**CSS**

**Selectores**: El elemento al que quieres aplicar el estilo. Identifican al elemento al que se le aplicará el CSS.

**NOTA: Por default, cada explorador tiene su propia hoja de estilos, es por eso que al una app sin agregar estilos, elementos como H1, <p>, etc, ya tienen sus estilos, es decir, tamaña de fuente, margen, etc. Y puede variar entre cada explorador. Sin embargo, es posible sobreescribirlos utilizando CSS para que la webapp sea vista de igual manera en todos los exploradores.**

**normaliza.css: L**ibrería que busca normalizar. Estandarizar que todos los h1 deben ser ma´s grandes, por ejemplo. La puedes descargar y guardar como un css, y llamarla a tu proyecto. Lo que hace es que da valores por default a los selectores de etiqueta, por ejemplo a Body le quita el margen, tamaños de salto de linea etc, para que tu definas cada uno de los estilos como mejor deseas.

**Clase (class): Atributo de un elemento de html o xml.**

Nos ayudan a identificar los elementos para agregarles estilos. Es lo más recomendable en vez de usar Id. Si utilizas Id para llamar una clase, en nuestro archivo css se deberá escribir en el nombre de la clase de la siguiente manera: #estilo (en vez de .estilo). Pero no se recomienda usar id para casarlos con estilos.

**Clase 2**

**Selector Universal.**

Con este selector, el estilo que definamos se aplicara en todos los elementos del html sin excepción.

Sintaxis:

\*{

color: red;

}

Hara que todos los elementos tengan letra de color rojo, y no se tiene que llamar en ningun elemento, se aplicará por default.

**Case sentive & Case unsensitive**

Los selectores son case unsensitive, es decir, no afecta si escribes mayúsculas o minúsculas al aplicar estilos, por ejemplo, si en tu archivo css escribes el selector DIV (en lugar de div), se aplicará el estilo. Sin embargo cuando creamos una clase/atributo, por ejemplo .miEstilo, para que se ejecute debe ser escrito de igual manera en el elemento html.

**Selectores compuestos**

En realidad no se llaman así, pero con ellos puedes aplicar un estilo a dos o más clases, ejemplo:

.titulo, .parrafo{

color: red

}

Lo que hará este estilo es que a los elementos que tengan la clase titulo y párrafo, asignará el color de fuente a rojo

**Selector Hijo Directo**

Si tenemos ciertos elementos, como por ejemplo una lista o menú como la siguiente:

<ul>

<li>Opción 1</li>

<li>Opción 2</li>

<li>Opción 3</li>

<li>Opción 4

<ul>

<li>Opción 4.1</li>

<li>Opción 4.2</li>

<li>Opción 4.3</li>

</ul>

</li>

</ul>

Y solo queremos ocultar los submenú de la opción 4, podemos realizarlo aplicando lo siguiente:

ul > li > ul{

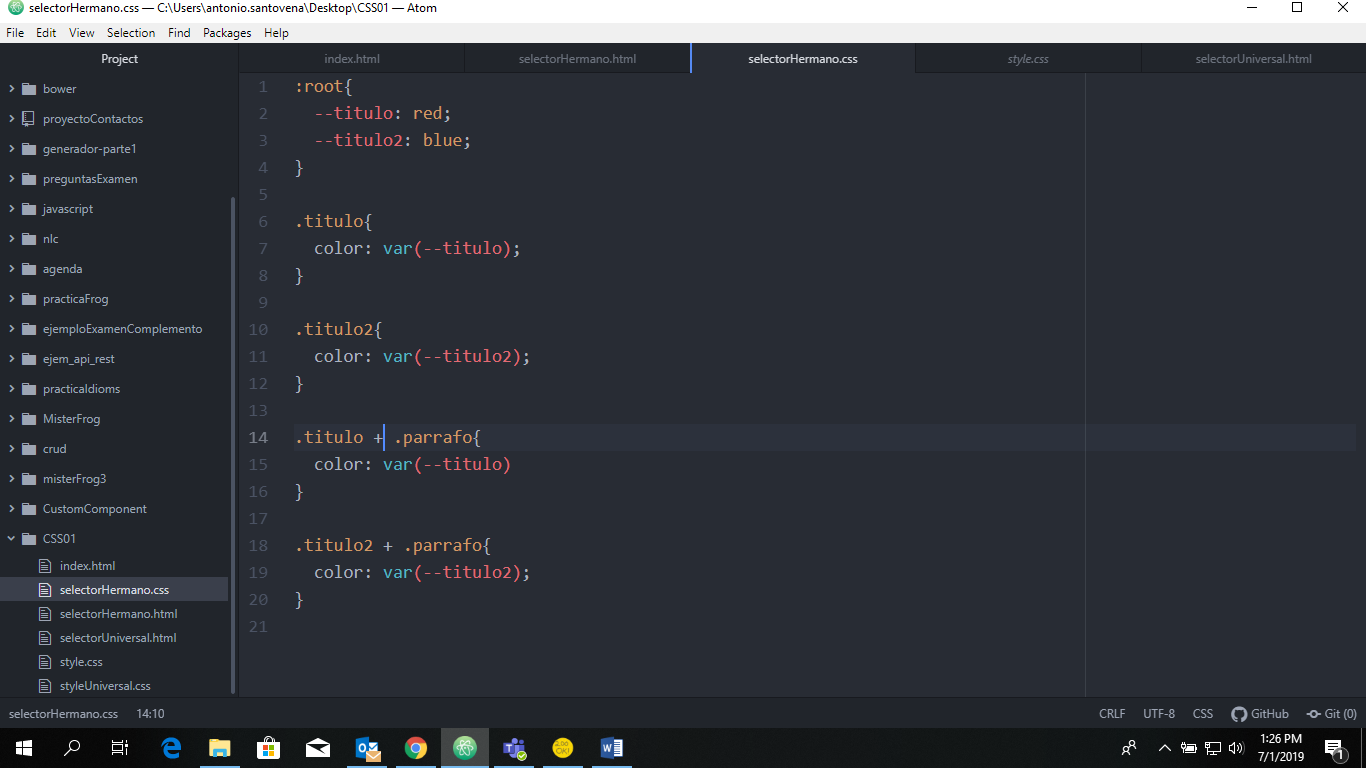
display: none,

}

Estamos indicando que el ul que es hijo de li, y li que es hijo de ul. También podemos aplicarlo con clases definidas.

**Selector hermano siguiente**

De esta manera podemos asignar un estilo a un elemento y al elemento que sigue. Por ejemplo, si tenemos un elemento h1, y tiene su elemento p, y queremos asignar el mismo estilo de color, podemos realizar los siguiente:

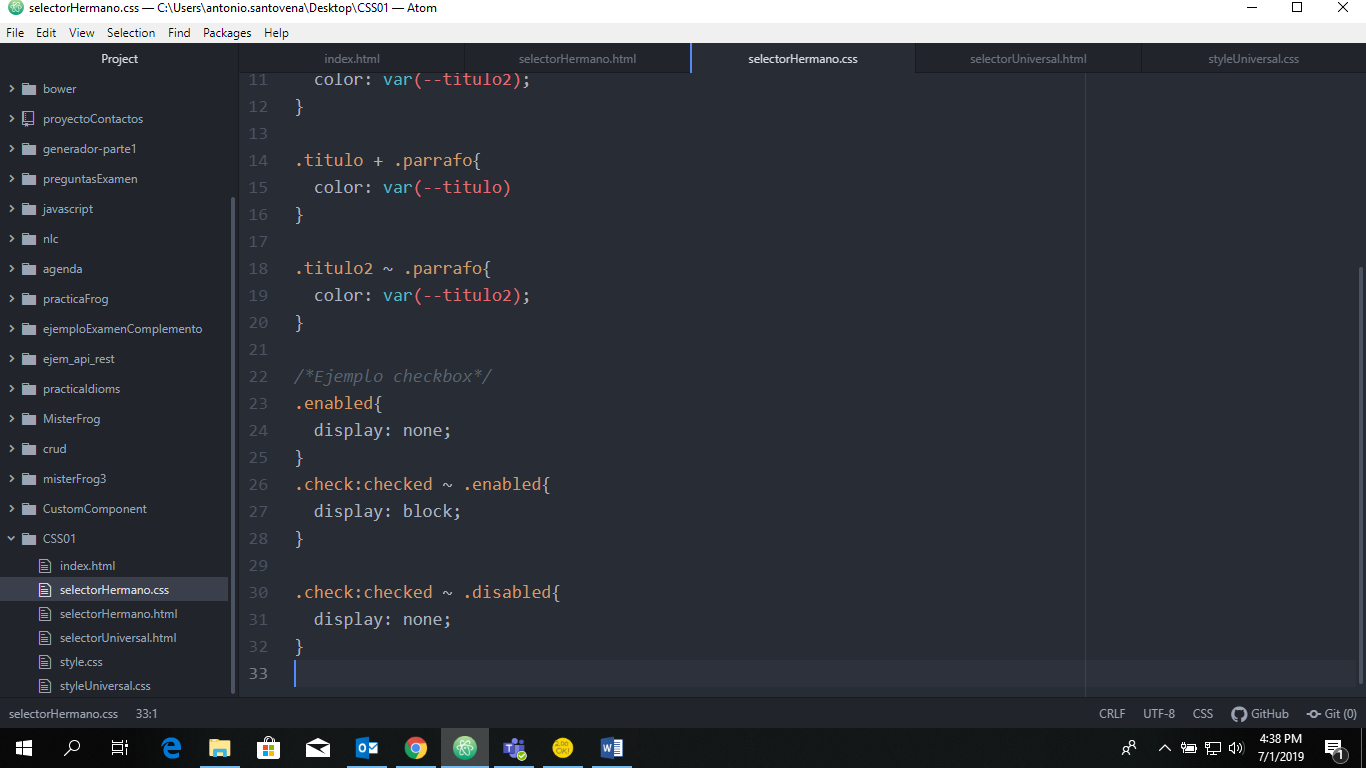


Lo que estamos haciendo es definir en primer lugar dos variables, que contienen los colores red y blue respectivamente. Después asignamos el color rojo al elemento que tendrá la clase titulo, y blue al h1 que tendrá la clase titulo2. Supongamos que debajo de cada uno de estos h1 hay un <p>, entonces si queremos que mantengan el mismo color que sus respectivos h1, lo hacemos sumando la clase de título con la de párrafo. Esto solo afectará al siguiente elemento que contenga la clase, en este caso párrafo. Si hay más elementos con dicha clase no se verán afectados.

**Hermanos Siguientes.**

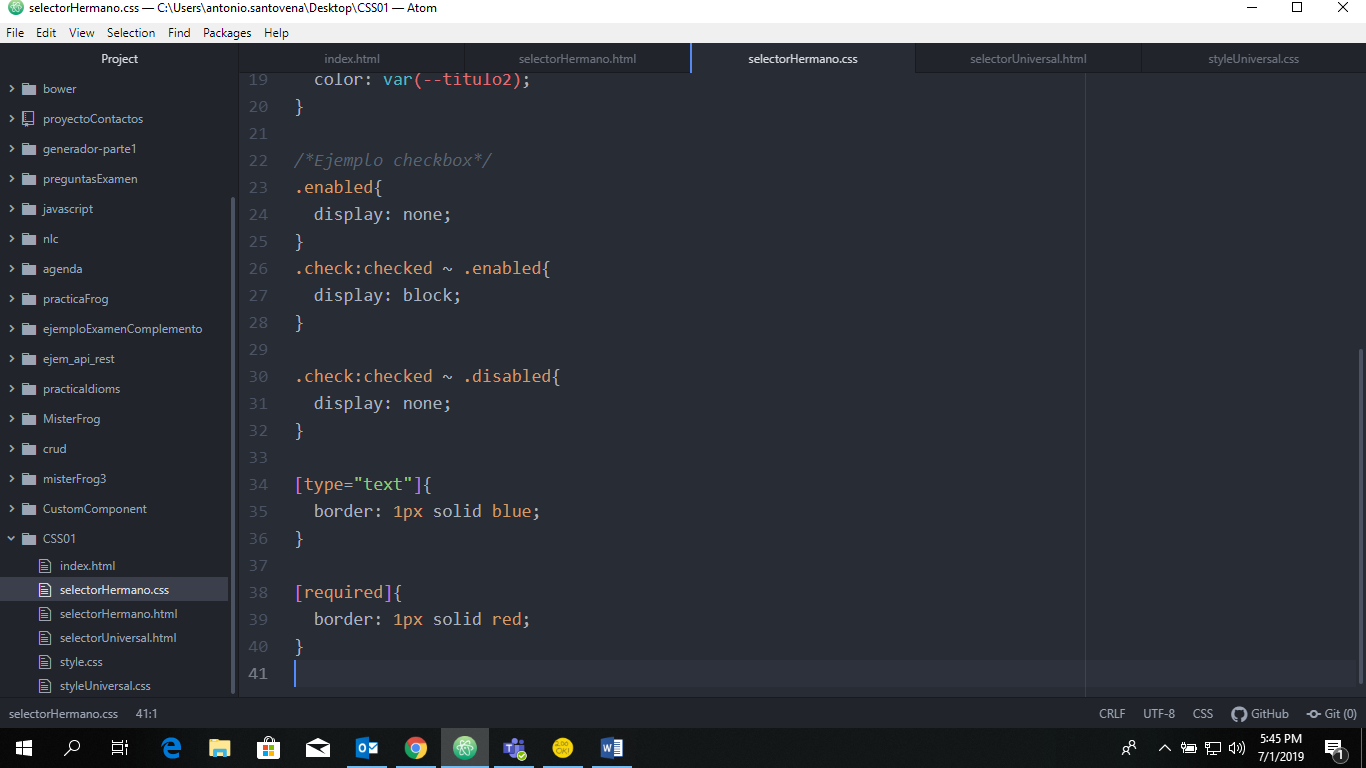
Al definir una clase hermano siguiente directo, el estilo solo afectará al elemento que se encuentra en seguida del elemento seleccionado. Hermano siguientes, te permite afectar no solo al alemente que sigue, si no a todos los elementos que tengan la clase definida para la asignación del estilo. Para que funcione, en lugar de colocar el signo de +, se colocará ~

Ejemplo:



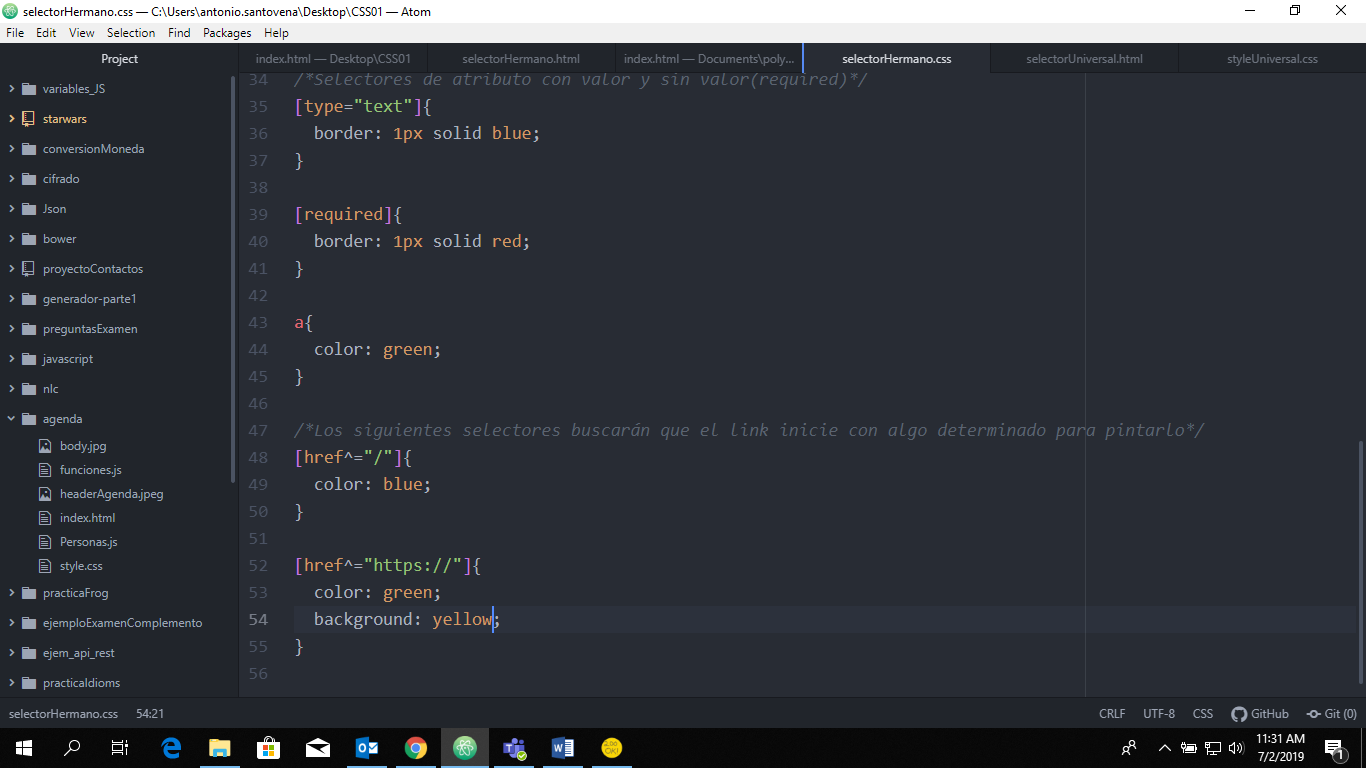
**Selector de atributo**

El selector de atributo te permite asignar estilos por medio del atributo. Puede ser atributo con valor p únicamente el atributo como en los siguiente ejemplos:



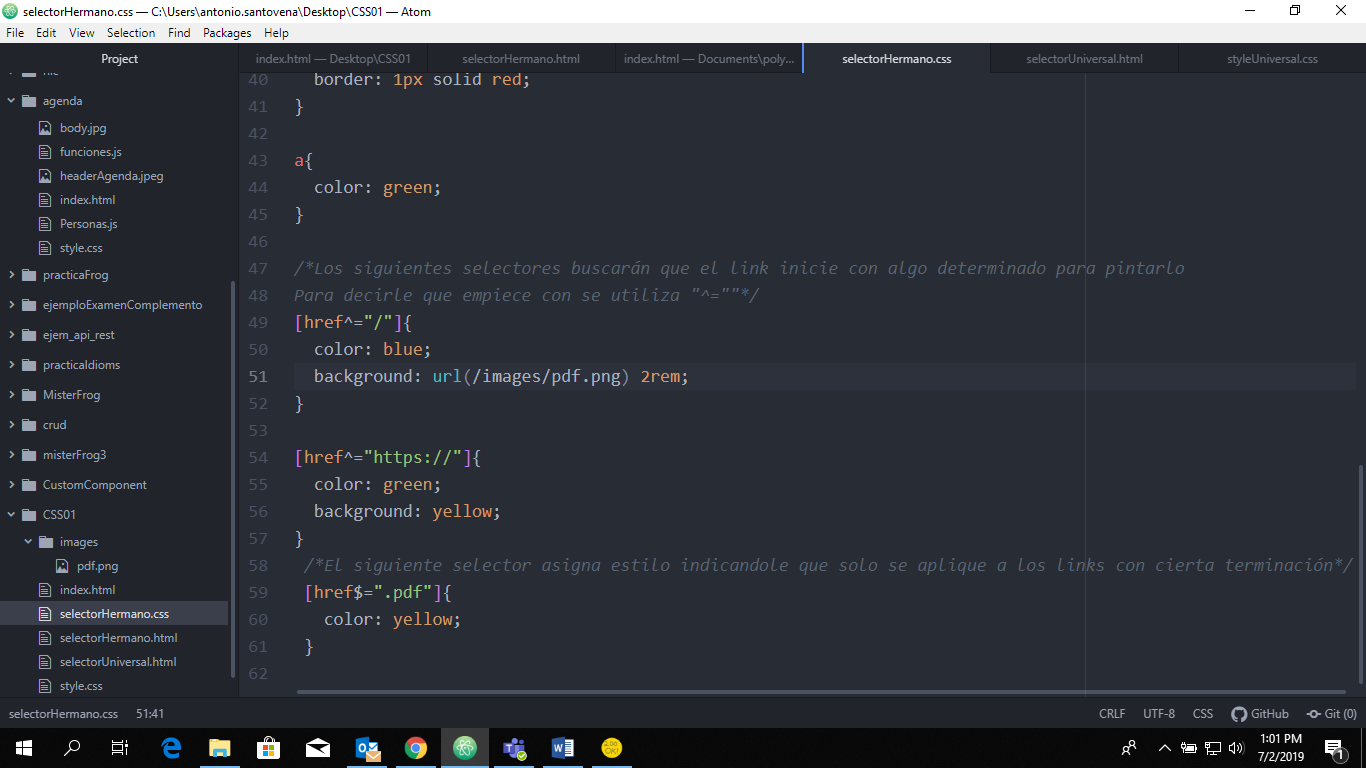
**Selector de atributo comienza con..**

Un ejemplo para poder utilizar este selector es cuando utilizamos links. Por ejemplo, si tenemos dos tipos de link, uno local que nos dirigirá a una página de nuestro proyecto, y un link externo que nos sacará del proyecto, y queremos pintar cada uno de diferente color:



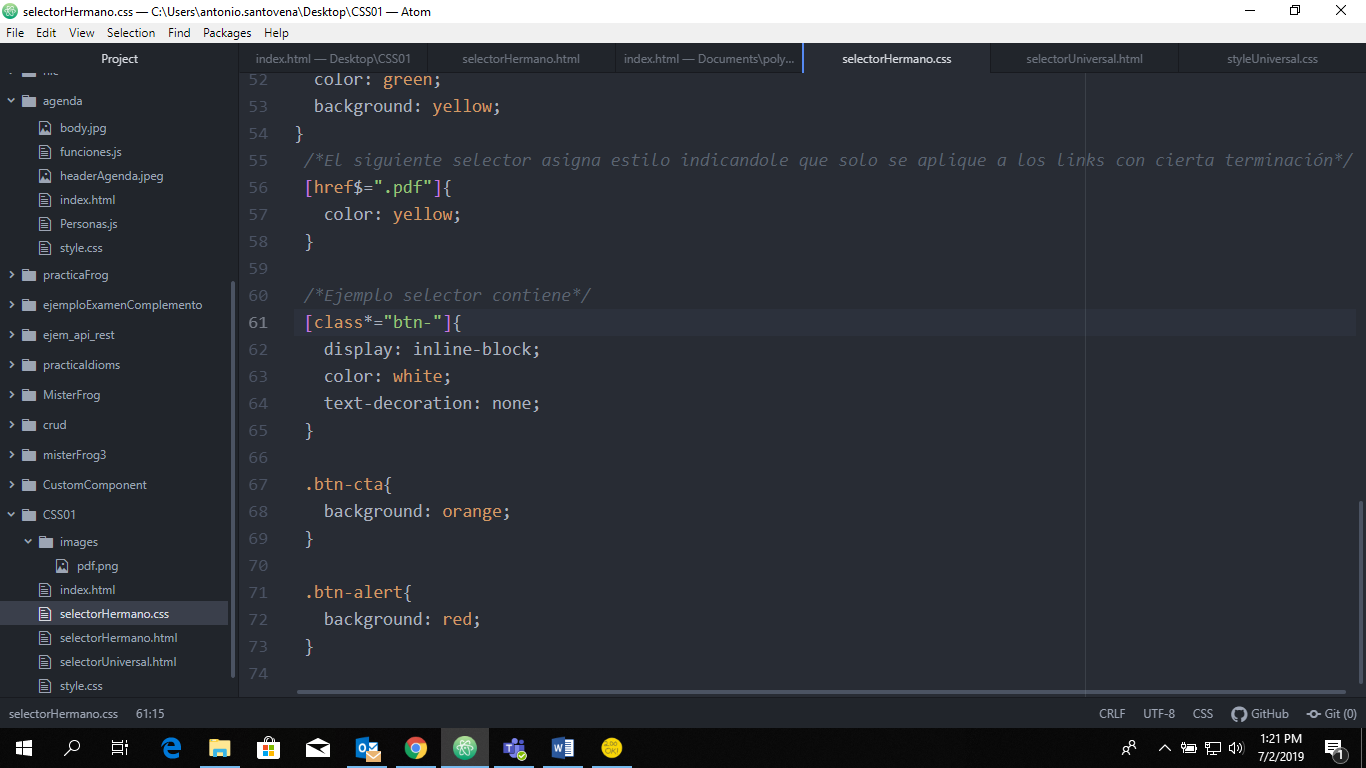
**Selector de atributo termina con…..**

Similar a la acción anterior de “Selector de atributo comienza con…”, pero ahora busca la terminación del link, para ello, en lugar de utilizar ^= se utiliza $=.



**Selector de atributo contiene…**

**Busca en el link una cadena determinada, por ejemplo algo de la clase:**



**Clase 3.- Especifidad, Herencia y Cascada**

**CSS Maneja arquitecturas: SMACSS, OOCSS, ITCSS (Se enfoca en la especifidad)**

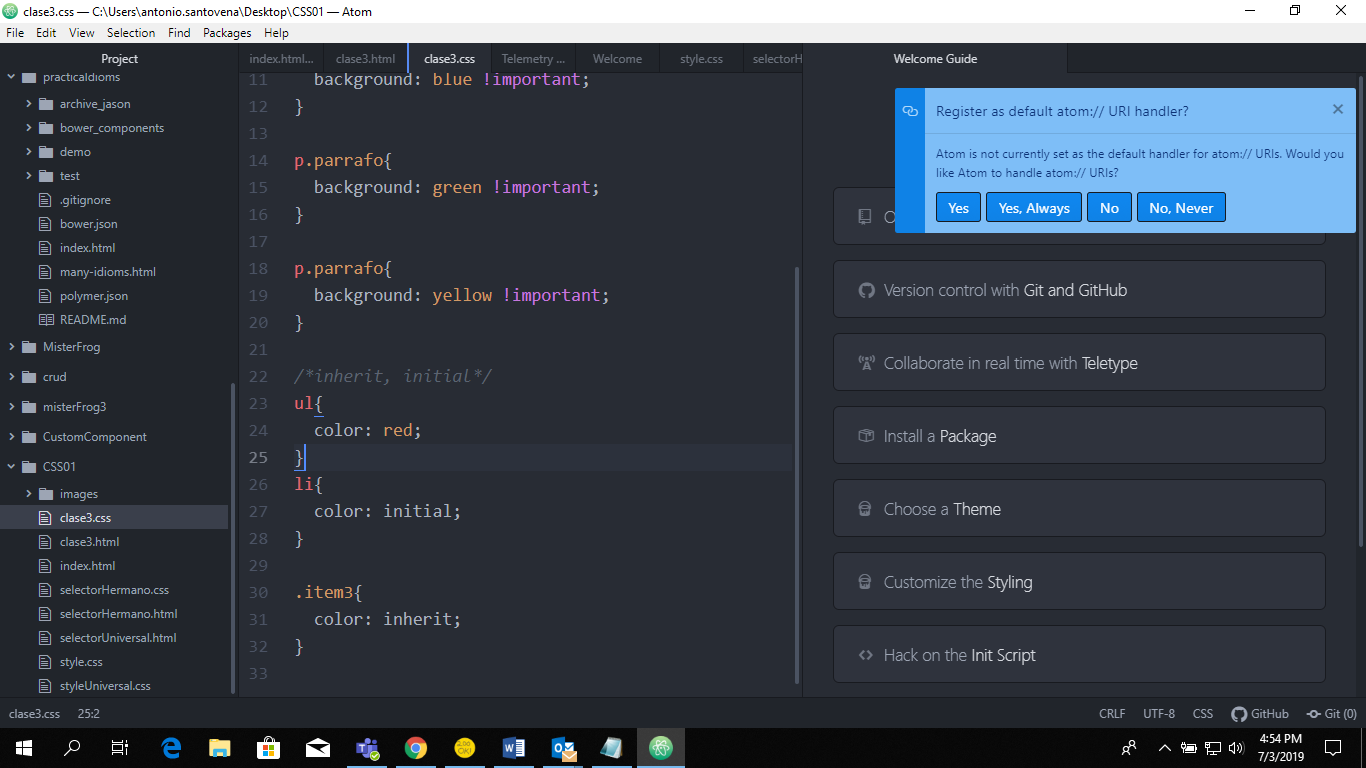
**Especifidad**: Cuando un navegador abre una pagina web, carga todos los estilos de los elementos, pero hay ocaciones en que un mismo elemento tiene asigando más de un estilo, es decir, puede que tu manualmente le hayas asignado al título un colos, pero después en otra línea lo cambiaste, y probablemente estes utilizando una librería que también afecta el título. Entonces la especifidad es decidir cual estilo asignarle al elemento. Para ellos hay ciertas reglas que siguen los exploradores, verificando la prioridad del elemento y estilo.

**Cascada:**

Lo que viene después sobre-escribe a lo de antes. Es decir, si definimos que una clase me pintará un <p> de rojo, y después escribimos que esa <p> se pintará de azul, tomará el ultimo estilo asignado, es decir, se pintará de azul sin ningún problema. Sin embargo, esto funcionara siempre y cuando respete la ESPECIFICIDAD, es decir, si el estilo asignado antes tiene mayor especificidad, entonces no tomará el estilo de abajo, por más que hasta utilicemos ¡important.

**Keyword inherit, initial**

Son keywprd que nos permiten definir si queremos heredar los estilos o conservar los iniciales de los elementos. Por ejemplo:



**Clase 4 Box Model**

**Layout:**

Geometría de los elementos que define como, con qué tamaño, con que separación respecto a los elementos adyacentes y en qué posición se van a dibujar en la pantalla.

Width y heigth, propiedades para definir el ancho y alto de un elemento.

**Elementos inline y de bloque**

**Todos los elementos son cajas.**

**Inline.**

Elementos que no crean nuevas líneas. Elementos que se colocan enseguida de otro, en la misma línea.

Tienen un ancho pero no alto. Es decir, por default tienen un ancho y en alto, pero no pueden ser modificables, por ejemplo un elemento spam. No puedes agrandarlo más de su contenido.   
Ejemplos: <p><small>

Se ajustan al contenido

Para convertir un elemento block a inline: display: inline;

**Bloque**

Crea una línea para cada elemento. Ejemplo: h1;(Puedes modificar su heigth y width que tiene por default).

Para convertir un elemento inline a block: display: block;

**Inline-block;**

Permite convertir un elemento a inline y block a la vez, es decir, un elemento de una sola línea pero con la cualidad de poder modificar su heigth y width.

**Box Model**

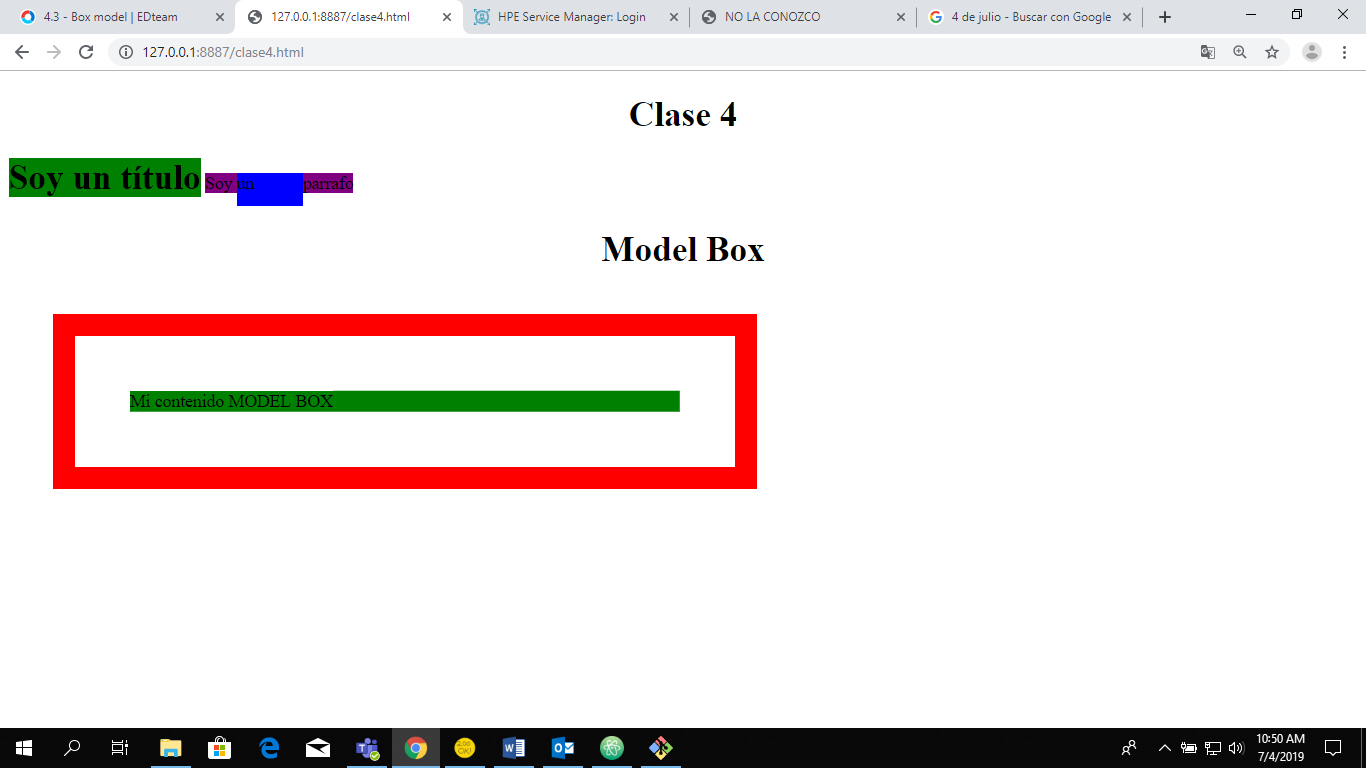
Algoritmo a través del cual el navegador dibuja las cajas en pantalla. Hay tras cajas:

* Caja de contenido (Content box)
* Caja del Padding (Padding box)
* Caja del Border (Border box)

Padding box

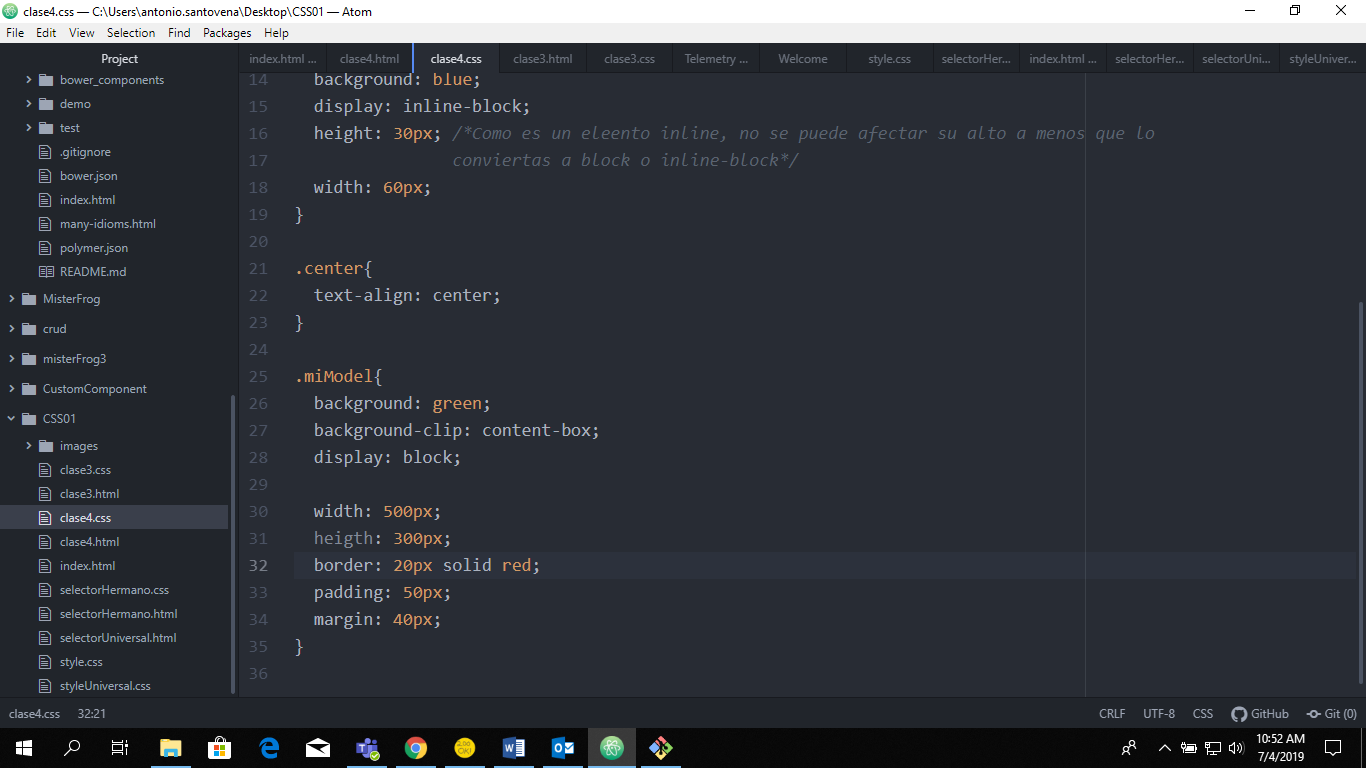
Content box

* Margen (Caja invisible alrededor)



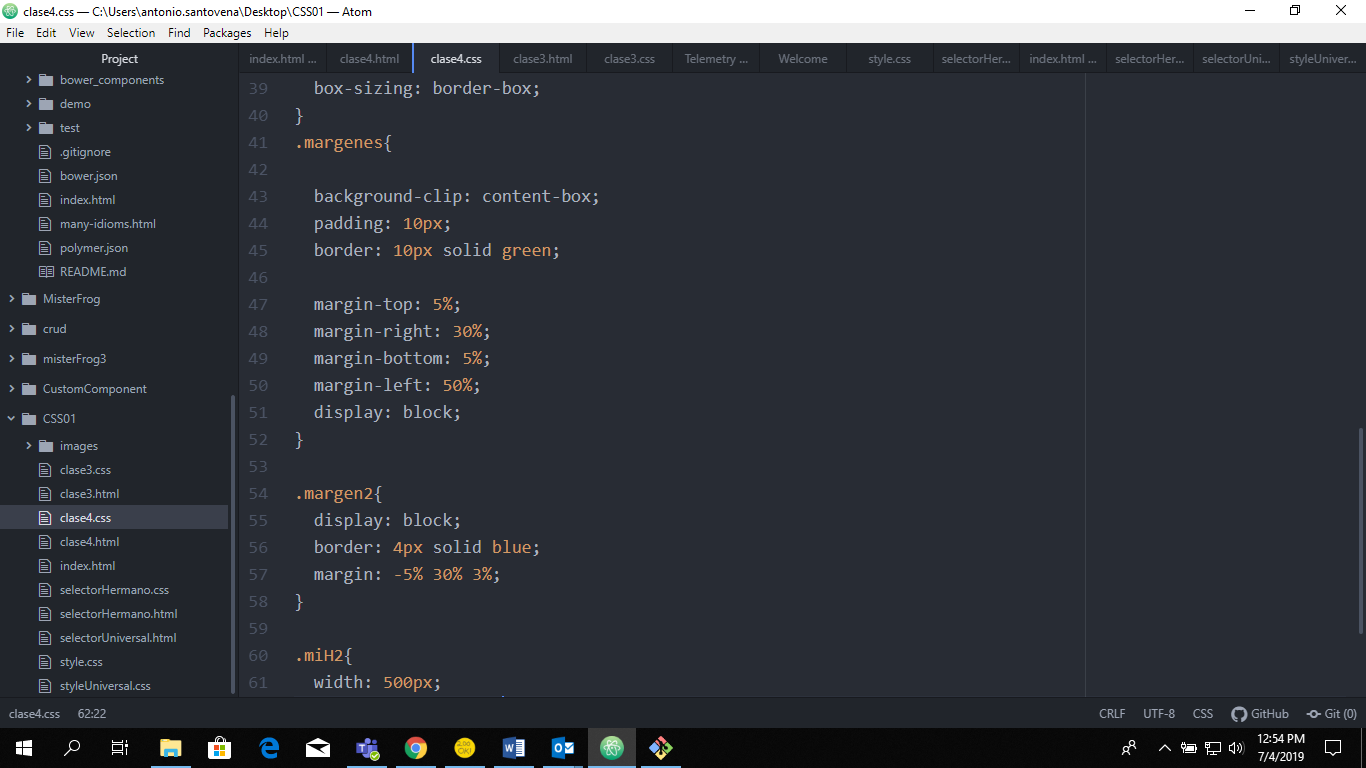
margin box

border box



**Margin**

Existen varias formas de aplicar un margen a nuestros elementos. La más utilizada es:



Sin embargo, podemos también colocar el margen en una sola línea:

margin: 5% 30% 5% 50%;

La forma en que se aplicará el margen será en orden de las manecillas del reloj, es decir, el margin-top tomará el 5%, margin-right tomará el 30%, margin-bottom tomará el 5% y margin-left tomará el 50%.

Otra forma es únicamente colocar dos valores si queremos que el top y bottom sean iguals así como el left y right: margin: 50% 50%;

Si únicamente colocams 3 valores, entonces el ultimo valor que define al left tomará el valor de right.

Otra manera es usando margin: auto. Pero se deberá especificar que lado quieres que sea auto para que no tomes top,bottom,left y right. Por ejemplo: margin-left: auto;

**Colapsado de márgenes.**

El colapsado de márgenes solo ocurre de manera vertical. Los márgenes verticales se colapsan, se juntan.

Se recomienda que para márgenes de elementos verticales siempre se utilice el margin-bottom, asi mismo, si tenemos un elemento con margin-bottom: 100px, y el siguiente elemento tiene un margin-top: 100px, no se creara un espacio de 200 px, por que no se suman, sin embargo si alguno de los valores es mayor que el otro ahí si tomara ese valor para aplicar el margin.

**Clase 5: Bordes y Sombras**

Cuando aplicamos un border a nuestra caja, el border se aplica dentro de la caja, no por fuera, y esto hace que el tamaño de nuestra caja que definimos en un principio incrementa dependiendo el tamaño de nuestro border. Sin embargo, existe un estilo que nos permite mantener el valor de tamaño definido a nuestra caja y que únicamente aumente o disminuya el tamaño del border:

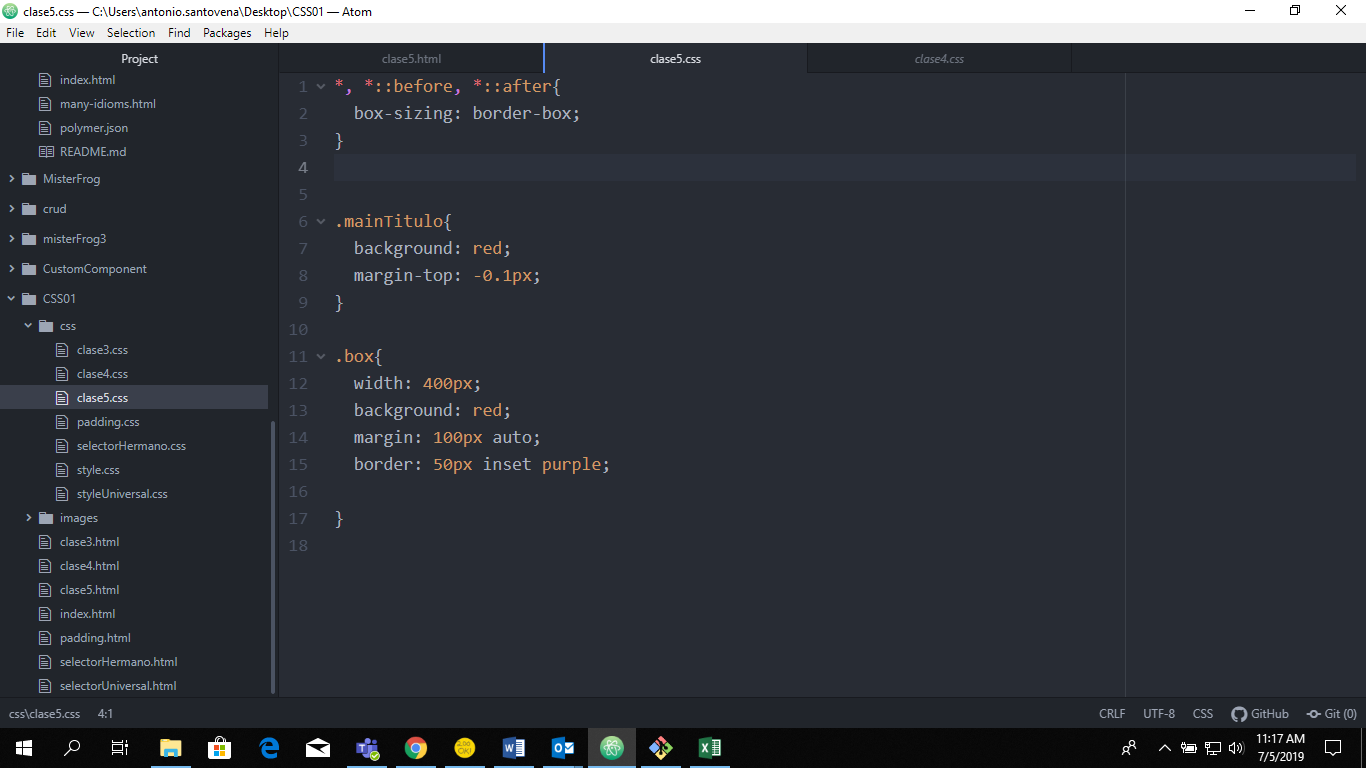
\*, \*::before, \*::after{

box-sizing: border-box;

}

Por buena práctica, se deberpa implementar el estilo con selector univeral, para que al aplicar un border no aumente el tamaño definido en nuestra caja

Ejemplo:



**Características de un border.**

Ancho, estilo y color (border: 20px solid red;)

También podemos definir nuestro border en propiedades diferentes

border-style:20px;

border-width: solid;

border-color: red;

* Podemos decir que hasta ahora hemos manejado short hands, es decir, originalmente para poder agregar el estilo se debía definir cada uno de sus lados, por ejemplo:

border-top-style: solid;  
border-right-style: solid;  
border-bottom-style: solid;  
border-left-style: solid;

* Ahora bien, podemos también hacerlo en una misma línea, por ejemplo, si deseamos que cada uno de los 4 lados de nuestra caja tenga diferente border (funciona igual que margin, dirección manecillas del reloj):

border-style: solid dotted dashed double

**short-hand:**

border-style:

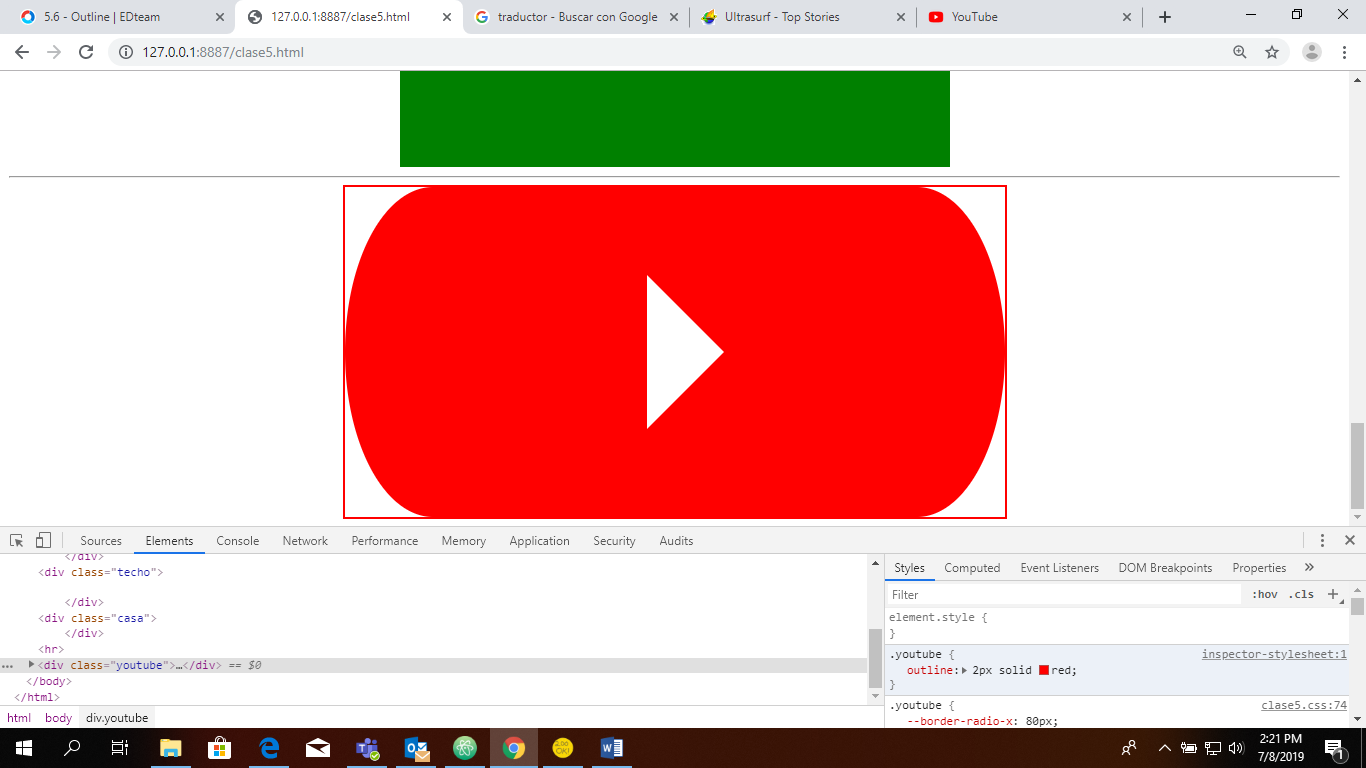
border-color:

border:width:

**border-radius:**

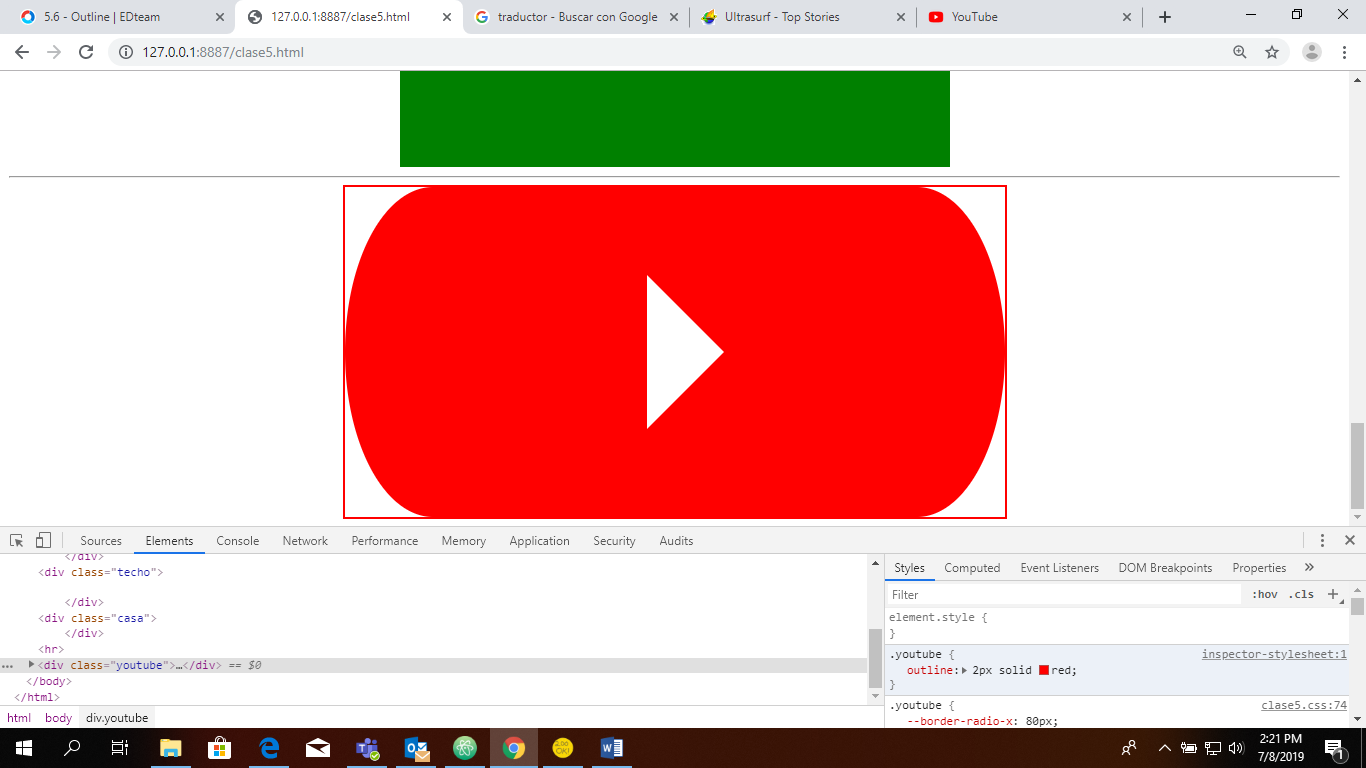
Permite redondear las esquinas de la caja, y al igual que la propiedad borer, la podemos colocar fija para todas las esquinas o una por una:





**outline:**

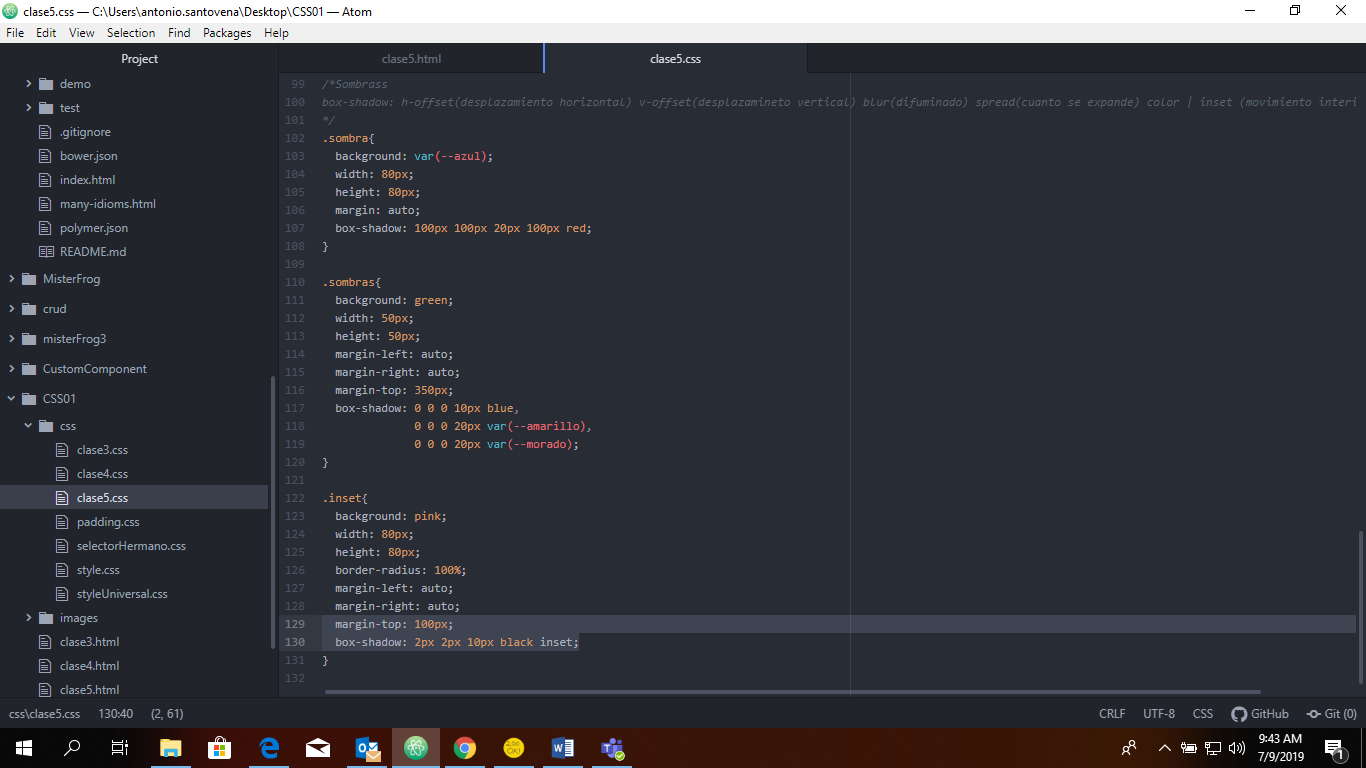
En realidad, nos permite detectar con la herramienta de inspeccionar código que elementos tienen determinada clase, por ejemplo, si queremos saber que elementos tienen la clase .play, entonces colocamos en el inspeccinado una nueva clase con el nombre de la clase a buscar y colocamos el outline: (numero)px solid red; y nos encerrara los elementos con esa clase:

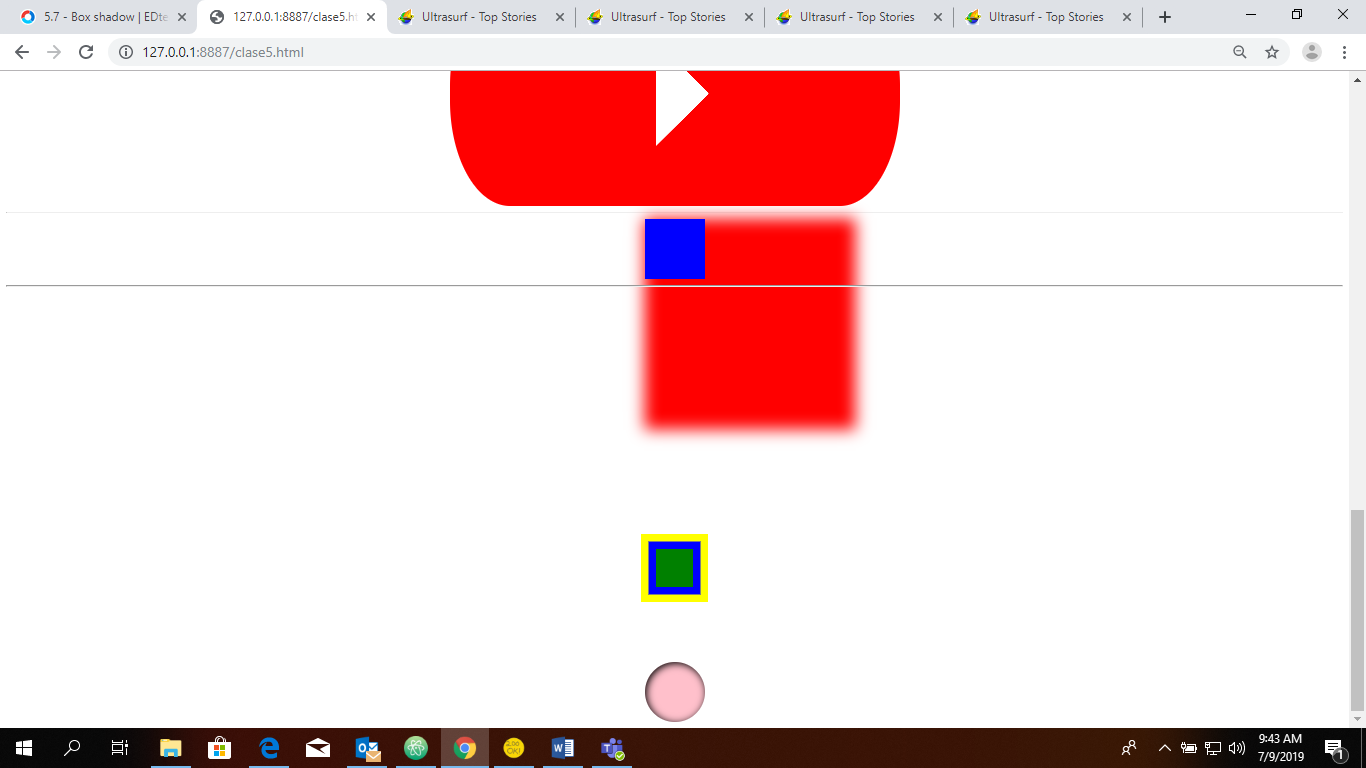


**box-shadow:**

box-shadow: h-offset(desplazamiento horizontal) v-offset(desplazamineto vertical) blur(difuminado) spread(cuanto se expande) color | inset (movimiento interior o exterior).

Con el uso de box-shadow podemos realizar bordes, cuadros sobre cuadros, etc…





[Grab your reader’s attention with a great quote from the document or use this space to emphasize a key point. To place this text box anywhere on the page, just drag it.]

**CLASE 6 Fondos**

**Shorthand:** es una propiedad que agrupa muchas propiedades, por ejemplo, para colocar un borde podemos definir en la propiedad borde tanto el tamaño de anchura, tipo de borde y color, en lihar de ir colocando cada valor en una propiedad diferente.

**background-color**: *red; //Coloca un color de fondo*

**background-image**: url(ruta\_imagen) *//Coloca una imagen de fondo. No es necesario que la ruta se encuentre entre comillas, ya que css no interpreta cadenas.*

**Background-repeat: repeat;** *//por defautl, backgraund-repeat siempre esta activado. Funciona por ejemplo, cuando colocamos una imagen pequeña, que se repita en todo el fondo.*

**Background-repeat: no-repeat;** *//elimina las repeticiones de la imagen*

**Background-repeat: repeat-(x o y)** *// repetirá la imagen horizontal o verticalmente respectivamente.*

**Backgroung-clip && backgroung-origin**

Bacgroungd-clip: (content-box, padding-box, border-box)

Te ayuda a dibujar el fondo únicamente en la zona de la caja que determines, por ejemplo, puede ser que se visualice desde el border, desde el padding o únicamente en el content

Background-orign: (content-box, padding-box, border-box)

Inicicia el fonod justo donde determines. La diferencia con background-clip es que no te ocultará nada del fondo, iniciara justo n el área detrminada, mientras que clip lo pone desde el border y solo lo muestra la parte proporcional a la zona que determines.

**Background-size**

Background-size te permite definir el tamaño de una imagen cuando la utilizamos como fondo.

**Background-size: 100%;** *//Solo tomara el 100% de width, y el height lo calcula automáticamente.*

**Background-size: 100% 100%;** tomará el 100% de ancho y alto.

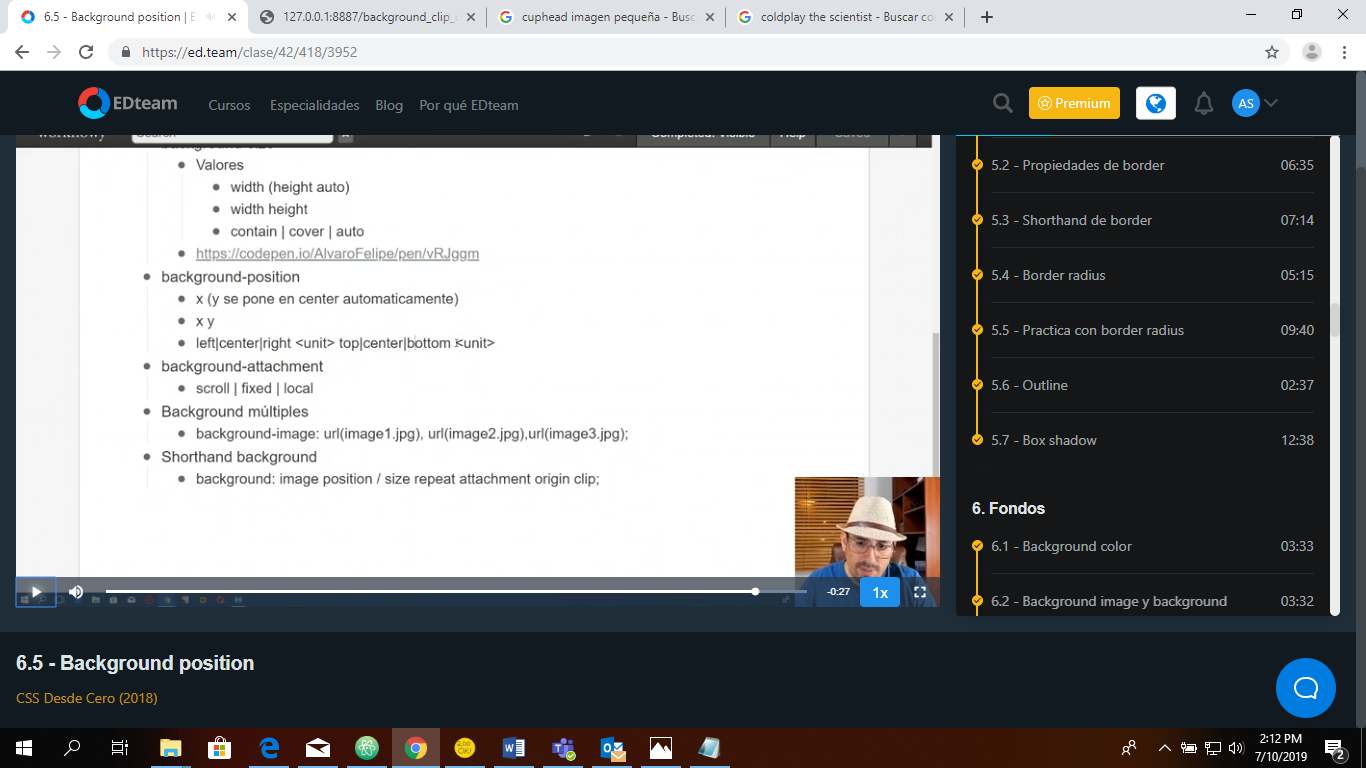
Exiten algunas palabras que nos permiten cubrir nuestra caja con la imagen dependiendo lo que desamos:

**auto:** es lo mismo que no utilizar background-size, pondrá la imagen en su tamaño real, sin importar que no se muestre toda la imagene en la caja  
**contain**: ocupa el mayor tamaño que puede ocupar

**cover**: expande la imagen para que se vea en la caja, aunque puede pixelear la imagen.

**Background-position**

Sirve para colocar nuestra imagen en una posición determinda



Podemos utilizar pixeles también:

Background-position: 20px 20px;

o

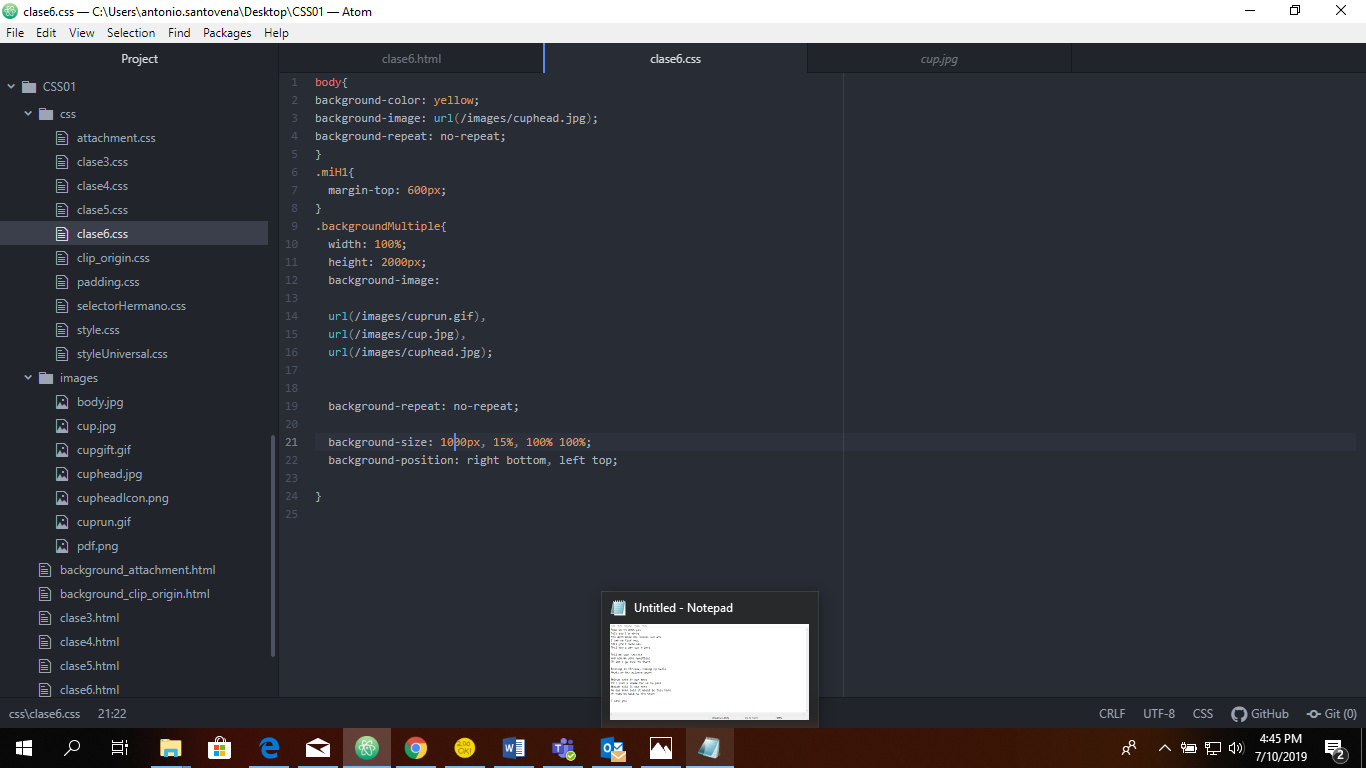
Background-position: center top;

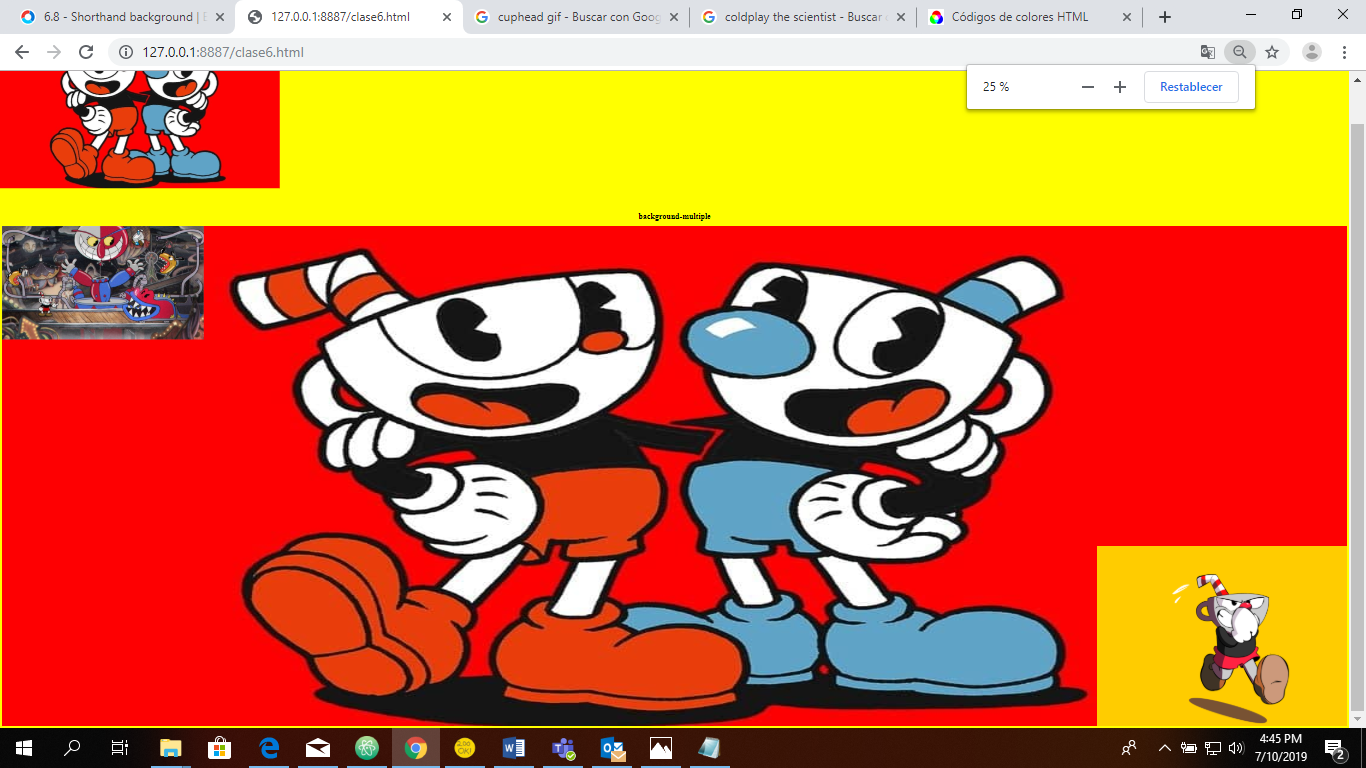
**Background-attachment**

Te permite fijar una imagen y crear un efecto que conforme te mueves en la página la imagen igual se recorre. Checar ejemplo.

**Backgroung-multiples**

Es posible ajustar varias imágenes en un solo background:





Shorthand background:

Hacer lo hecho en la imagen anterior con menos líneas:

**Clase 7 Texto**

Existen dos estilos que actualmente prácticamente ya no se utilizan.  
**direction: rtr o ltr** // Lo que realiza es alinear el texto a la izquierda o derecha. Por defatul ya se tiene en ltr, que inicia a la izquierda.

**text-ident**: 3em //Da una sangría

**text-align:**

Tiene los siguientes atributos: start(por defautl ya se tiene y varía si tulizamos direction: rtr), end, center, right, left y justify.

Existe otro estilo llamado text-align-last, que alineara únicamente la última línea de un párrafo.

**line-height:**

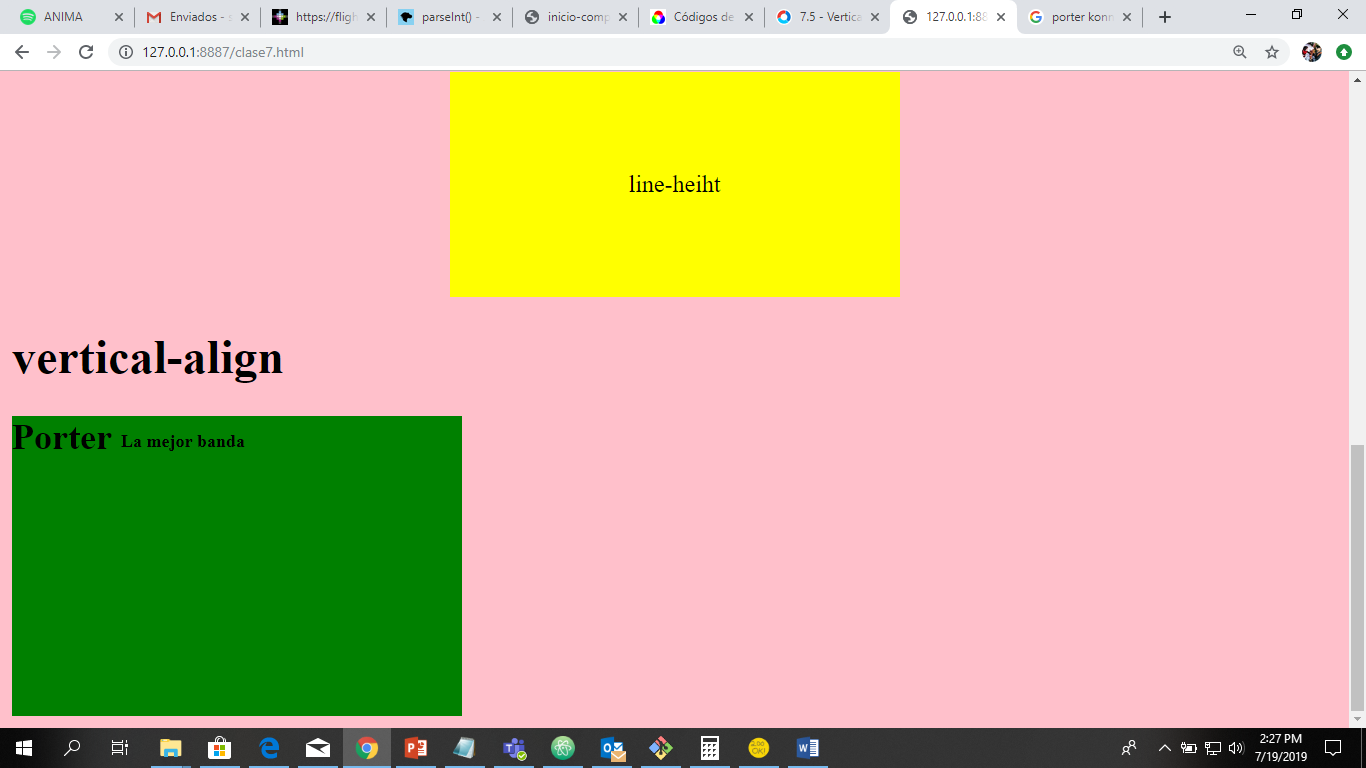
Te ayuda a la separación entre líneas: **line-height: 1.5 em;** // Se recomienda usar em, en ligar de px por que hay que ser muy exactos

**Vertical-align:**

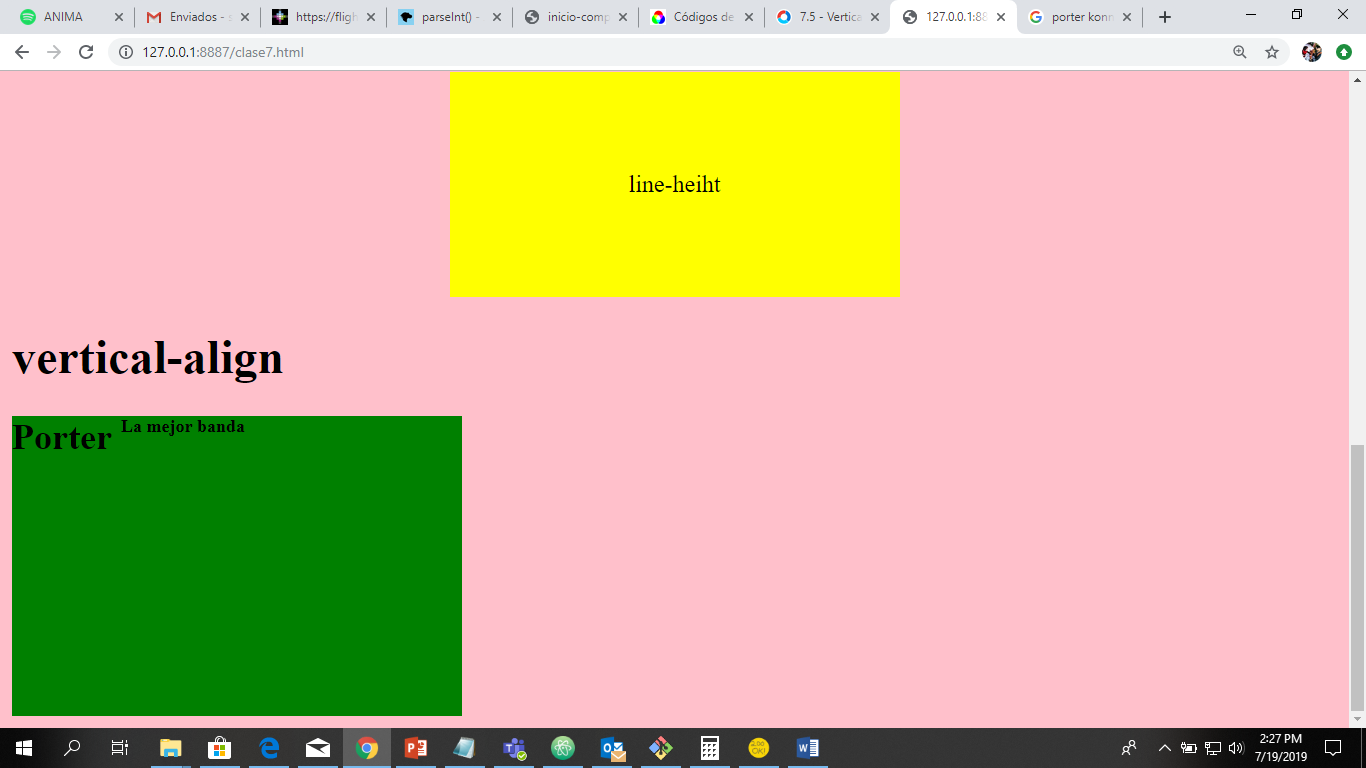
Nos ayuda a centar igualmente de forma vertical. De igual forma nos ayuda a centrar un texto respecto a su línea base.

Lás imágenes igual podemos alinearlas con este estilo para que queden en la mismas base, centro o top del texto.

Middle:



Top:



**Flujo del texto**

Letter-spacing: .3em //Separación de letras en una palabra

Word-spacing: //separación de palabras

**text-decoration**

**Dibuja líneas sobre el texto. Sirve por ejemplo al utilizar enlaces.**

Es un shorthand, de las propiedades tex-decoration-line, text-decoration-color y text-decoration-style.

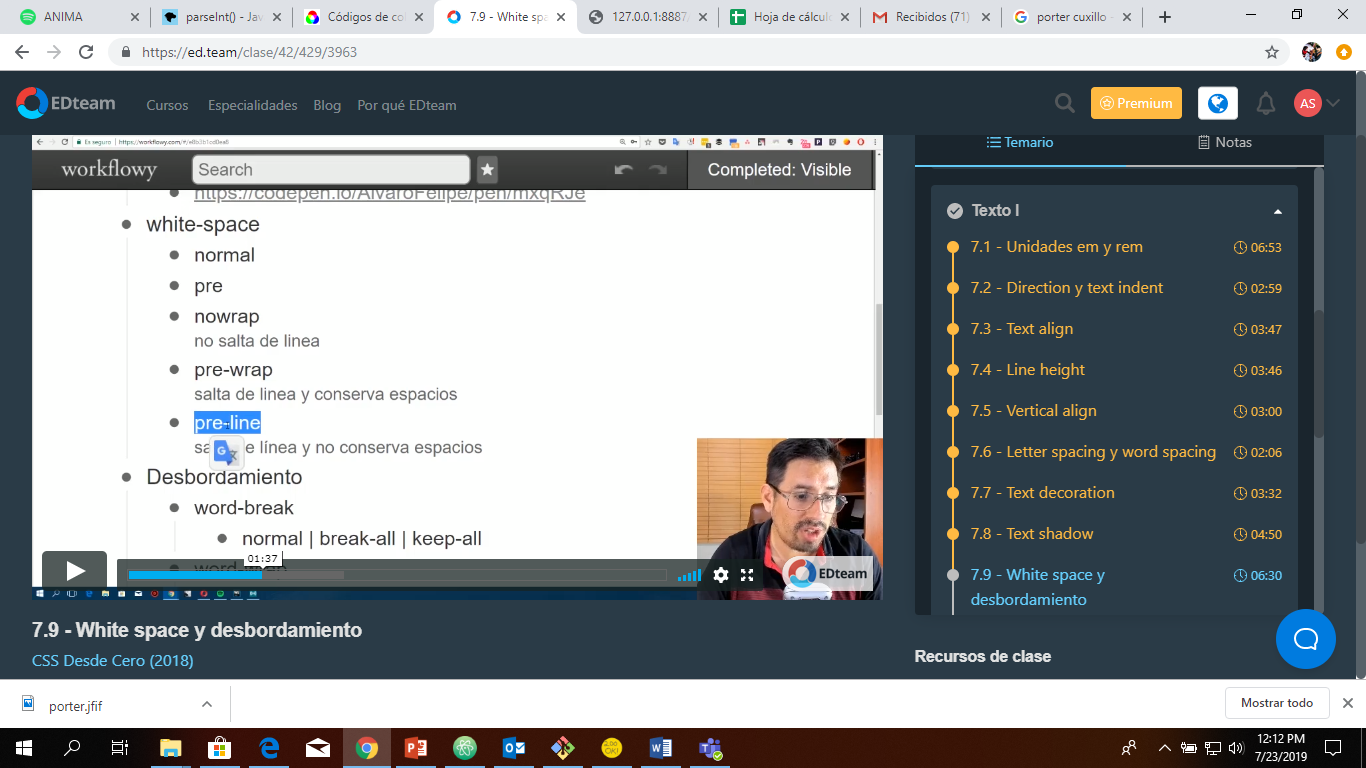
**Text-shadow**

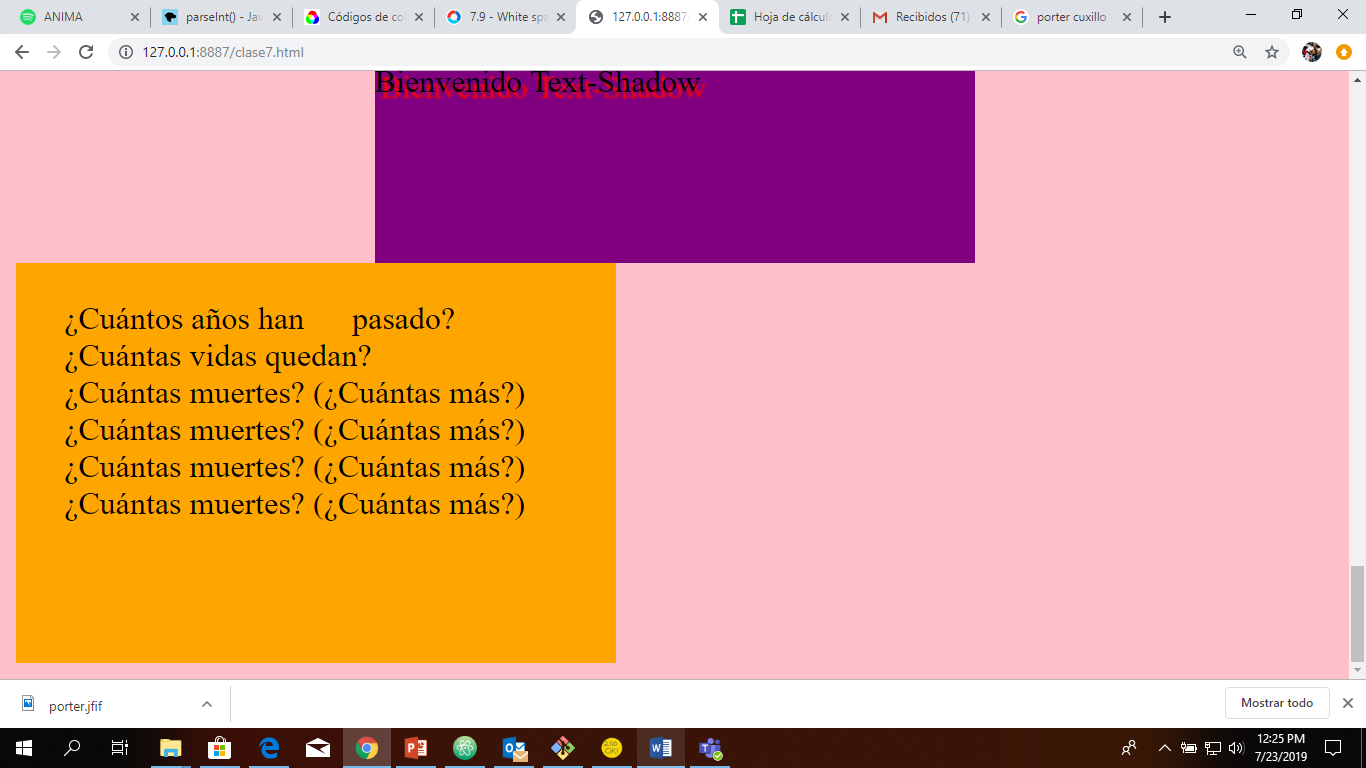
Similar a box-shadow. Da sombras a los textos:

text-shadow: 2px 2px 4px red;

**White-space**

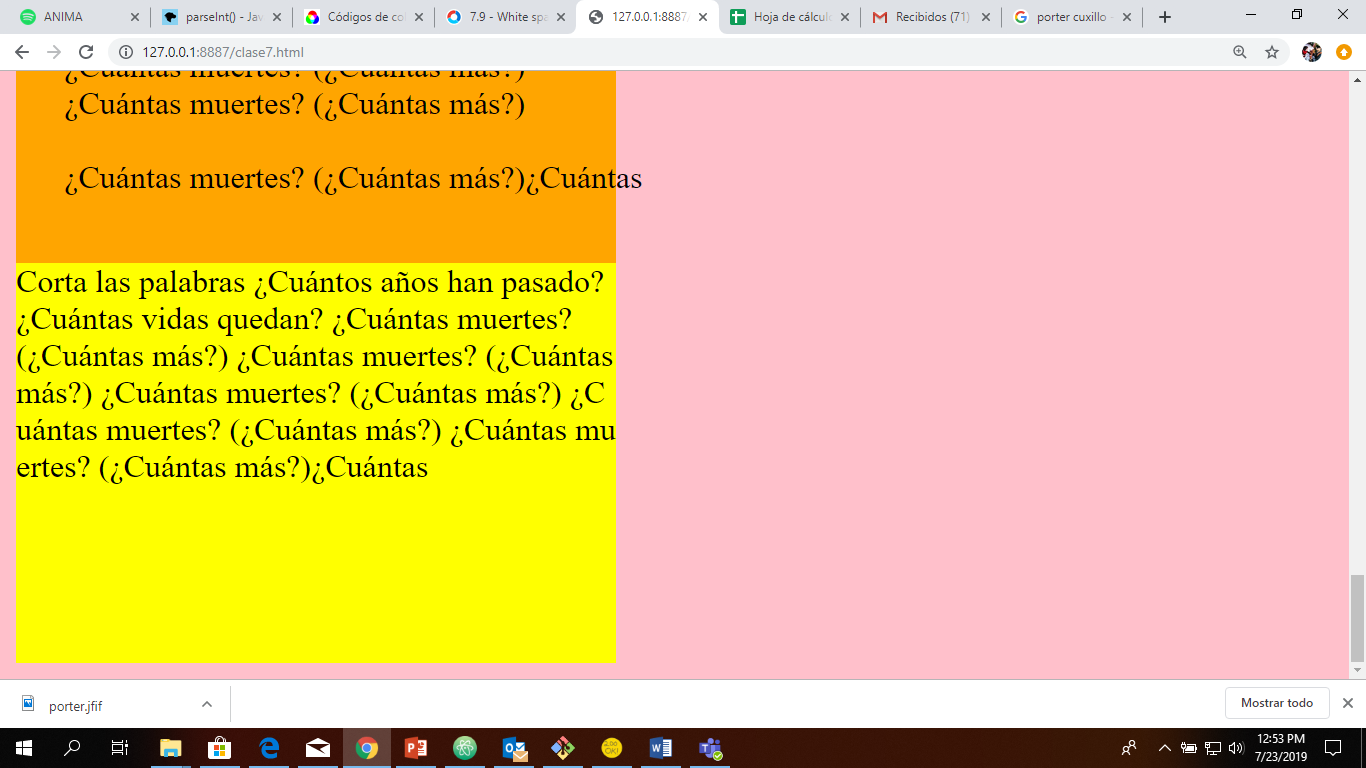
White-space: pre //El texto escrito se verá igual que en el código, es decir, cada que saltemos una linera también lo respetará en el front. Respeta los espacios.





**Word-break**

**Rompe la palabra para que no se salga del ancho de nuestras cajas y coloca la continuación de la palabra en la siguiente línea:**

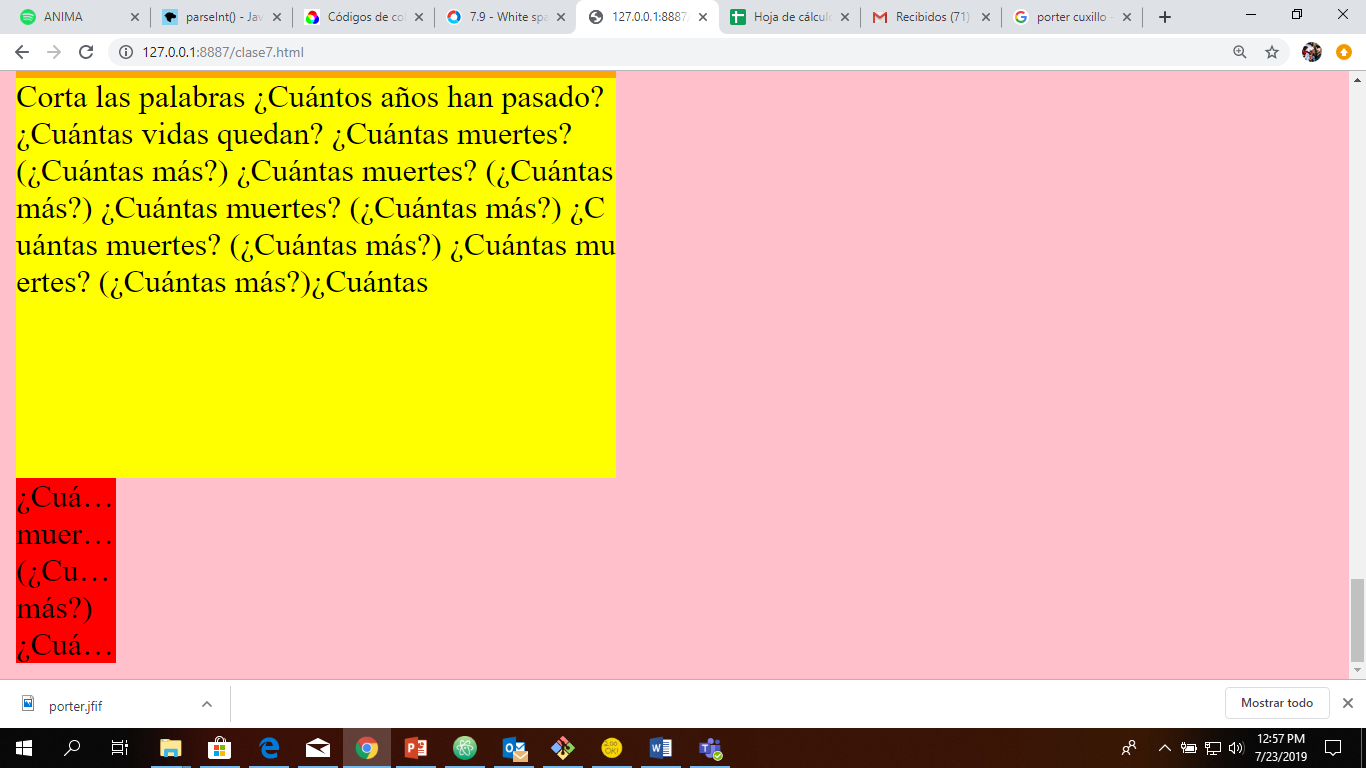


**Text-overflow**

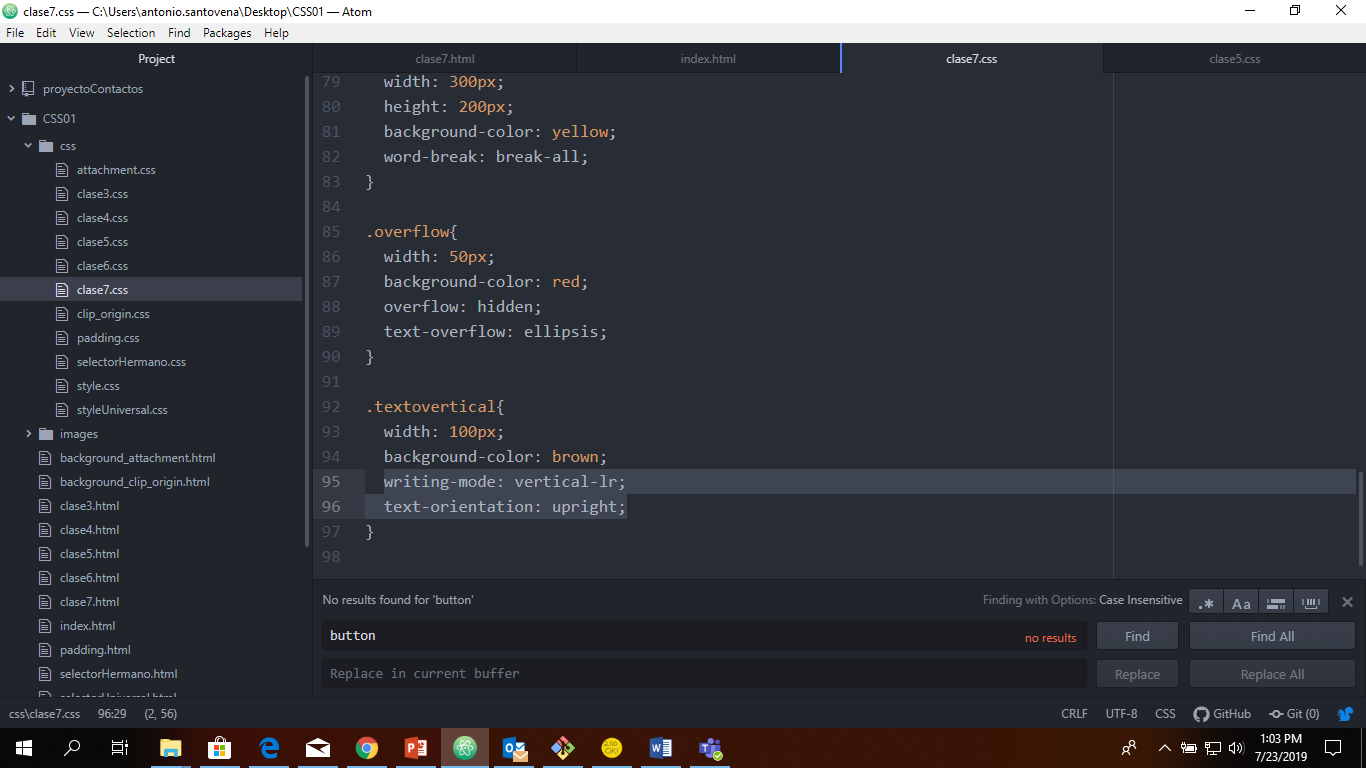
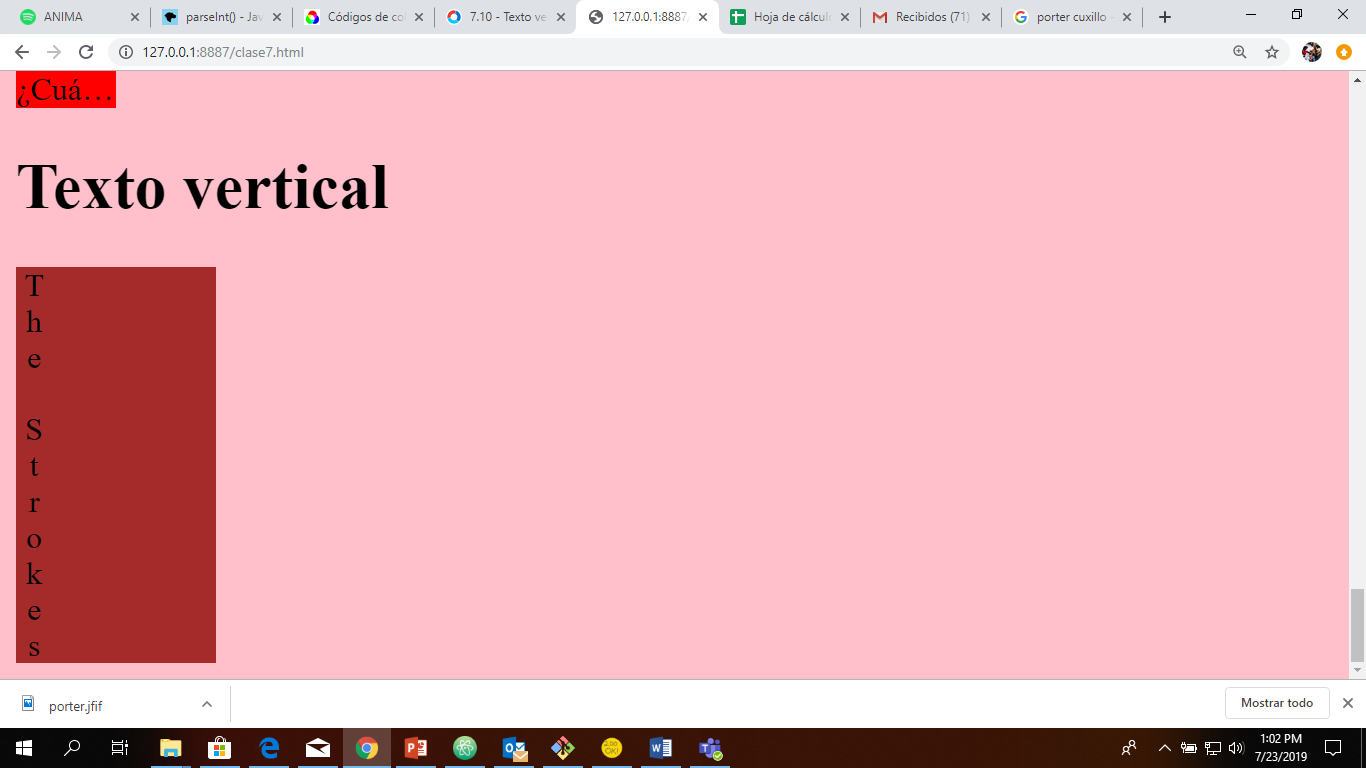
Evita que el texto salga de su casa:

Necesitamos aplicar un overflow:hidden //todo lo que se salga de la caja lo esconde

Text-overflow: ellipsis; // Le pone tres puntitos al texto que ya no se vea en la caja



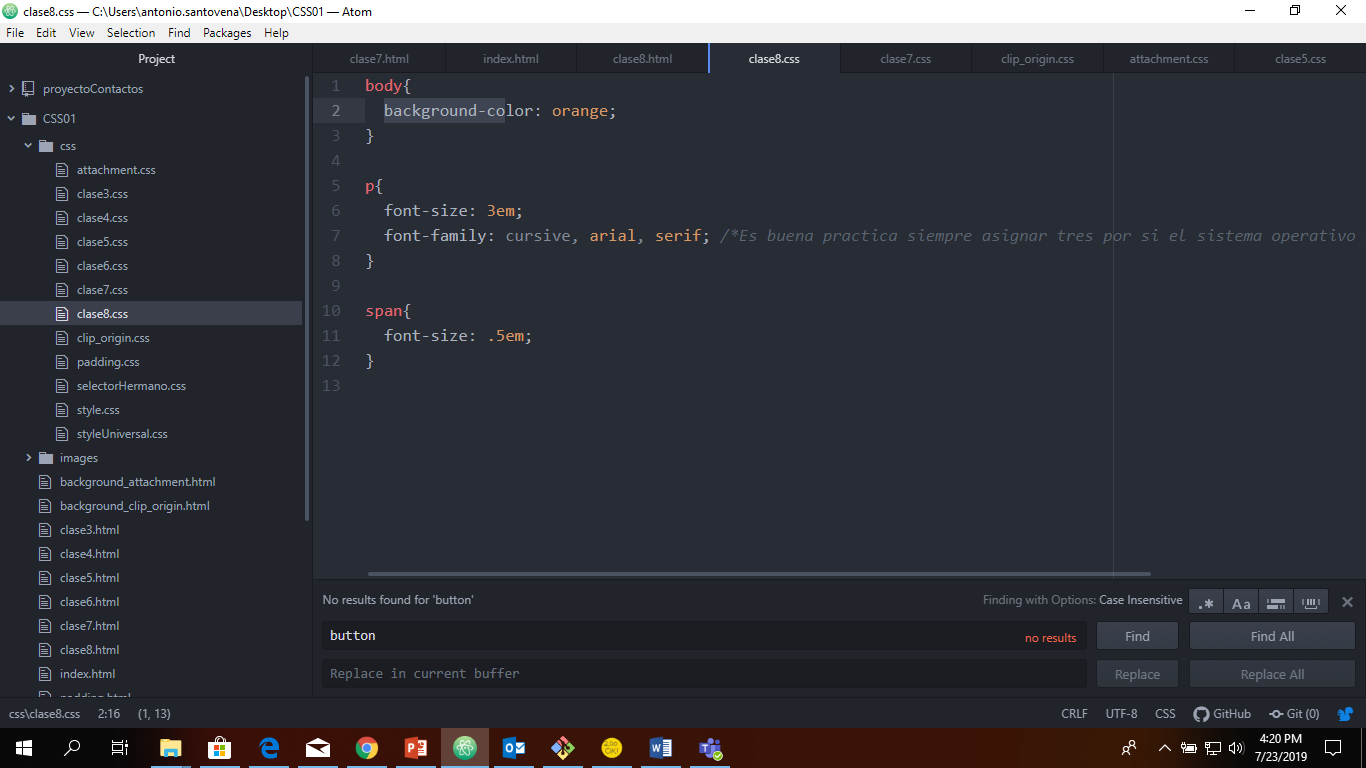
**Texto vertical**



**Text-transform**

Transforma el texto a mayúsculas, minúsculas, capitalize.

**Clase 8 Texto II**



**Clase 9 Pseudoclases**

Que son?

Son estilos especiales que van a depender del contexto en donde se encuentren los elementos del html, también en base a las respuestas de alguna acción dl usuario, por ejemplo cuando pase el mouses de un lugar a otro, clics etc.

**Pseufoelementos**: En base a si es la primer letra del texto, características, etc..

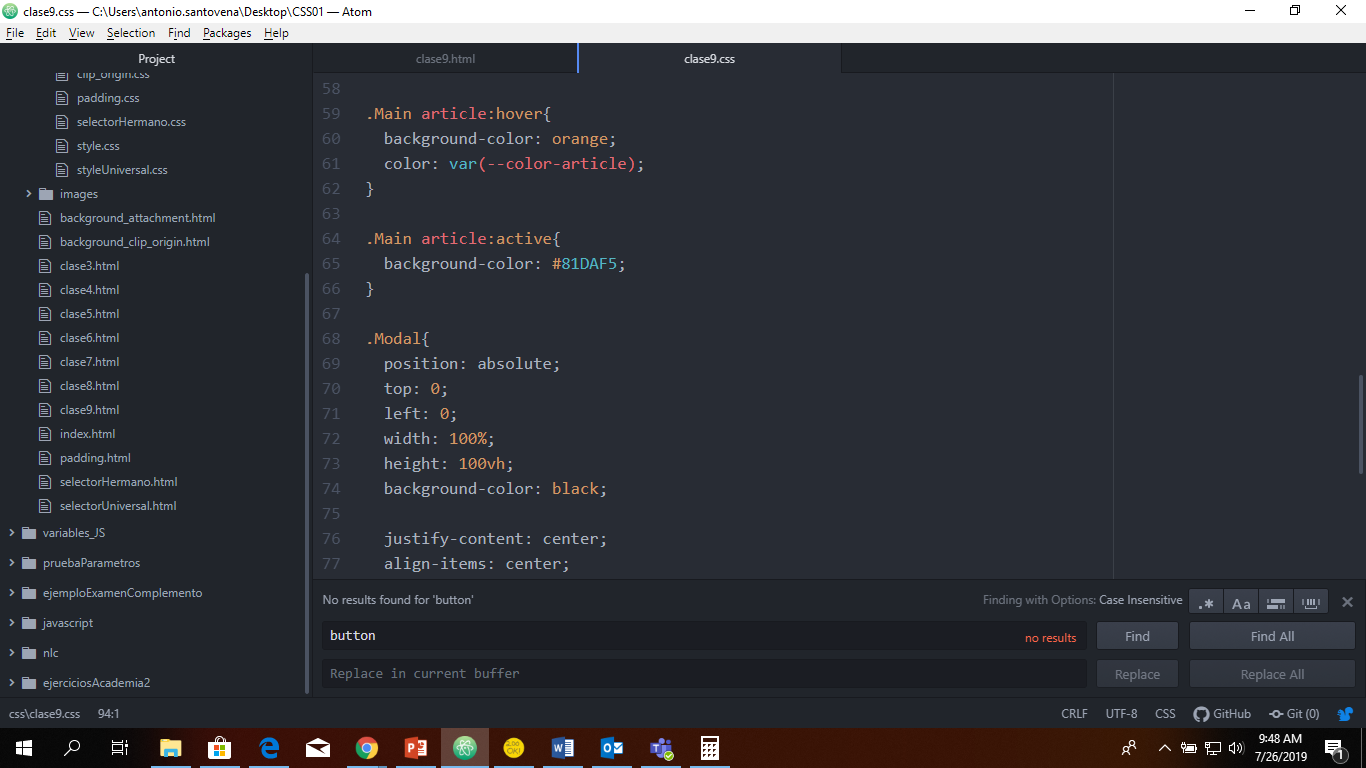
**::hover**

**Se ejecuta cuando el usuario coloca el puntero sobre el elemento definido.**



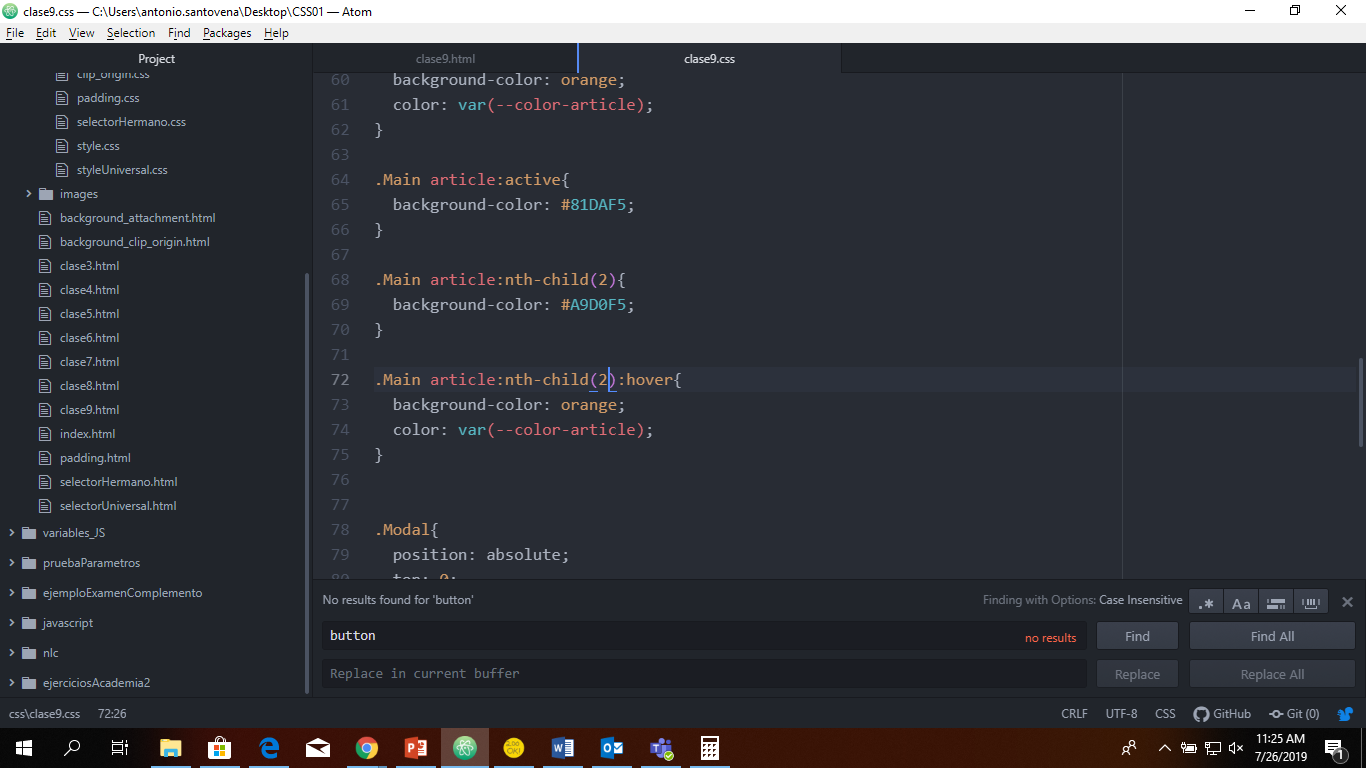
**:active**

**Se ejecuta cuando el usuario hace clic sobre un elemeto.**



**nth-child:**

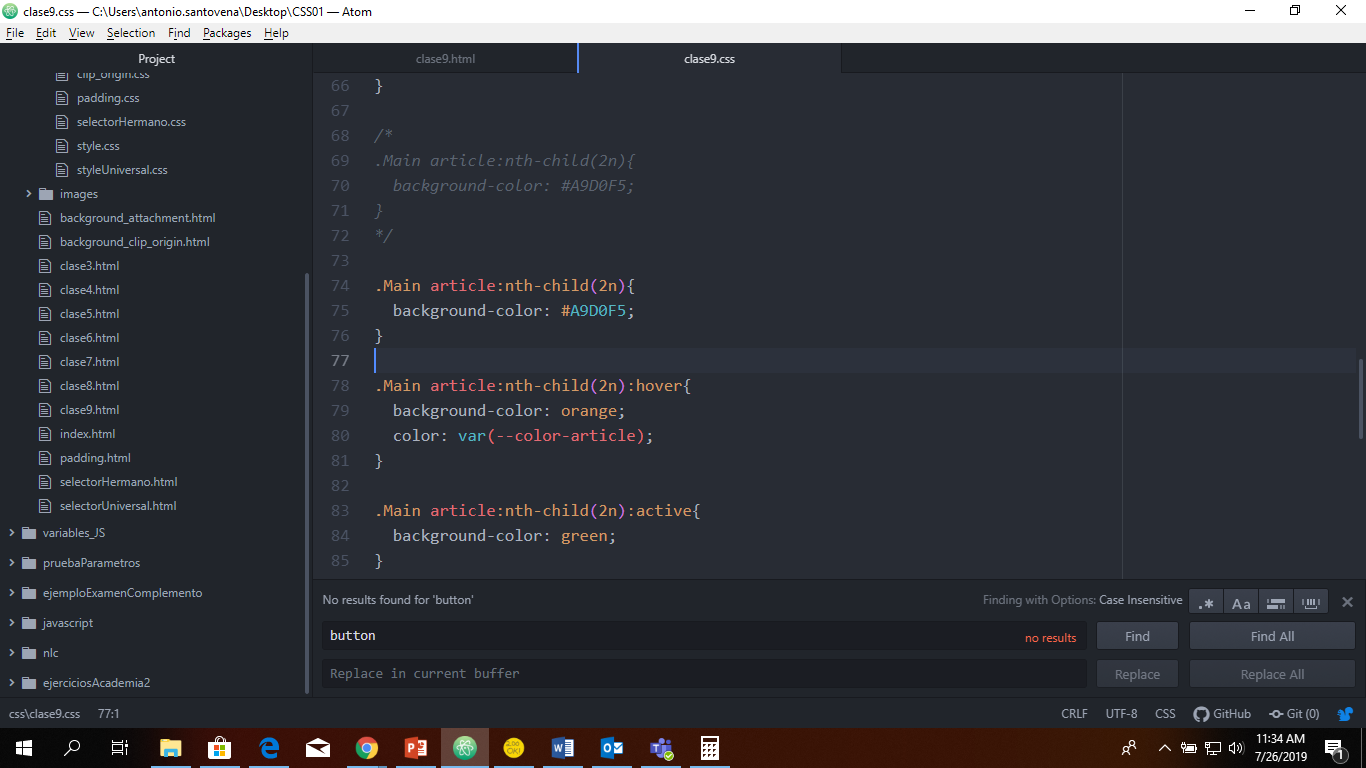
**Define la posición del elemento.   
Puede ser la posición que deseamos.**



Estamos indicando que en el article 2, hijo de Main, aplique un background color.  
  
Al hacer esto, puede que pierde el estilo que se tenía con la psudoclas ehover, por lo que se deberá definir de nuevo enseguida, aplicando nth-child y hover al elemento deseado.

Permite también manejar expresiones. Ejemplo:

Si colocamos la expresión (3n), por ejemplo; “.Main article:nth-child(3n){..}” lo que hará es que solo aplicara los estilos definidos cada 3 elementos. Por ejemplo si tenemos 6 elementos article, solo se lo aplicará al 3ro y 6to.

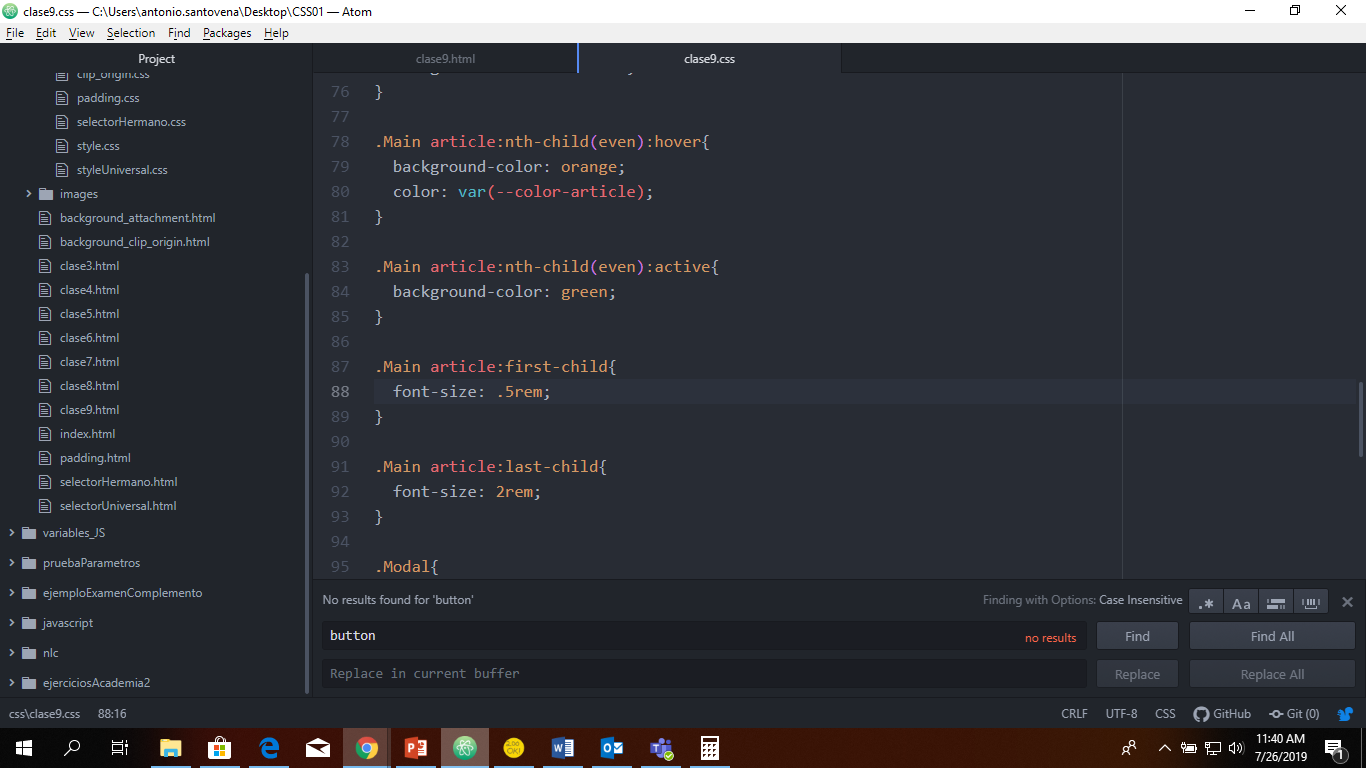


Utilizando odd (impar), even (par):

Nth-child(odd){….}

Nth-child(even){….}

**First-child y last-child,** afectarán únicamente al primer y último elemento respectivamente:



**Nth-of-type**

**Es muy parecido a nth-child, sin embargo este solo busca los elemento del mismo tipo, mientras que child busca la posición sin importar el tipo. Ejemplo:**

**Con button:nth-of-type(2) //Solo buscara los divs en la posición 2**

**<ul>**

**<li><div><button><button></li>**

**<li><button><button><div></li>**

**<li><div><div><button></li>**

**</ul>**

**Mientras que child hubiera pintado el Segundo div de el trecer li ya que solo busca la posición dos sin importar el tipo de elemento**

**<ul>**

**<li><div><button><button></li>**

**<li><button><button><div></li>**

**<li><div><div><button></li>**

**</ul>**

**Guía u ejemplos de child y type:** [https://css-tricks.com](https://css-tricks.com/)

**Not**

**Permite hacer una excepción sobre un elemento que pertenece a un padre pero que tal vez no quieres que se vea afectato. Por ejmplo, si tienes varios input y entre ellos hay un type=”submit”, y no quieres que ese se vea afectado lo utilizas:**

**.Main input:not([type=”submit”]){**

**Backgrougn-color: red;**

**}**

**Empty**

Nos puede servir cuando no queremos que se apliqué un estilo a los elementos vacíos. Por ejemplo, si tenemos una clase que determina que los <p> tengan un Font-size de 3rem, y colocamos un <p> en el html nos dará un espacio aunque no diga nada, utilizando el empty podemos quitarle ese estilo:

.Main p:emty{

Font-size: 0;

}

**Clase 10 PseudoElementos**

Son pseudoelementos que no se encuentran en el DOM, son como los hijos de los elementos.

Tenemos dos parejas de psudoelementos:

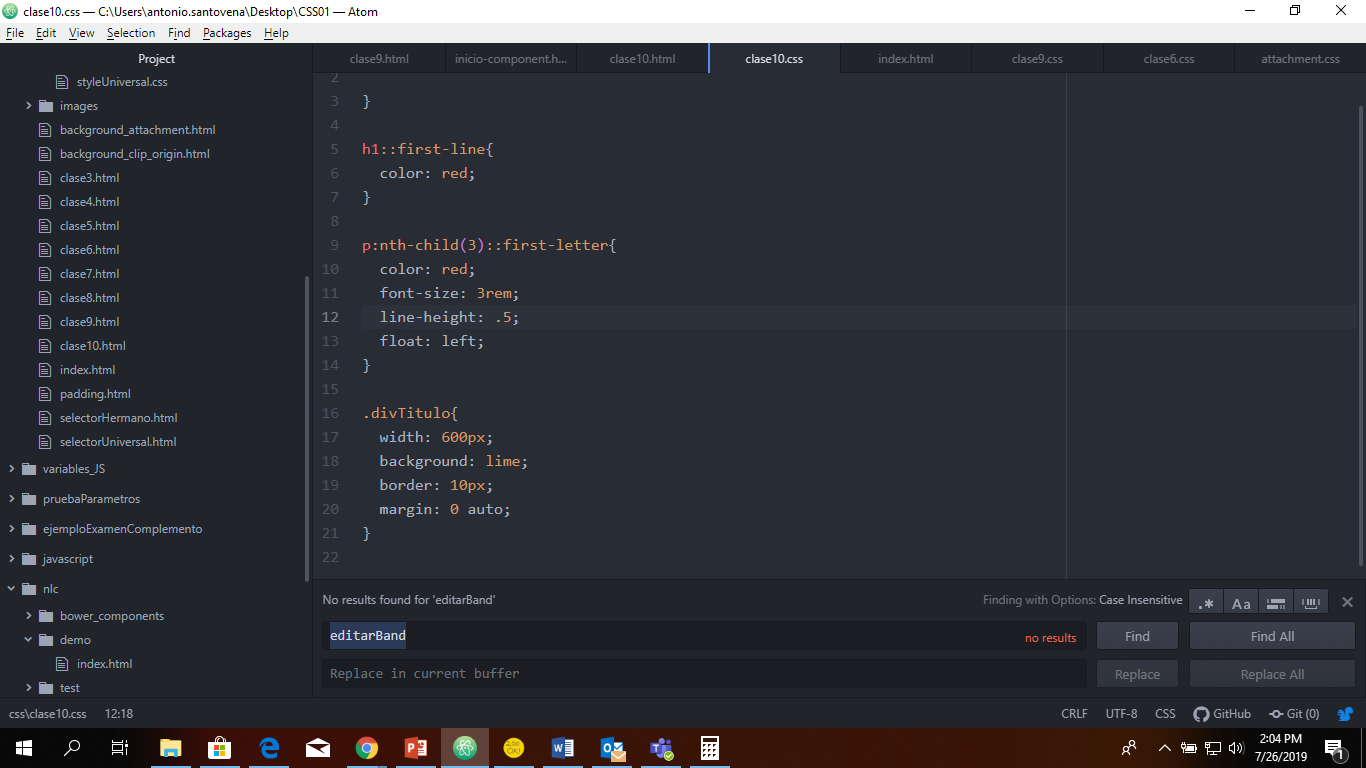
Un pseudoelemento se declaran con 4 puntos “::”

::first-line | ::first-letter

\*Solo funcionan para elementos de bloque

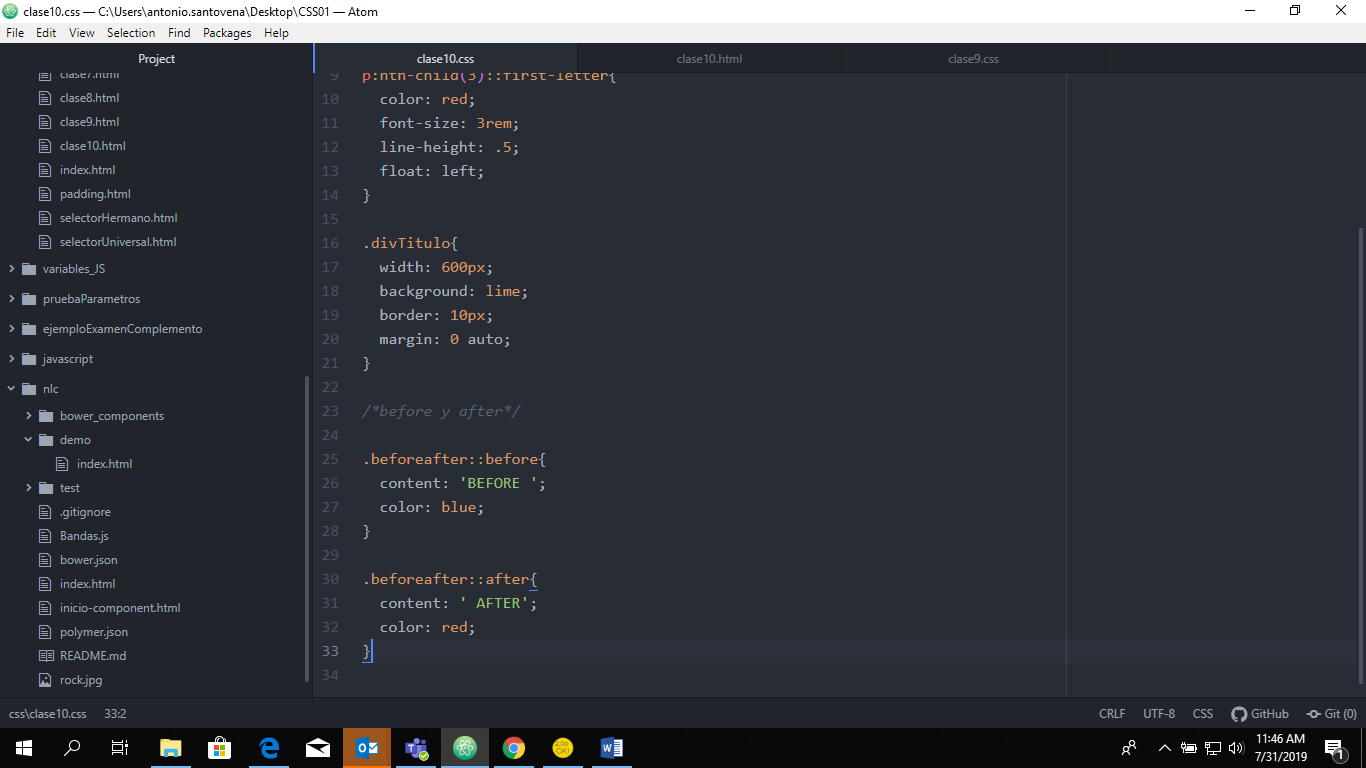
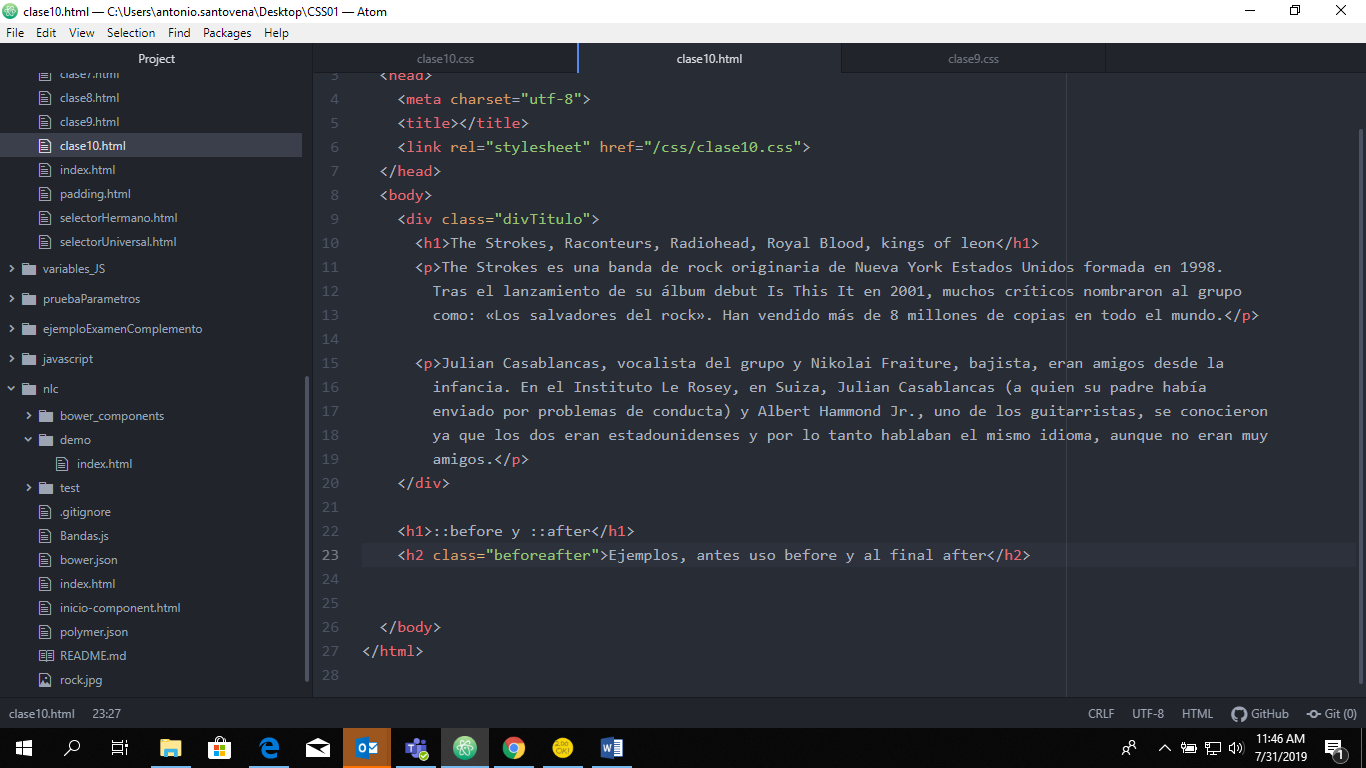
\*Selecciona la primera línea o letra

Ejemplos:



**::before | ::after**

Los pseudoelementos before y after generalmente tienen una propiedad llamada content, en la cual podemos agregar texto en forma de cadena que se concatenará a algún texto si lo deseamos:

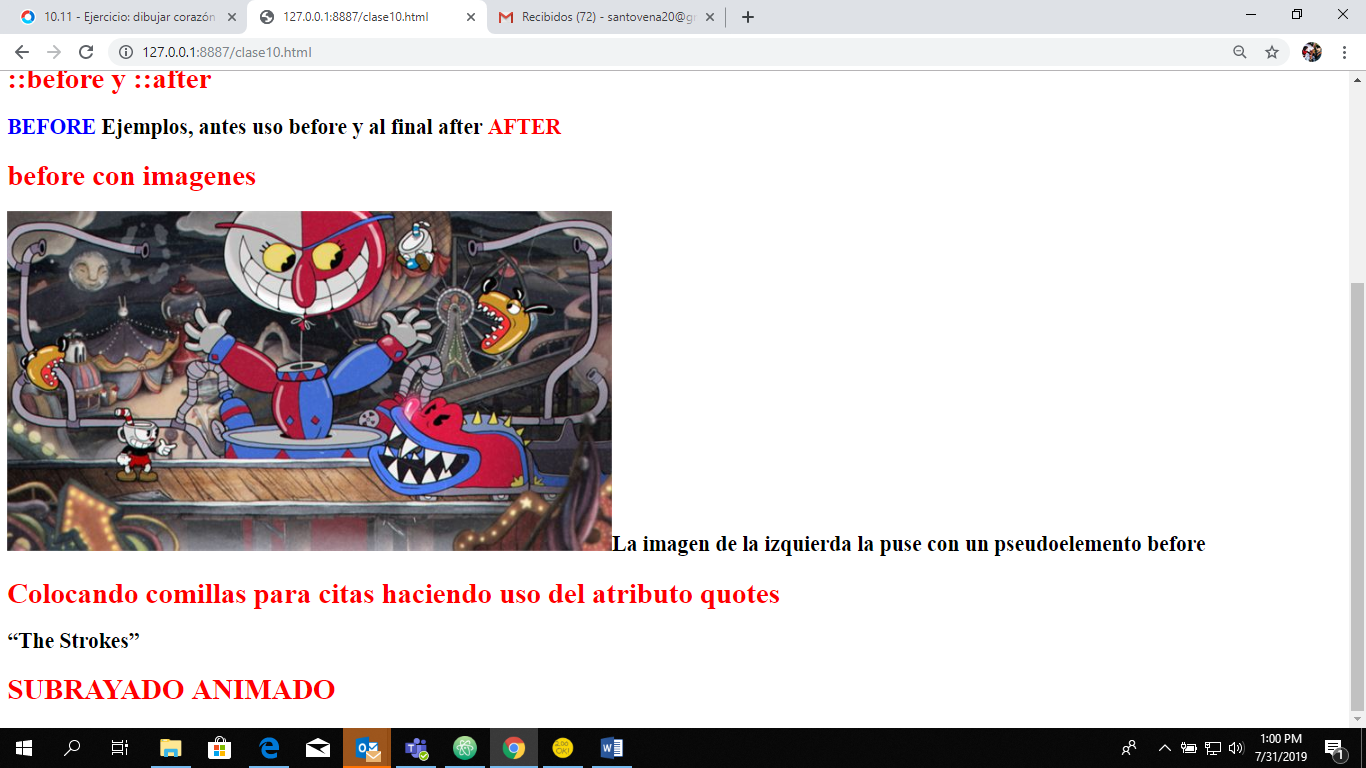
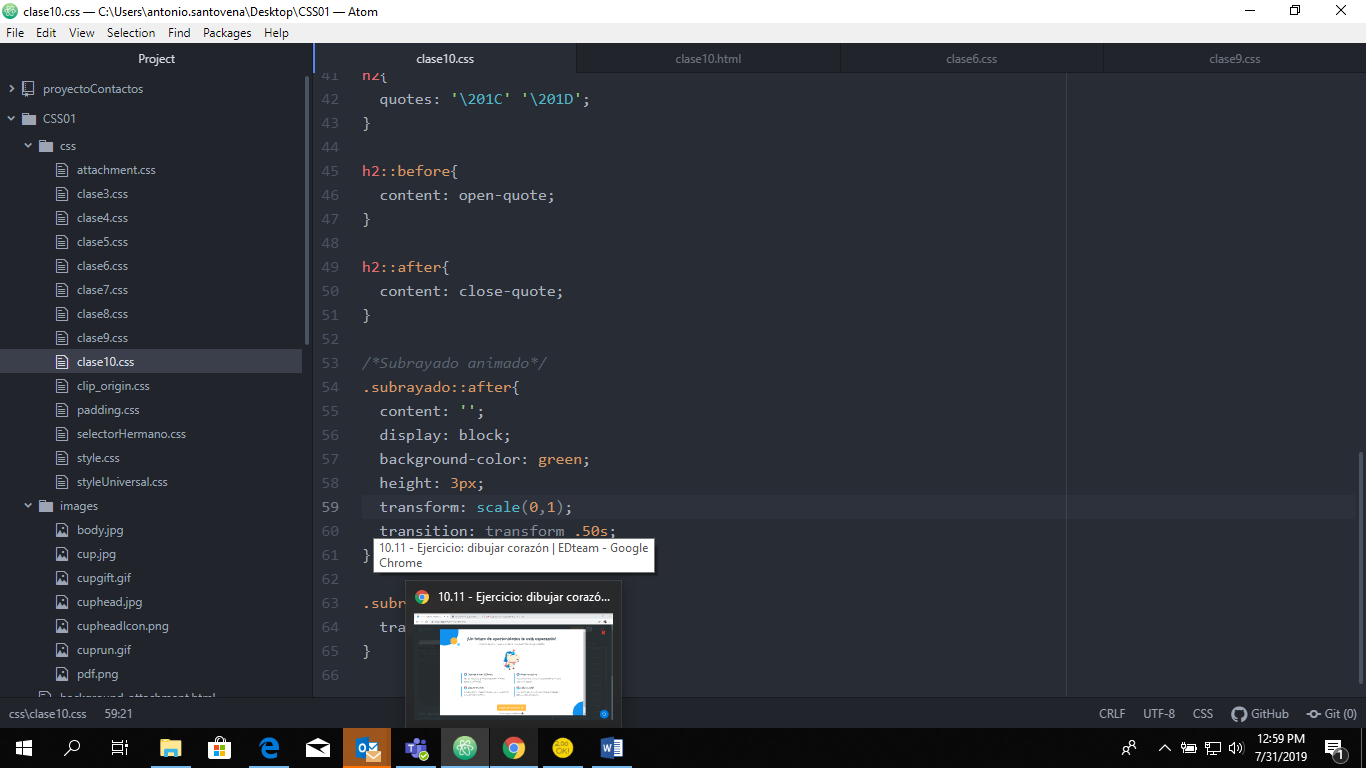




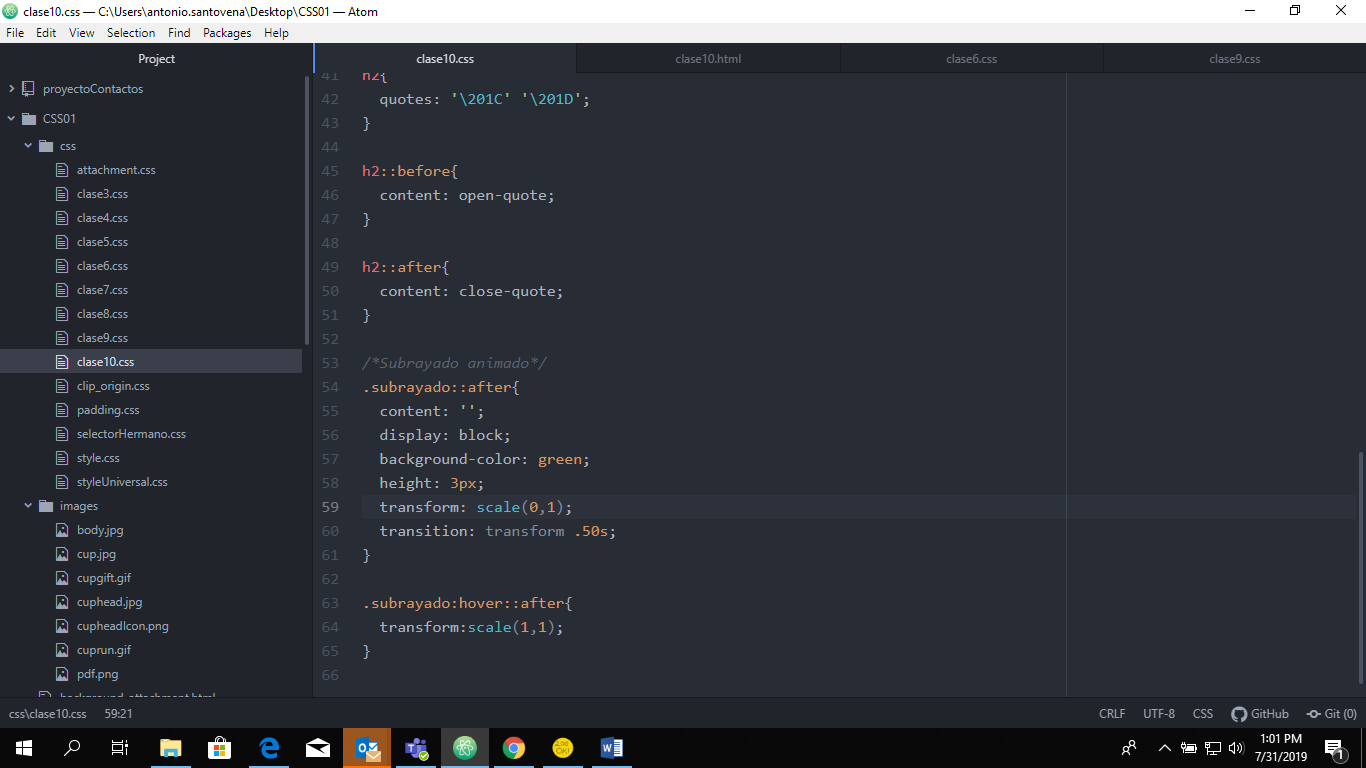
Los pseudolomentos por naturaleza son elementos inline, por lo que si queremos manipularlos deberemos convertirlos en block (display: block;).

También puedes imprimir imágenes en lugar de texto, en la propiedad content, colocamos la url(imagen);

**Colocar comillas:**



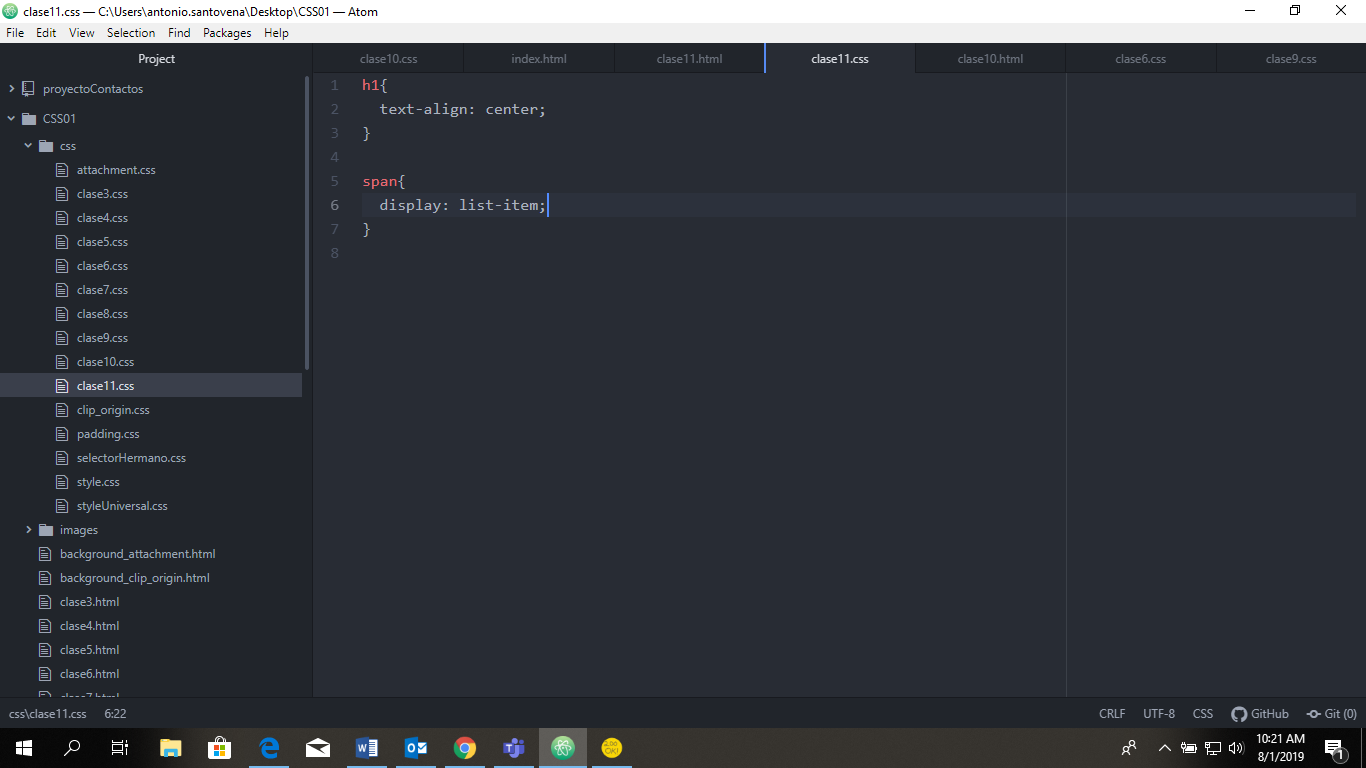
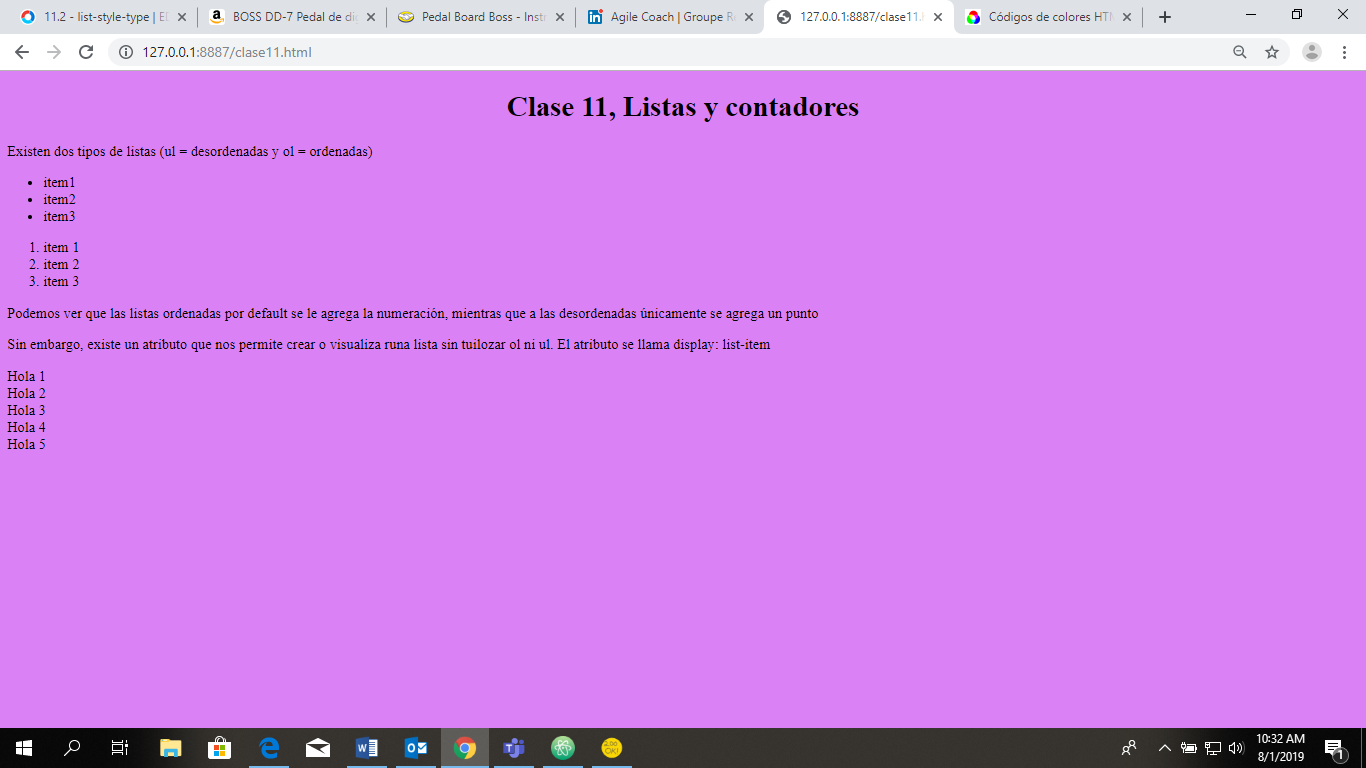
**Subrayado animado:**



**Clase 11: Listas y Contadores**

Existen dos tipos de listas <ul> (desordenadas y con viñeta) y <ol> (ordenadas y enumeradas)

Existe un estilo que crea un aspecto de lista. Por ejemplo, si tenemos varios elementos span, podemos ordenarlos de la siguiente manera:

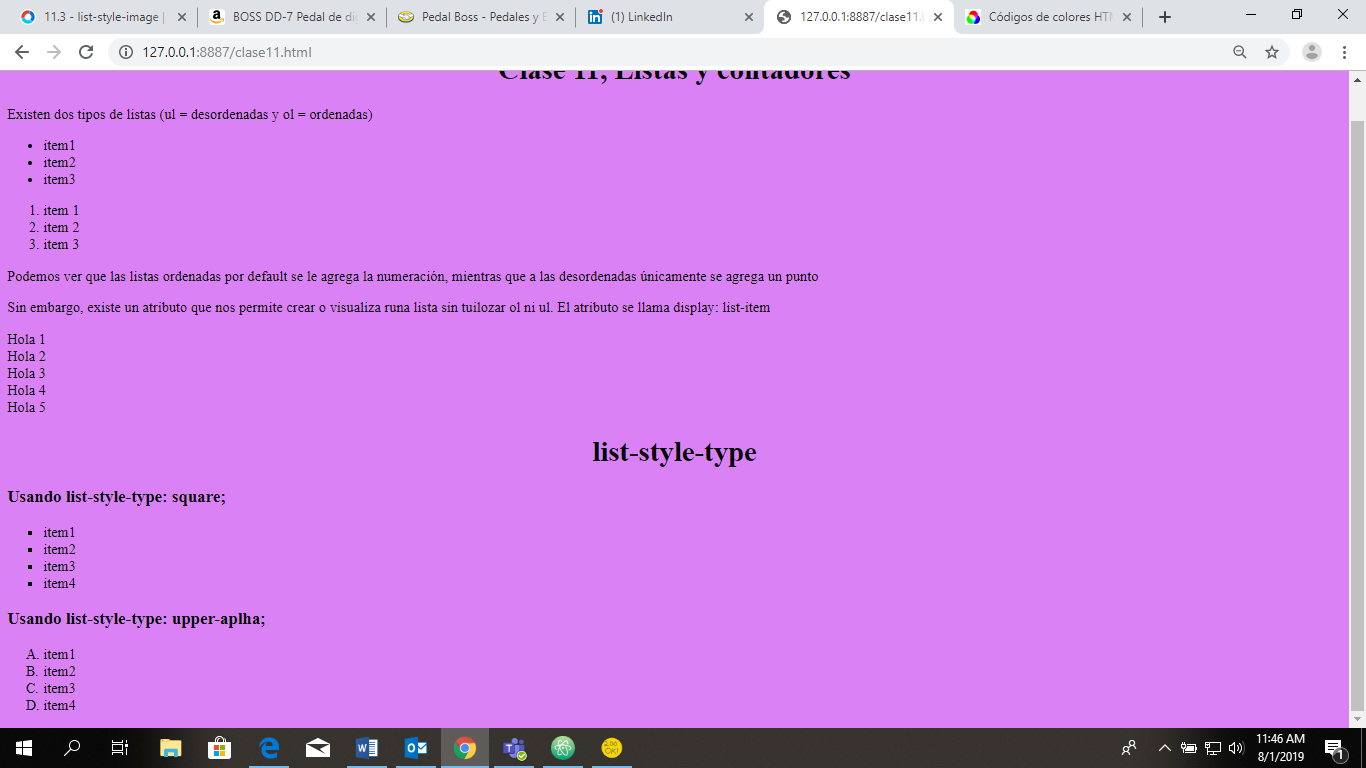


Lista desordenada <ul>

Lista ordenada <ol>

Span aplicando display: list-item;

List-style-type: disc/circle/square/decimal/decimal-leading-zero/lower-alpha/upper-aplha;



**List-style-image**

No es muy usado por que no se puede dar un tamaño a la imagen que seleccionamos. Depende del tamaño de la imagen que empleamos.

Lo más recomendable en caso de querer emplear imágenes es hacerlo por medio de un pseudoelemento before, agregando la imagen y ajustándola a lo que necesitamos.

list-style-image: url();

**LIST-STYLE-POSITION**

Cuando utilizamos cualquier tipo de lista, notamos que la viñeta o número en su caso no forma parte del li. Se encuentra dentro de UL o OL pero no en el li. List-style-position nos permite incluirlo dentro del li.

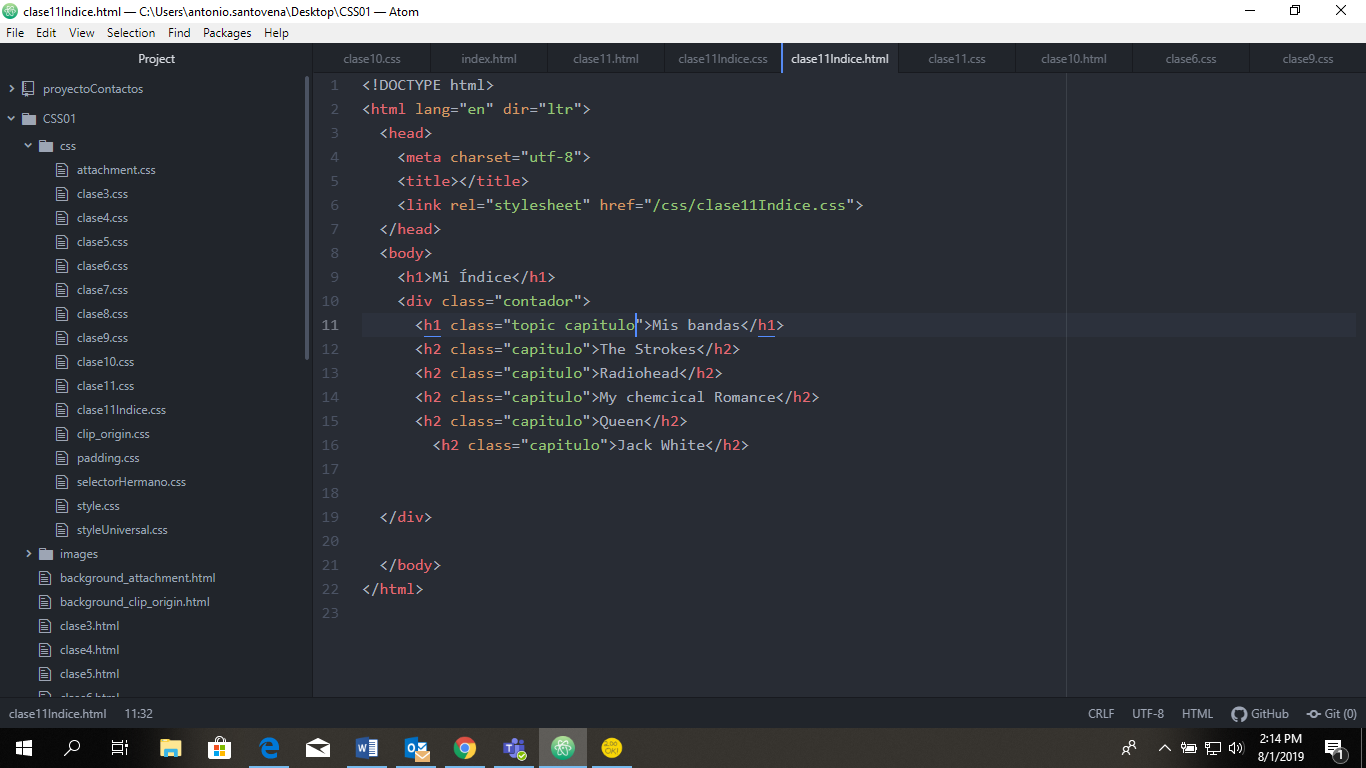
SI queremos separar el li de nuestro borde podemos usar un padding

List-style-position: inside;

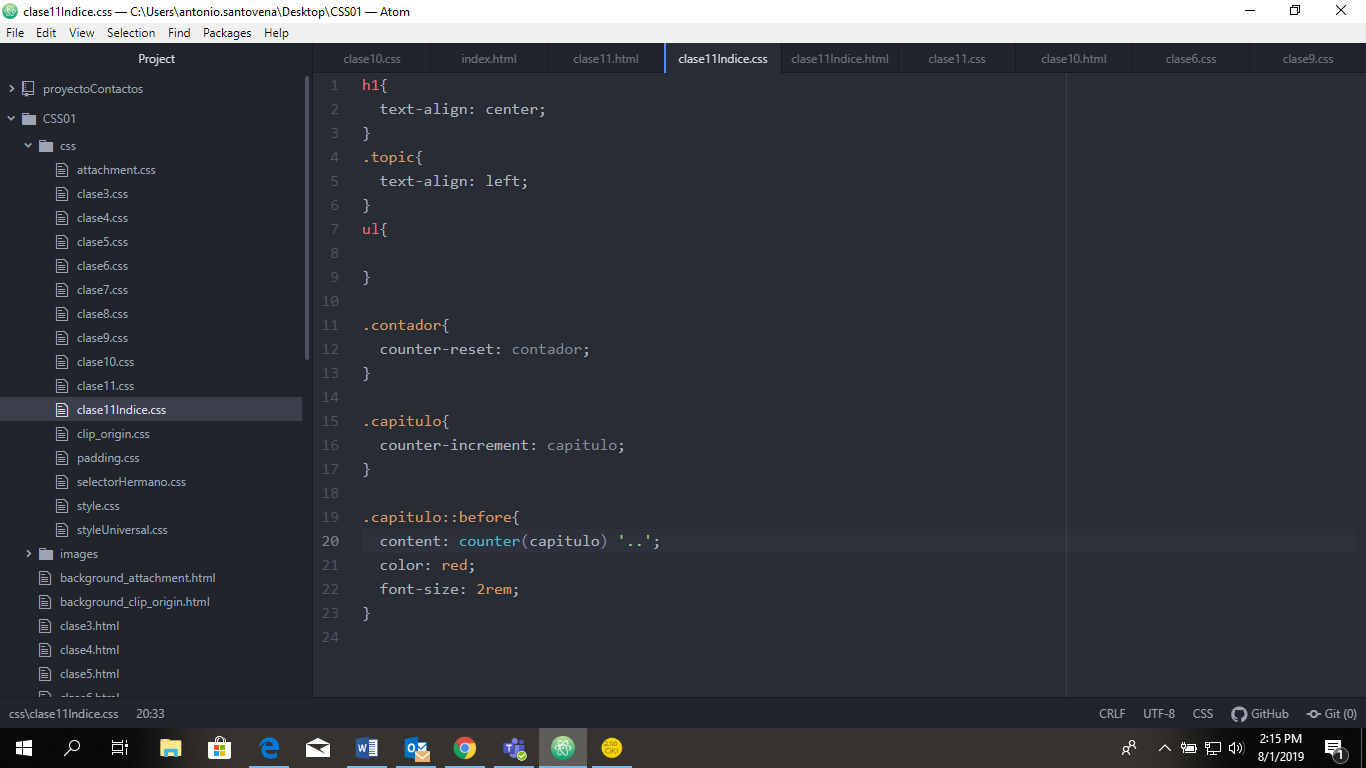
**Contador**

Nos permitere realizar índices personalizados, empreando otros elementos que nos listas, como por ejemplo elementos h1 o h2. Para ellos se debe implementar de la siguiente manera:

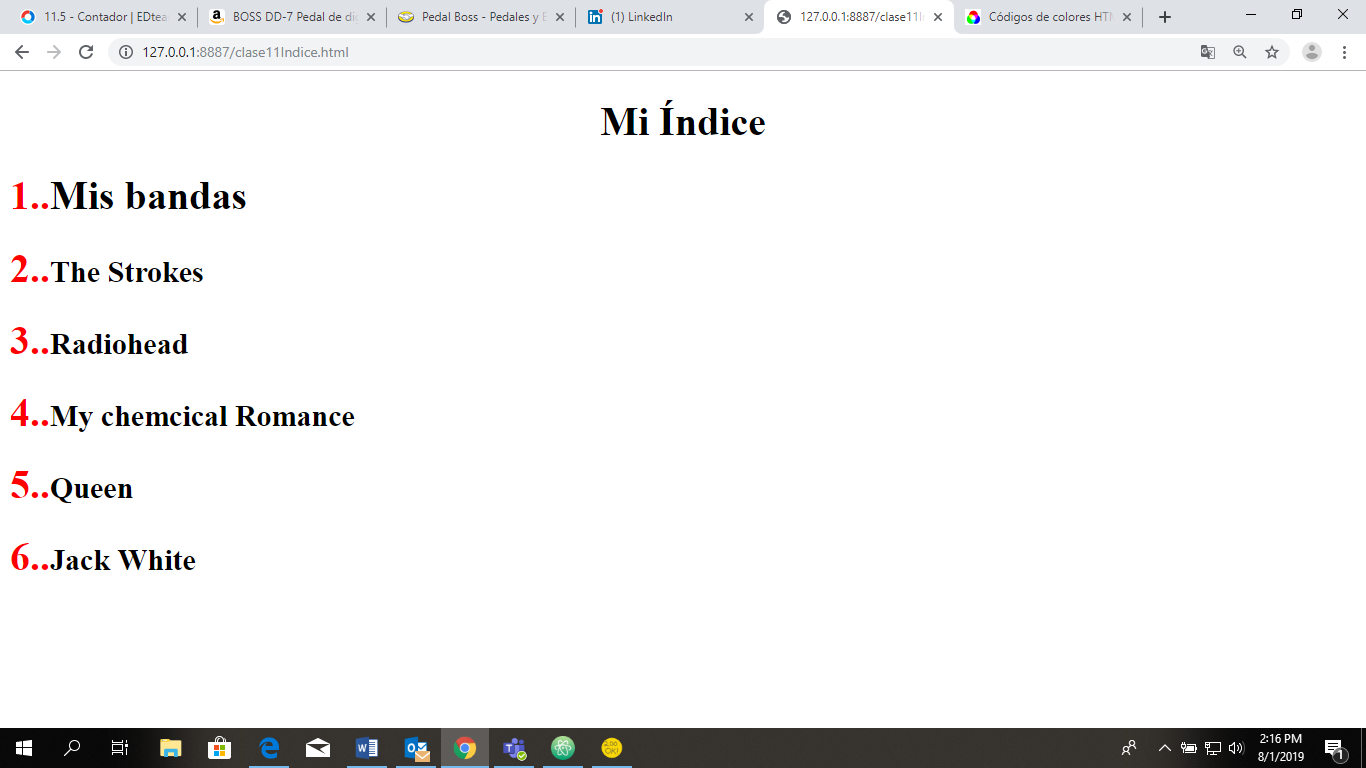
Es importante determinar el alcance de nuestro índice, por ello en este ejemplo lo encarramos en un div con la clase contador. Despues cada capitulo de nuestro índice tiene la clase capitulo.



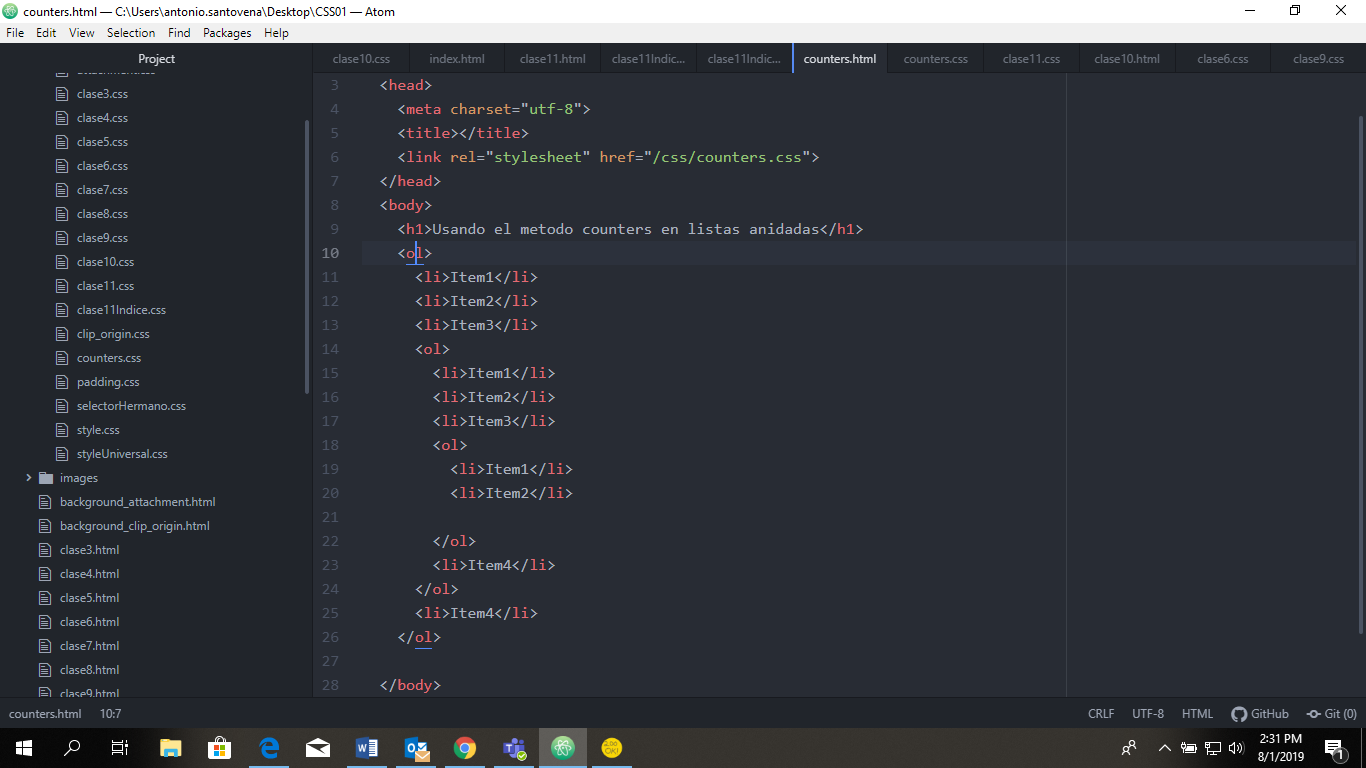
**En el archivo css creamos lo siguiente:**

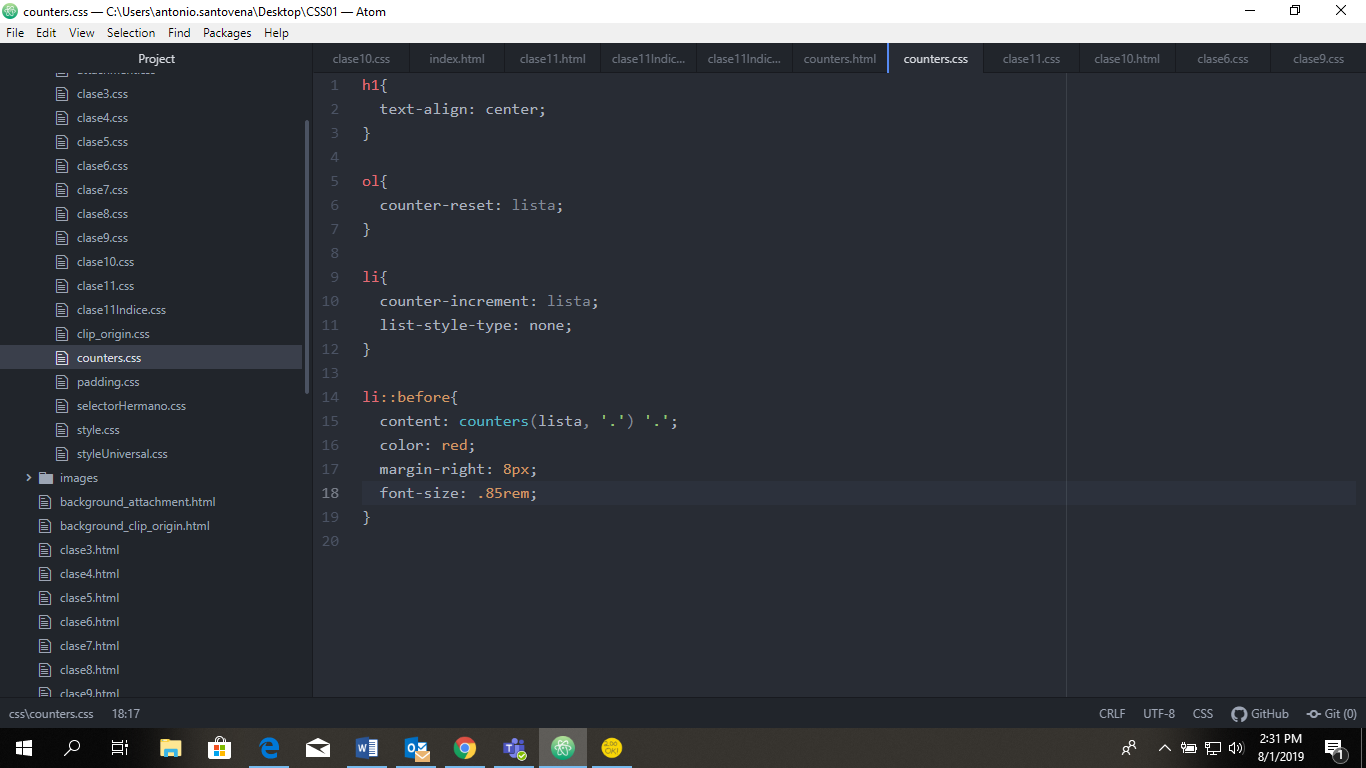


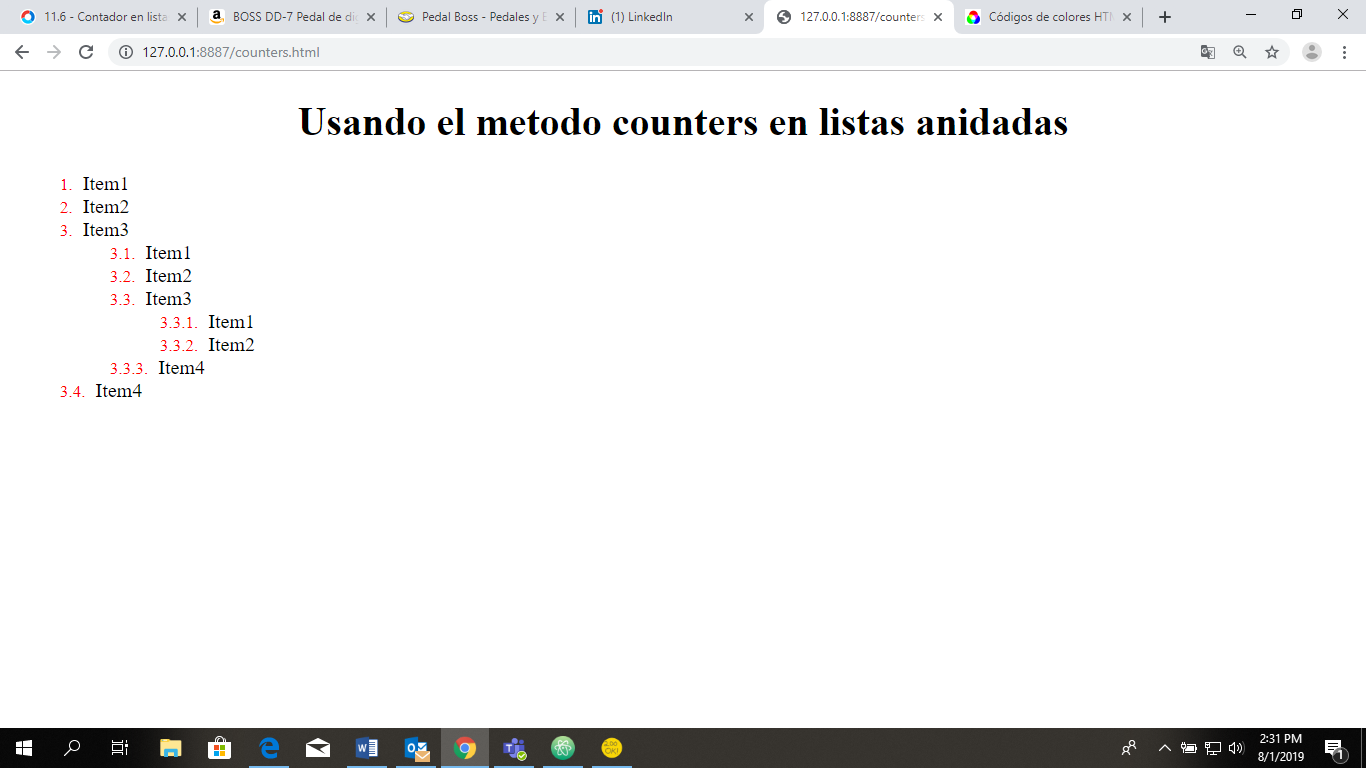
Como resultado obtendremos lo siguiente.



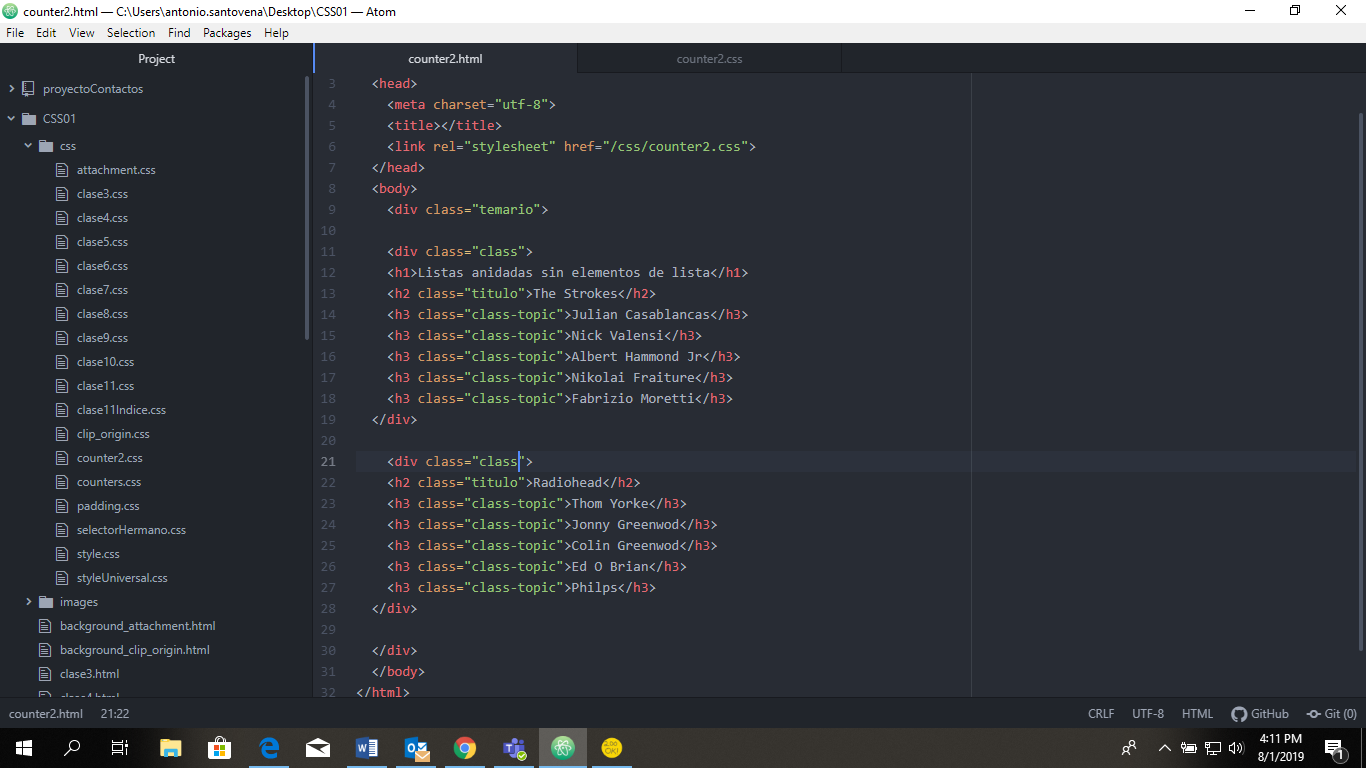
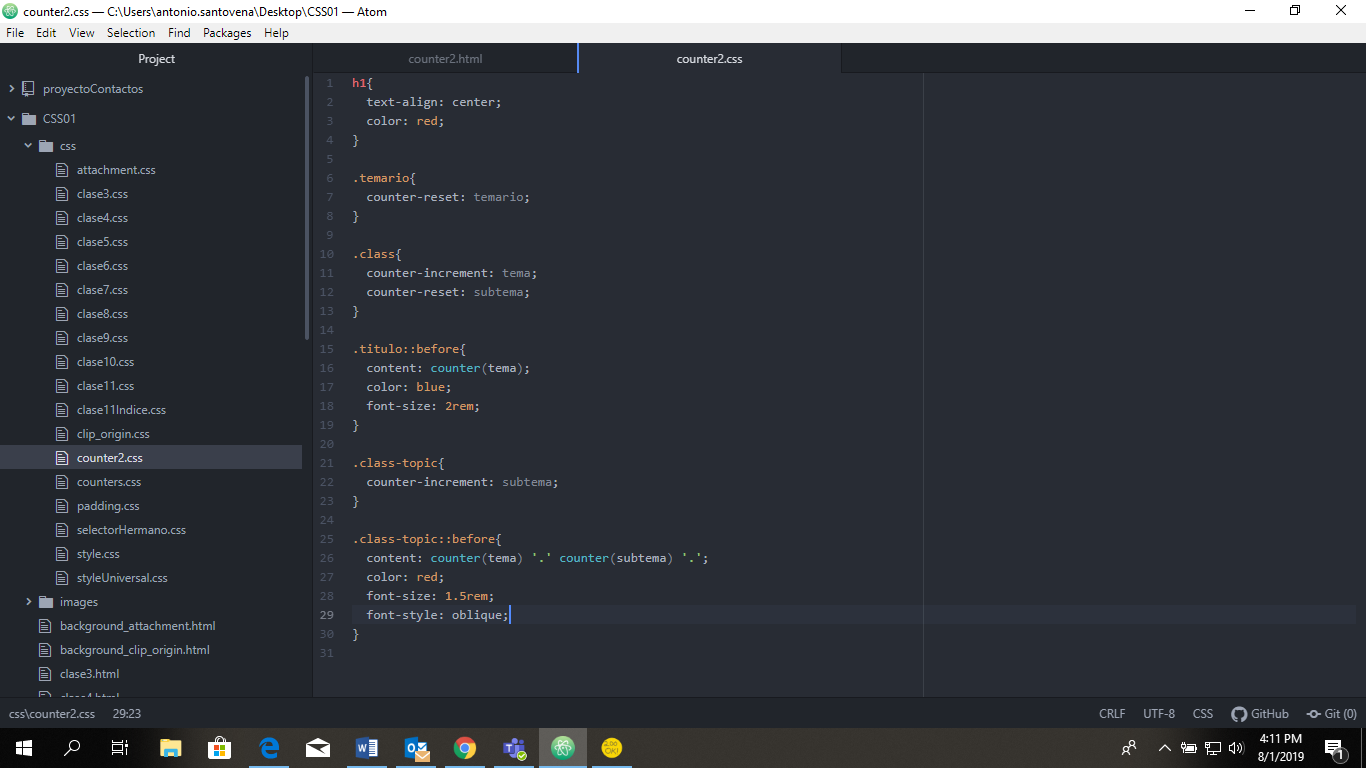
**Counters**

**Con counters podemos personalizar aún mas nuestras listas anidadas utiliznado elementos <ul> y <ol>.**





**Para realizar un índice anidado sin elementos ul u ol:**



**Clase 12 Color**

Color: Color del texto

Background: Color de fondo

Currentcolor: Se aplica el color actual. Si tenemos definido un color, se aplicará el mismo en el background-color.

**Colorkeywords**

Red, yellow, Green son ejemplos de keyword.

Sólo los colores básicos y algunos colores extendidos tienen keyword.