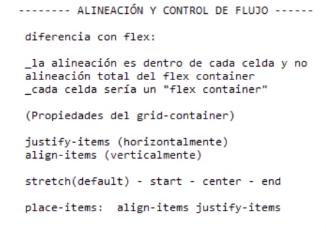
Se trabajan con propiedades, cantidades y estructuras dinámicas

Se tienen las siguientes propiedades:



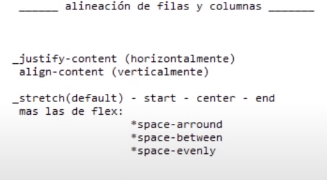
* Min-content y max-content: Las celdas de la grilla se ajustan al contenido, ya sea al mínimo o al máximo  
  Mínimo Máximo  
   .
* Minmax(): Para definir cuanto es lo mínimo que puede contraerse o estirarse las celdas. . Además, se puede juntar con el min-content y max-content, definiendo que lo mínimo y máximo será el contenido.
* Auto-fill: Llena automáticamente el espacio de acuerdo a las dimensiones puestas , por lo que si hay espacio para poner otra columna de 120px lo hará, sino llenará el espacio hasta que se pueda.
* Auto-fit: Lo que hace es escalar los elementos cuando son expandidos (también se le pueden adicionar el Minmax()) 

## Alineación en Grid



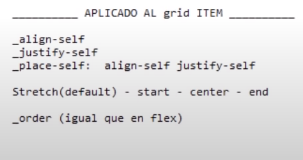
Con justify-items se puede organizar el contenido en la pantalla **hacia los costados** con los valores start, center y end.

Para organizar verticalmente se usa Aling-items y sirve con los valores ya vistos.

Las propiedades anteriores organizan items, para organizar columnas y filas se usa:  
Además, se pueden usar los mismos valores de flex.

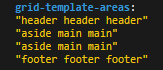
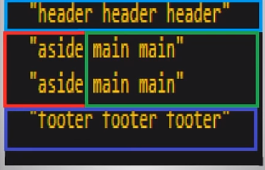
Justify-content: Para los costados.

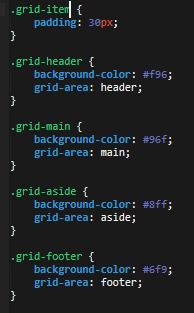
Aling-content: Para organizar verticalmente.

En cuanto a los Grid items se usan las siguientes propiedades, se moverán en su contenedor grid  


* Aling-self/justify-self: Se ajusta individualmente al contenido como se le defina ya sea horizontal o verticalmente.
* Place-self: Para escribir menos líneas de código se utiliza esta propiedad. El primer parámetro sería para organizar en vertical y el segundo para horizontal.
* Order: Funciona para organizar de acuerdo al valor asignado, da la preferencia de menor a mayor.

## Grid Areas

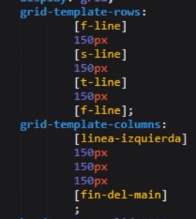
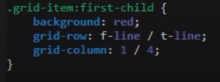
Se define como un conjunto **consecutivo** de celdas. Se define por áreas como si fuesen nombres. De la siguiente manera:  
 Y lo que sucede en la página web es:

Y para que se posiciones de acuerdo a lo escogido se les pone la propiedad grid-area:  


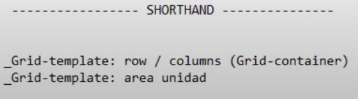


## Nombrar Grid-line

Funciona para tener más control y no andar escribiendo parámetros como 2 / 4

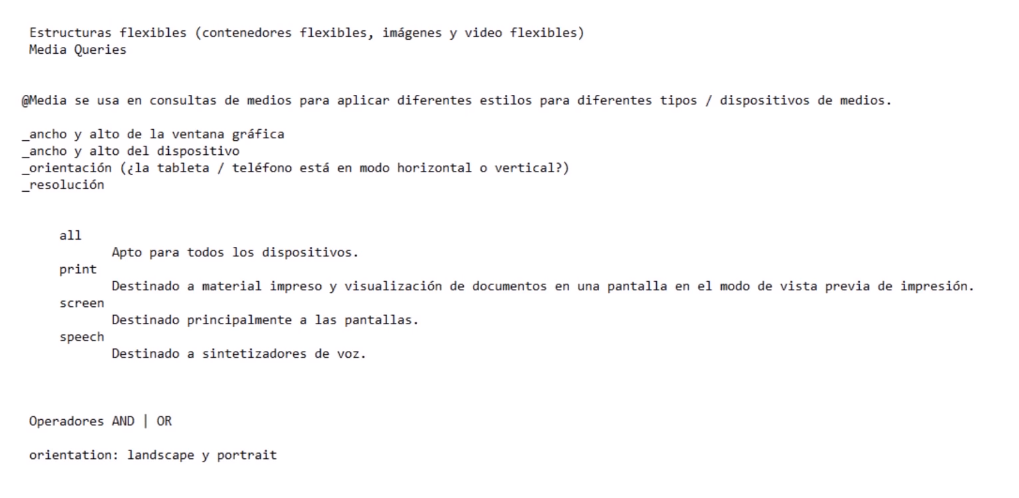
 

## Grid shorthand

 Sirven para acortar, en este caso:

* Gird-template: Acorta tres propiedades en una, es decir, unidades, filas y columnas.

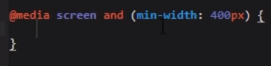
## Responsive Design – Tema full

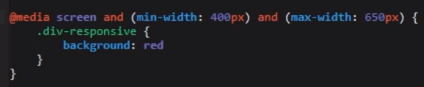


Existen dos metodologías.

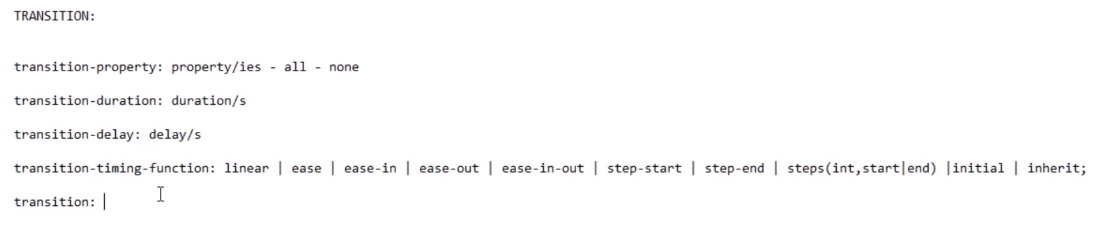
* Mobile first: Diseñar el sitio web primero para dispositivos móviles y después a resoluciones mayores como tablet y escritorio. (Recomendado)
* Desktop first: Diseñar el sitio web primero para escritorio, después tablet y por último para móvil

Para crear una regla se usa: @media all/print/screen/speech and (condicional)  
Por ejemplo:

 En la pantalla si baja más del mínimo (400px) sucede lo que está entre corchetes. También se pueden poner varias condiciones con AND o OR



## Transition



Una propiedad que permite realizar transiciones dentro de los elementos (mostrando el proceso de un cambio). La propiedad abreviada es Transition, pero hay otras como:

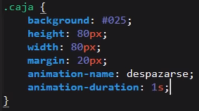
* transition-propiety: Define a que se le va a hacer la transición, all no se recomienda porque lo que se busca es ahorrar recursos. Puede definir varias transiciones teniendo distintos valores:  
  
* transition-duration: El tiempo que va a tardar en realizar el cambio de la transición.
* transition-delay: El tiempo que tarda en iniciar el cambio de la transición.
* transiton-timing-function: El tipo de movimiento que se quiera tener en la transición (Lineal, acelerada al principio, acelerada al final, etc.)

Se le tiene que dar al elemento por lo menos las dos primeras propiedades y un valor a ellas para que funcione. Se aplica a cualquier cambio del elemento.

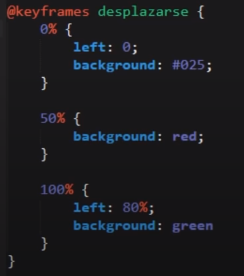
## Animaciones

Son parecidas a las transiciones solo que necesitan la regla @keyframes junto con el nombre de la animación, así se definen las animaciones.



En from son las propiedades que se tienen al inicio y to son las que se tienen al final. Y con la propiedad animation-name: nombre-de-la-animacion se le da la animación al elemento, además animation-duration: tiempo para definir cuanto tarda en hacer la animación.

También se puede trabajar con porcentajes para tener constantes cambios en la animación.



Con animation-iteration-count se definen las veces que se repetirá la animación.

animation-direction:

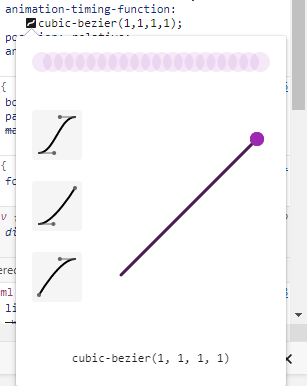
* Normal: De 0 a 100, como por defecto.
* Reverse: De 100 a 0, invertido.
* Alternate: De 0 a 100 y de 100 a 0, va y viene.
* Alternate reverse: De 100 a 0 y de 0 a 100.

animation-fill-mode:

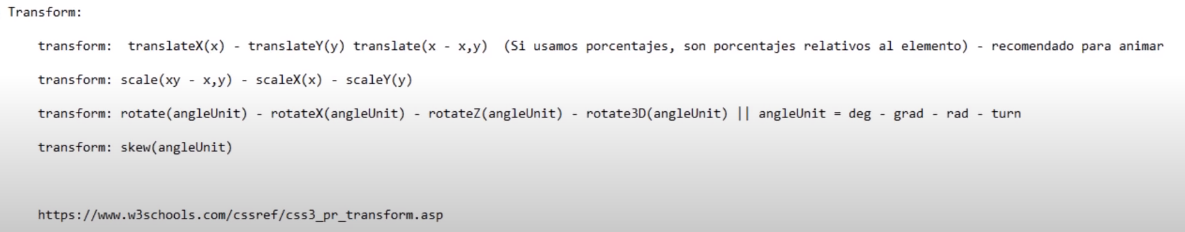
* Forwards: Se queda en la última posición al terminar la animación.
* Both: La animación arranca como se le indica al inicio de la animación (color, etc)

## Cubic bezier

Curva que define la aceleración en la transición de un objeto, se define de la siguiente forma: .

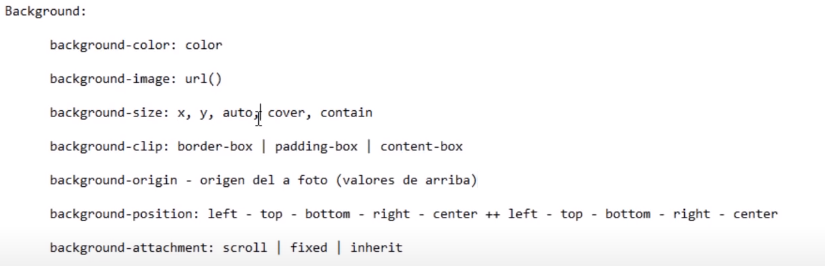
Y se vería asi:  
 No se recomienda hacerlo manualmente, para ello hay generadores en internet o herramientas del inspeccionar del google.

## Transform



* Translatex(): Mueve el elemento en el eje x
* Translatey(): Mueve el elemento en el eje y
* Translate(): Primer valor eje x segundo eje y 
* Scale(): Aumenta el tamaño del elemento 0.5 = mitad del tamaño  
  1 = Tamaño original, 2 = Doble de grande, 3 = triple de grande.
* Scalex(): Estira el elemento en el eje x
* Scaley(): Estira el elemento ene l eje y
* Skew(deg/rad): Gira el elemento, aunque es mas útil usar clip path
* Clip-path: La forma que deseo que tenga el elemento [Clippy — CSS clip-path maker (bennettfeely.com)](https://bennettfeely.com/clippy/)

## Background



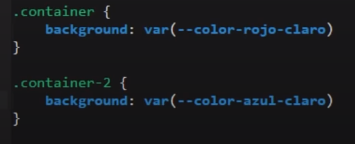
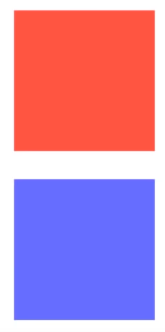
* Background-image: url(link): Usar una imagen como fondo.
* Background-size: Para acomodar la imagen que se usó como fondo.
* Background-repeat: Si se repite la imagen cuando tenga Cover en size
* Background-clip: Desde donde tiene que mostrar la imagen (recortándola)
* Background-origin: Desde donde se origina la imagen (Sin recortarla)
* Background-position: Ubica el fondo con 2 valores, el primero horizontal y el segundo vertical.
* Background-attachment: Scroll = Simula que es parte del fondo. Fixed = Se desplaza junto al contenido, pero mientras se queda fijo.

## Variables

Son espacios en memoria para su posterior uso.

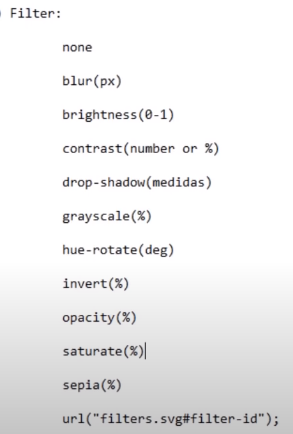
* Globales: Cualquier elemento puede hacer uso de ellas.
* Locales: Solo los elementos definidos hacen uso de ellas.

Para definir una global se usa :root { --nombre: valor; }  
 Y para usarla es con el parámetro var(nombre)

Para crear variables locales, se crean dentro del elemento que se desea usar.  
 Solo se crearía para las etiquetas p

## Filter

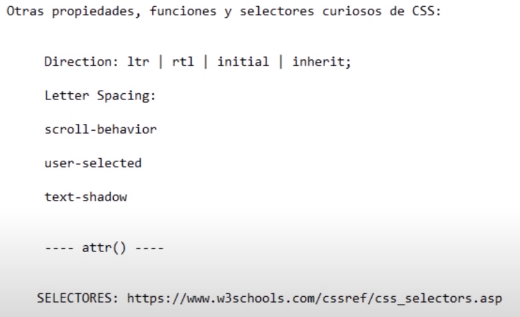


Para trabajar con filtros, color, exposición, brillo, escala de grises, etc.

Se usa la propiedad filter y se va probando valor por valor.

* None: Por defecto.
* Blur(px): desenfocarlo.
* Brightness(0 nada, 1 normal, # aumenta): Para aumentar o disminuir el brillo.
* Contrast(0 nada, 1 normal, # aumenta): Aumentar o disminuir el contraste.
* Drop-shadow(): Genera una sombra en los bordes de las imágenes png.
* Grayscale(0-1): Definir la escala de grises.
* Hue-rotate(deg): Para girar la gama de colores.
* Invert: inverte los colores.
* Opacity(0-1): Opacidad de la imagen.
* Saturate(%): Satura la imagen para tener colores cercanos a los puros.
* Sepia(0-1): Convierte en un color sepia.

## Otras propiedades



Direction: Parecido al text-aling, es la dirección de lectura.

Letter-Spacing: El espacio entre letra y letra

Scroll-behavior: Con el valor smooth realiza un efecto de desplazado lento en la parte de la página web que se le indique (con un <a>)

User-selected: Con el valor none, no le permite seleccionar al usuario.

Text-shadow: Sombra alrededor del texto.

## Sigue aprendiendo -