

CSS INTERMEDIO

LIMA UD2 – Tema 1

IES Plurilingüe Antón Losada Diéguez



Tabla de contenido

Tabla de contenido	1
1. Introducción.....	3
2. Tamaños avanzados	3
2.1. Respetar el ratio	3
3. Posicionamiento	3
3.1. Posición relativa	4
3.2. Posición absoluta	4
3.3. Posición fija	5
3.4. Posición pegajosa	6
3.5. Nivel. Índice Z	6
4. Elementos flotantes	7
4.1. Clear.....	8
4.2. Diseño flotante. Float grid	9
5. box-sizing.....	9
6. Desbordamiento	10
6.1. Horizontal o vertical	11
6.2. Corte de palabra.....	11
7. Tipo de caja/elemento	12
8. Alineamiento de imágenes en texto	12
9. Alineamiento de elementos	13
10. Texto avanzado	14
10.1. Transformación de texto	14
10.2. Indentación	15
10.3. Interlineado.....	15
11. Selector de atributo	15
11.1. Definido	15
11.2. Igualdad	16
11.3. Similitud	17
12. Pseudoclases.....	18
12.1. Pseudoclases de enlaces.....	18
12.1.1. Anclado	19
12.2. Foco	19
12.3. Vacío	19

12.4. Pseudoclasas de campos	20
12.5. Orden de hijo	20
12.6. Por tipo de elemento.....	21
13. Sombras.....	22
13.1. Sombra de texto	22
13.2. Sombra de caja	23
13.2.1. Sombra interna	23
13.3. Otros usos de las sombras	24

CSS Medio

1. Introducción

Existen infinidad de propiedades de CSS, por lo que se irán dividiendo en documentos en función de su nivel de conocimientos necesarios, utilidad o complejidad.

En este tema se verán los intermedios. Algunos forman parte de los anteriores o los complementan, mientras que otros son totalmente nuevos.

2. Tamaños avanzados

Como se vio anteriormente, con las propiedades de ancho y alto, se pueden estipular los tamaños de los elementos, pero hay casos en los que esto no es suficiente para adaptarse a las necesidades de la página.

Los tamaños relativos como el porcentual, pueden llegar a hacer que el elemento pierda su estructura o dificulte su lectura. Por esto, existen propiedades para estipular tamaños mínimos y máximos.

Estas son, *min-height* y *min-width* para establecer los valores mínimos y *max-height* y *max-width* para los máximos.

```
div {  
  width: 75%;  
  min-width: 300px;  
  max-width: 1000px;  
}
```

En este ejemplo, el ancho es el 75% del espacio, pero como mínimo ocupará 300px y como máximo 1000px.

Nota, estas propiedades no funcionan con elementos inline.

2.1. Respetar el ratio

Cuando se establecen valores mínimos y máximos combinados con otros, puede ocurrir que se rompa el ratio propio del elemento. Esto en elementos normales no es problemático, pero en imágenes, por ejemplo, puede hacer que se distorsionen.

Para solucionar este problema, basta con poner el tamaño que se distorsiona como *auto*.

3. Posicionamiento

Con HTML, los elementos aparecen en línea y en el orden en el que se estipulan. Con CSS este orden y posicionamiento puede ser modificado.

La propiedad *position* permite estipular el tipo de posicionamiento del elemento en la página.

El valor por defecto de todo elemento es *static*, que lo mantiene en su posición original.

3.1. Posición relativa

Con el valor *relative*, el elemento se posicionará con relación a su posición original, moviéndose desde ahí.

```
.relative {  
    position: relative;  
    left: 30px;  
    top: 25px;  
}
```

Un elemento con `position: relative;` se posiciona de forma relativa a su posición original:

Este div tiene `position: relative;`
Otro elemento que va después del div

Con las propiedades *left*, *top*, *right* y *bottom*, se estipula el desplazamiento tomando como origen ese lado. En este ejemplo, se moverá 30px hacia la derecha y 25px hacia abajo. Estos valores también pueden ser negativos, indicando un desplazamiento en sentido contrario.

Estas propiedades pueden combinarse estipulando un movimiento horizontal y uno vertical, pero nunca combinando dos del mismo tipo.

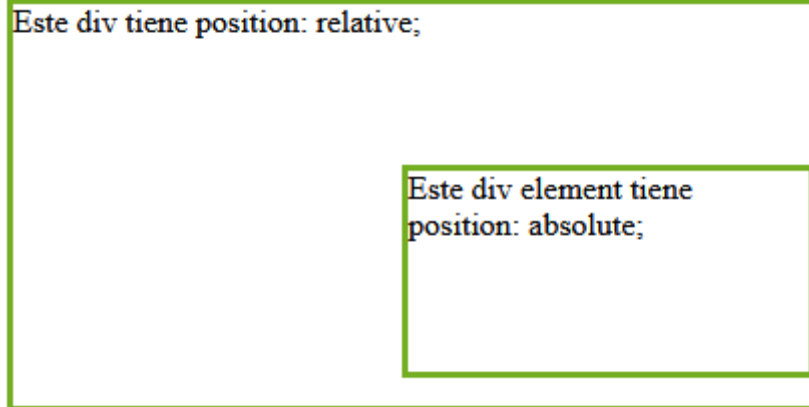
Hay que tener en cuenta que el elemento **sigue ocupando** en su posición original como si estuviese realmente ahí.

Otro factor importante es que se desplaza desde el **exterior de la caja**, es decir, desde el margen, por lo que el tamaño del elemento, sus márgenes interno y externo y hasta su borde afectan a la posición final.

3.2. Posición absoluta

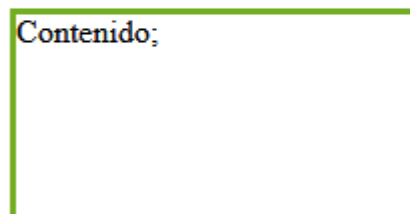
Con el valor *absolute*, el elemento se posicionará con referencia al ancestro más cercano que tenga una posición distinta a *static*.

```
div.relative {  
    position: relative;  
    width: 400px;  
    height: 200px;  
    border: 3px solid #73AD21;  
}  
  
div.absolute {  
    position: absolute;  
    top: 80px;  
    right: 0;  
    width: 200px;  
    height: 100px;  
    border: 3px solid #73AD21;  
}
```



Como se puede ver en el ejemplo, la posición absoluta va en función del tamaño interno del elemento, es decir, su tamaño y el padding.

Hay que tener en cuenta que el elemento **no ocupa**, sale del flujo normal como si no existiese, moviendo todos los elementos que hay debajo. Esto hay que tenerlo en cuenta con referencia al alto del elemento, puesto que sin el posicionamiento absoluto el elemento contenedor se agrandaba para contener este elemento y ahora no.



Al igual que en el caso anterior, se desplaza desde el **exterior de la caja**, desde el margen, por lo que el tamaño del elemento, sus márgenes interno y externo y hasta su borde afectan a la posición final.

3.3. Posición fija

El valor *fixed* es similar al *absolute*, con la salvedad de que se posiciona en relación a la ventana, de tal forma que al desplazarse por la página se mantiene en su sitio.

```
.fixed {
  position: fixed;
  bottom: 0;
  right: 0;
}
```

Al igual que el absoluto, **no ocupa** y se desplaza desde el **exterior de la caja**.

3.4. Posición pegajosa

El valor *sticky* hace que el elemento se comporte como un elemento normal hasta que sale del espacio de la ventana, cuando continúa con el desplazamiento de la página.

```
.sticky {
  position: -webkit-sticky;
  position: sticky;
  top: 5px;
  padding: 5px;
}
```

En este caso, el elemento usará el desplazamiento para estipular cuando se vuelve sticky y dónde se posiciona. En el ejemplo, cuando la ventana se desplace hacia abajo y lo sobrepase, se quedará a 5px del borde superior, pero no hará nada si la ventana lo sobrepasa por arriba o los lados.

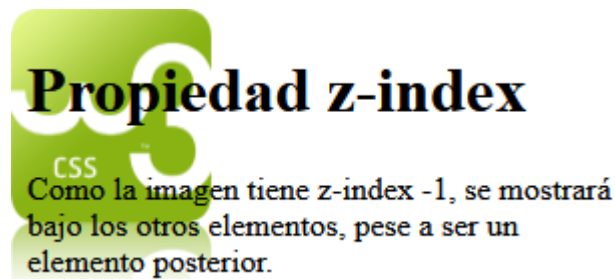
Al igual que en posicionamiento relativo, el elemento **sigue ocupando** en todo momento.

3.5. Nivel. Índice Z

Cuando se utilizan posicionamientos, es posible que los elementos se solapen unos encima de otros de una forma no deseada.

Por defecto, aquellos elementos que se han estipulado posteriormente en el HTML están por encima de los anteriores. Para controlar esto, se utiliza la propiedad *z-index*.

```
img {
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: -1;
}
```



En este ejemplo, los elementos han sido añadidos de la siguiente forma, *h1*, *p* e *img*, por lo que, por defecto, la imagen iría por encima de los otros elementos, pero al haber estipulado el *z-index* como -1, se queda detrás.

El valor puede ser positivo o negativo, colocándose los elementos con valores mayores por encima de los que tienen menores valores. Por defecto, todo elemento tiene un valor de 0. Si dos elementos coinciden, se superpone el que haya sido añadido más abajo en el documento HTML.

Hay que tener en cuenta que aquellos valores **positivos** solo se tienen en consideración si el elemento tiene algún tipo de **posicionamiento**. Si se necesita que un elemento sin posicionamiento aparezca por encima de otro posicionado, habrá que establecer un posicionamiento relativo a ese elemento.

4. Elementos flotantes

La propiedad *float* permite establecer el posicionamiento del elemento hacia el lado izquierdo o derecho de la página, pero con una salvedad, y es que el resto de elementos fluyen y se reposicionan en la página, ocupando el espacio dejado por el elemento.

Esto es muy útil para posicionar imágenes en combinación con textos, pues la imagen se coloca a un lado y el texto la envuelve.

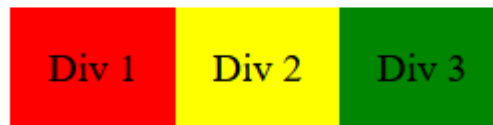
```
img {  
    float: right;  
}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.



Con los `div` u otros elementos en bloque, permite colocarlos uno después del otro, adaptándose al tamaño disponible, de tal forma que, dependiendo del tamaño de la ventana o el dispositivo, se colocan en un lugar u otro.


```
div {  
    float: left;  
}
```



4.1. Clear

Cuando se usan elementos flotantes y se necesita que un elemento no fluya con ellos, se utiliza la propiedad `clear`.

Esta propiedad estipula que el elemento no fluye con los otros elementos flotantes de un tipo específico.

```
.div3 {  
    float: left;  
}  
  
.div4 {  
    clear: left;  
}
```

El valor del `clear` ha de coincidir con el de `float`, de tal forma que se estipula con qué tipo de `float` no fluye, pudiendo ser `left`, `right` o `both`.

Sin clear

div1 div2 - div2 está colocado después del 1 en el HTML. Pero, como div1 tiene `float: left`, su texto fluye alrededor del div 1.

Con clear

div3

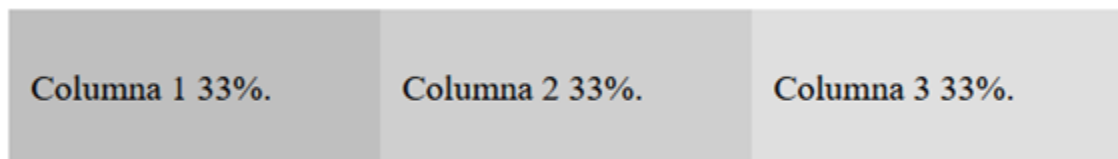
div4 - Aquí, `clear: left`; mueve el div4 debajo del div3 que flota. El valor "left" evita los elementos que flotan a la izquierda.

4.2. Diseño flotante. Float grid

Aprovechando las utilidades de la propiedad *float*, se puede crear un diseño de página estructurado en columnas y adaptativo.

Para ello, se disponen unos elementos en bloque tipo *div* a modo de columnas con un tamaño relativo, de tal forma que la suma de las columnas de el 100% de la página.

```
.column {  
  float: left;  
  width: 33.33%;  
}
```

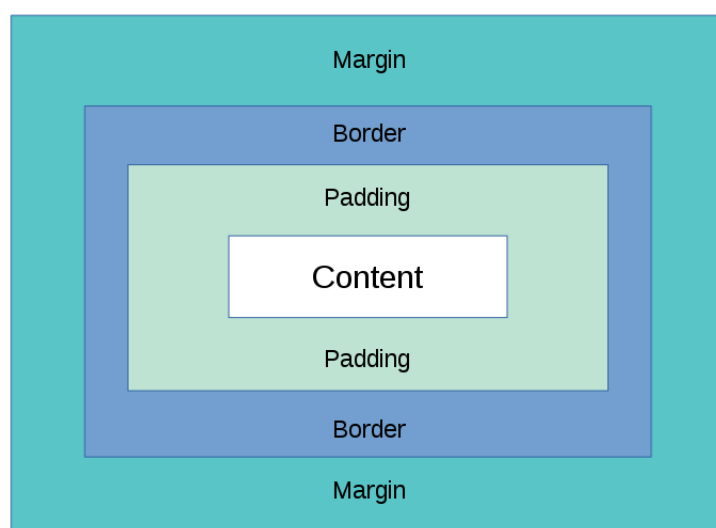


Esto puede adaptarse a las necesidades de la página, poniendo una columna más ancha, márgenes englobando las columnas dentro de otro contenedor, etc.

El problema viene cuando estas columnas tienen propiedades que aumentan su tamaño, como márgenes o bordes, que romperían el diseño.

5. box-sizing

Tal como se vio en el tema anterior, el tamaño real del elemento venía definido por el tamaño que se le estipula al contenido, más su padding, más su borde, lo que puede acarrear que dos elementos, con el mismo tamaño establecido sean de tamaños reales distintos.



Con la propiedad *box-sizing* este comportamiento se puede modificar, de tal forma que si toma el valor *content-box*, que es el valor por defecto, actuará de esa forma.

```


.div1 {
  width: 150px;
  height: 100px;
  border: 1px solid blue;
}



.div2 {
  width: 150px;
  height: 100px;
  padding: 20px;
  border: 1px solid red;
}


```

150x100.

150x100 con 20px de padding.

Por el contrario, si se establece con *border-box*, en el tamaño estipulado se tendrá ya en cuenta el padding y el borde.

```


.div1 {
  width: 150px;
  height: 100px;
  border: 1px solid blue;
  box-sizing: border-box;
}



.div2 {
  width: 150px;
  height: 100px;
  padding: 20px;
  border: 1px solid red;
  box-sizing: border-box;
}


```

150x100.

150x100 con 20px de padding.

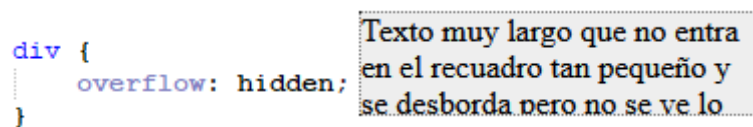
6. Desbordamiento

Cuando el tamaño del contenido de un elemento sobrepasa el tamaño del contenedor, este se desborda, saliendo por fuera del elemento.

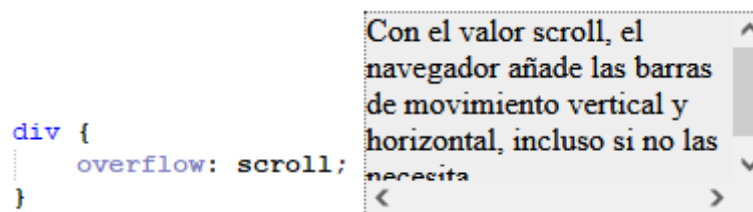
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem.

Para modificar este comportamiento, está la propiedad *overflow*. Su valor por defecto es *visible*, que hace que lo que sobre se vea por fuera.

Con el valor *hidden*, todo lo que sobra se corta, desaparece de la vista.

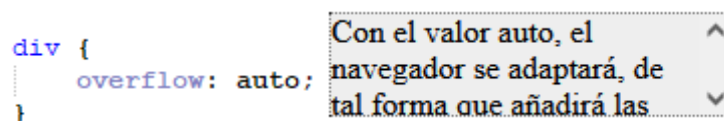


Con el valor *scroll*, el navegador añade las barras de desplazamiento vertical y horizontal para que se pueda ver todo el contenido.



Hay que tener en cuenta que añadirá las dos barras, incluso si no las necesita en absoluto.

Con el valor *auto*, el navegador se adaptará, de tal forma que añadirá las barras cuando las necesite.



6.1. Horizontal o vertical

Con la propiedad *overflow* se puede estipular el comportamiento de forma general, pero podemos estipularlo de forma específica para el horizontal o el vertical usando *overflow-x* y *overflow-y*.

6.2. Corte de palabra

El navegador adapta el texto al tamaño disponible, de tal forma que coloca las palabras en una línea u otra para que entren dentro del elemento.

El problema viene cuando una palabra es más larga que el ancho de la línea, y esta, por defecto, saldrá para fuera del elemento.

Con la propiedad *word-wrap* se puede estipular este funcionamiento. Con *normal* actuará como hasta ahora, mientras que con *break-word* cortará la palabra por dónde considere.

```
div.a {  
  word-wrap: normal;  
}  
  
div.b {  
  word-wrap: break-word;  
}
```

Este div contiene una
palabra muy larga:
largalargalargalargalargalargalarga.
La palabra no se
romperá, saliendo
fuera del elemento.

Este div contiene una
palabra muy larga:
largalargalargalargalar
galargalargalarga. La
palabra se romperá,
poniendo el sobrante
en la siguiente línea.

7. Tipo de caja/elemento

Como se vio en temas anteriores, cada elemento HTML tiene estipulado un tipo de comportamiento por defecto, ya sea *inline* o *block*.

Mediante la propiedad *display*, esto se puede cambiar, poniendo como valor el que se necesite.

```
p {  
    display: inline;  
}  
  
img {  
    display: block;  
}  
  
span {  
    display: inline-block;  
}
```

Con esta propiedad se dispone de un tercer tipo, *inline-block*. Su funcionamiento es igual que el *inline* pero con la salvedad que permite definir su tamaño y todas aquellas propiedades propias de los elementos de tipo *block*.

8. Alineamiento de imágenes en texto

Cuando se combinan imágenes con texto, sobre todo aquellas de pequeño tamaño a modo de icono, su colocación con referencia a la línea del texto es importante. Para ello, con la propiedad *vertical-align* se puede estipular dicho comportamiento.


```
img.a {
    vertical-align: baseline;
}


img.b {
    vertical-align: text-top;
}


img.c {
    vertical-align: text-bottom;
}


img.d {
    vertical-align: sub;
}


img.e {
    vertical-align: super;
}
```

Una  imagen con alineamiento baseline (por defecto).

Una  imagen con alineamiento text-top.

Una  imagen con alineamiento text-bottom.

Una  imagen con alineamiento sub.

Una  imagen con alineamiento super.

Aunque esta propiedad se usa principalmente para alinear imágenes, también puede ser aplicada a otros elementos.

Hay que tener en cuenta que la alineación se hace en referencia a las líneas del texto, base, central, letra más alta, letra que baja más, etc. Para más información, revisar el estándar.

9. Alineamiento de elementos

El alineamiento de elementos es una de las cosas más complicadas en CSS a medida que el número de elementos y reglas aumentan, por lo que existen múltiples alternativas para hacerlo. **(Véase anexo)**

Para un simple elemento, la forma más sencilla es estipularlo como tipo bloque y poner la propiedad de los márgenes izquierdo y derecho a auto.

```
img {
    display: block;
    margin-left: auto;
    margin-right: auto;
}
```



Hay que tener en cuenta que esto solo funciona para los elementos de tipo bloque.

10. Texto avanzado

En el tema anterior se vieron propiedades básicas de textos. En este, se verán algunas más avanzadas para personalizar todavía más los documentos.

Existen muchas más propiedades para casos muy concretos, pero que no se verán dado su carácter extremadamente específico.

10.1. Transformación de texto

Aunque no sería algo propiamente de CSS, con la propiedad *text-transform* se puede modificar el texto para pasarlo a mayúsculas con el valor *uppercase*, a minúsculas con *lowercase* o poner en mayúscula la primera letra de cada palabra con *capitalize*.

```
div.a {  
    text-transform: uppercase;  
}  
div.b {  
    text-transform: lowercase;  
}  
div.c {  
    text-transform: capitalize;  
}
```

text-transform: uppercase:

LOREM IPSUM DOLOR SIT AMET, CONSECTETUR A

text-transform: lowercase:

lorem ipsum dolor sit amet, consectetur adipiscing elit.

text-transform: capitalize:

Lorem Ipsum Dolor Sit Amet, Consectetur Adipiscing Elit.

10.2. Indentación

Una cosa muy habitual en los textos es la indentación de la primera línea de un párrafo. Esto también se puede estipular con CSS mediante la propiedad *text-indent*. El valor puede ser positivo, negativo o relativo.

```
p {  
    text-indent: 50px;  
}
```

Lorem ipsum dolor sit amet, consectetur
 adipiscing elit. Etiam semper diam at erat
 pulvinar, at pulvinar felis blandit. Vestibulum
 volutpat tellus diam, consequat gravida libero
 rhoncus ut.

10.3. Interlineado

Otra propiedad importante en los textos es el interlineado, el espacio entre líneas. En CSS esto se establece mediante la propiedad *line-height*. Este valor se establece mediante valores relativos, absolutos o mediante factor de multiplicación con respecto al tamaño de fuente, tal como hacen los editores de texto.

```
* {  
    line-height: 1.6;  
}
```

Esto es un párrafo con el interlineado recomendado.

Aquí tiene un interlineado de 1.6. Este es un valor sin unidades.

El valor se establece como factor al tamaño de fuente.

11. Selector de atributo

En temas anteriores se vieron los selectores básicos, en este, aquellos referentes a los atributos.

Con estos selectores se pueden seleccionar elementos en base a los valores de sus atributos.

Los selectores de atributo se estipulan con [atributo] o [atributo operando valor].

11.1. Definido

Este es el más básico, simplemente verificamos que dispone de dicho atributo, independientemente de su valor.

```
a[target] {  
    background-color: yellow;  
}
```



```
<a href="https://www.w3schools.com">w3schools.com</a>
<a href="http://www.disney.com" target="_blank">disney.com</a>
<a href="http://www.wikipedia.org" target="_top">wikipedia.org</a>
```

[w3schools.com](https://www.w3schools.com) [disney.com](http://www.disney.com) [wikipedia.org](http://www.wikipedia.org)

11.2. Igualdad

Con este selector escogemos el valor en concreto del atributo.

```
a[target="_blank"] {
    background-color: yellow;
}
```

```
<a href="https://www.w3schools.com">w3schools.com</a>
<a href="http://www.disney.com" target="_blank">disney.com</a>
<a href="http://www.wikipedia.org" target="_top">wikipedia.org</a>
```

[w3schools.com](https://www.w3schools.com) [disney.com](http://www.disney.com) [wikipedia.org](http://www.wikipedia.org)

Hay que tener en cuenta que tiene que tener ese valor exacto, sin extras.

Por otro lado, si queremos que tenga un valor pero nos da igual si tiene más además de ese, antepone la virgulilla.

```
[title~=flower] {
    border: 5px solid yellow;
}
```

```



```



De esta forma, el elemento puede tener la palabra flower entre otras muchas.

Hay que tener en cuenta que funciona a nivel de palabra, si no es una palabra independiente no lo seleccionará. CSS entiende como palabra lo que esté separado por espacio o por -.

Por último, si queremos que esa palabra sea la primera, se antepone una barra vertical |.

```
[class|=top] {
    background: yellow;
}
```

```
<h1 class="top-header">Welcome</h1>
<p class="top-text">Hello world!</p>
<p class="topcontent">Are you learning CSS?</p>
```

Welcome

Hello world!

Are you learning CSS?

11.3. Similitud

Con estos selectores se pueden filtrar los valores de los atributos en base a valores parciales. Estos también funcionan con fragmentos de palabras y no solo palabras completas como los anteriores.

Con el ^ se pueden seleccionar aquellos que empiecen por ese valor.

```
[class^="top"] {
    background: yellow;
}
```

```
<h1 class="top-header">Welcome</h1>
<p class="top-text">Hello world!</p>
<p class="topcontent">Are you learning CSS?</p>
```

Welcome

Hello world!

Are you learning CSS?

Con el \$ se pueden seleccionar aquellos que terminen por ese valor.

```
[class$="test"] {
    background: yellow;
}
```

```
<div class="first_test">The first div element.</div>
<div class="testsecond">The second div element.</div>
<div class="test">The third div element.</div>
<p class="mytest">This is some text in a paragraph.</p>
```

The first div element.
The second div element.
The third div element.

This is some text in a paragraph.

Con el * se pueden seleccionar aquellos que contengan ese valor, ya sea como valor por sí mismo o parcial.

```
[class*="te"] {  
    background: yellow;  
}
```

```
<div class="first_test">The first div element.</div>  
<div class="setend">The second div element.</div>  
<div class="te">The third div element.</div>  
<p class="mytest">This is some text in a paragraph.</p>
```

The first div element.
The second div element.
The third div element.

This is some text in a paragraph.

12. Pseudoclasses

Las pseudoclasses son una característica añadida a un selector que indican un estado o cualidad del afectado.

Estas se estipulan con el selector, : y la pseudoclase todo junto.

12.1. Pseudoclasses de enlaces

Estas pseudoclasses están diseñadas para los elementos *a*.

Con la pseudoclase *:link* se puede seleccionar aquellos enlaces que no hayan sido visitados.

```
a:link {  
    color: red;  
}
```

Por el contrario, con la pseudoclase *:visited*, se seleccionan aquellos que si han sido visitados.

```
a:visited {  
    color: green;  
}
```

Hay que tener en cuenta que si se estipulan ambas, tienen que aparecer en este orden o no funcionarán.

12.1.1. Anclado

La pseudoclase `:target` permite seleccionar aquel elemento referenciado mediante el ancla actual. Es decir, el elemento al cual el navegador lleva tras hacer clic en el ancla.

```
:target {  
    border: 2px solid #D4D4D4;  
    background-color: #e5eccc;  
}
```

12.2. Foco

Estas pseudoclases seleccionan a los elementos cuando el usuario interactúa con ellos.

La pseudoclase `:hover` es la más común, selecciona el elemento cuando el usuario pone el ratón por encima.

```
a:hover {  
    color: #FF00FF;  
}
```

Hay que tener en cuenta que si se usa en un enlace, ha de estar tras las pseudoclases `:visited` y el `:link` si estas estuviesen estipuladas.

Cuando un elemento está seleccionado por el usuario, por ejemplo un campo de un formulario, ese elemento puede ser seleccionado mediante la pseudoclase `:focus`.

```
input:focus {  
    background-color: yellow;  
}
```

Aunque no es muy habitual en los ordenadores, se pueden seleccionar los elementos mediante el tabulador y las flechas y también serían seleccionados por esta pseudoclase.

Cuando se interacciona con un elemento, ya sea mediante un clic de ratón o un enter, la pseudoclase `:active` permite seleccionar dicho elemento. En la mayoría de los casos, la página cambia, por lo que el tiempo en el que es visible está acción suele ser corta.

```
a:active {  
    background-color: yellow;  
}
```

En caso de usarse en los enlaces, esta pseudoclase ha de aparecer tras la `:hover`, `:visited` y `:link` si las hubiera.

12.3. Vacío

Estas pseudoclases seleccionan a los elementos vacíos, sin contenido ni hijos. Existen dos, `:empty` y `:blank`. Su funcionamiento es prácticamente idéntico, con la diferencia de que `blank` permite que tenga espacios vacíos en su interior para considerarlo vacío, mientras que `empty` necesita que no haya nada.

```
div:empty {  
    background-color: lime;  
}
```

Para los campos de los formularios existe otra pseudoclase, *:placeholder-shown* que selecciona aquellos que no tienen contenido.

```
input:placeholder-shown {  
    background-color: tomato;  
}
```

12.4. Pseudoclases de campos

Aunque estas pseudoclases pueden llegar a afectar a otros elementos más allá de los de los formularios, es su uso más habitual.

Con la pseudoclase *:enabled* se seleccionan aquellos elementos habilitados, mientras que con *:disabled* aquellos que están deshabilitados.

```
input:disabled {  
    background-color: red;  
}
```

Por otro lado, la pseudoclase *:required* selecciona los elementos estipulados como obligatorios.

```
input:required {  
    background-color: wheat;  
}
```

12.5. Orden de hijo

Estas pseudoclases permiten seleccionar los elementos según su posición como hijos de otra. Suelen ser utilizadas mucho para dar formato a tablas.

Las pseudoclases *:first-child* y *:last-child* permiten seleccionar el primer hijo y el último respectivamente.

```
p:first-child {  
    color: blue;  
}
```



```
<body>  
<p>This is some text.</p>    This is some text.  
<p>This is some text.</p>  
<div>                        This is some text.  
    <p>This is some text.</p>  
    <p>This is some text.</p>    This is some text.  
</div>  
</body>                      This is some text.
```

Las pseudoclase *:nth-child(n)* y *:nth-last-child(n)* permiten seleccionar el elemento número n empezando desde arriba o desde abajo respectivamente.

```
p:nth-child(2) {  
    background: red;  
}
```

The first paragraph.

The second paragraph.

`<p>The first paragraph.</p>`

`<p>The second paragraph.</p>`

The third paragraph.

`<p>The third paragraph.</p>`

`<p>The fourth paragraph.</p>`

The fourth paragraph.

Esta pseudoclase permite establecer una fórmula matemática, por ejemplo, $4n$ permite seleccionar el 4, 8, 12...

Si en vez de un número se estipula *even* u *odd*, seleccionará los pares e impares respectivamente. Esto es equivalente a $2n$ y $2n+1$.

12.6. Por tipo de elemento

Las pseudoclases *:first-child* y *:last-child* solo permiten estipular por orden, independientemente del tipo de elemento. Tan solo se fijan en si es el primer o último hijo de alguien respectivamente.

```
p:last-child {  
    color: blue;  
}
```

<code><body></code>	
<code><p>P Hijo 1 de body</p></code>	P Hijo 1 de body
<code><p>P Hijo 2 de body</p></code>	P Hijo 2 de body
<code><div></code>	
Div Hijo 3 de body	Div Hijo 3 de body
<code><p>P Hijo 1 de div</p></code>	P Hijo 1 de div
<code><p>P Hijo 2 de div</p></code>	P Hijo 2 de div
<code></div></code>	
<code></body></code>	

En este ejemplo, el único *p* que es el último elemento de alguien es el segundo que está dentro del *div*, ya que el último elemento del *body* es el *div* y no el *p*.

Por el contrario, si se necesita seleccionar el primero o el último elemento de un tipo dentro de cada elemento, se utilizarían las pseudoclases *:first-of-type* y *:last-of-type*.

```
p:last-of-type {  
    color: blue;  
}
```

P Hijo 1 de body

P Hijo 2 de body

Div Hijo 3 de body

P Hijo 1 de div

P Hijo 2 de div

Como se puede ver en este ejemplo, estas pseudoclasas buscan por el primero y último de su tipo, independientemente de si son el primer o último elemento dentro de otro.

13. Sombras

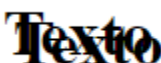
CSS dispone de propiedades para aplicar sombras a los elementos y diferencia entre dos, para textos y elementos de caja.

13.1. Sombra de texto

Con la propiedad *text-shadow* se puede estipular la sombra del texto. Esta tiene dos valores obligatorios, desplazamiento vertical y desplazamiento horizontal.

Estos valores estipulan el movimiento de la sombra hacia la derecha y abajo y permiten valores absolutos tanto positivos como negativos para moverse en dirección contraria.

```
h1 {  
    text-shadow: 5px 5px;  
}
```




Como se puede observar, la sombra es idéntica al texto, con su misma fuente y tamaño.

A mayores, tiene dos valores opcionales, el *blur* o difuminado y el color.


Por defecto, el color es el mismo que el texto, pero se puede modificar poniéndolo en última posición.

```
h1 {  
    text-shadow: 5px 5px tomato;  
}
```



El otro valor opcional, el *blur*, permite difuminar la sombra, haciéndolo borrosa. Este se coloca antes del color y después de los desplazamientos.

```
h1 {  
    text-shadow: 5px 5px 2px tomato;  
}
```



Como puede observarse, el difuminado le da un efecto de sombra más realista.

Si se necesita añadir más de una sombra, se pueden poner los conjuntos de valores separados por comas.

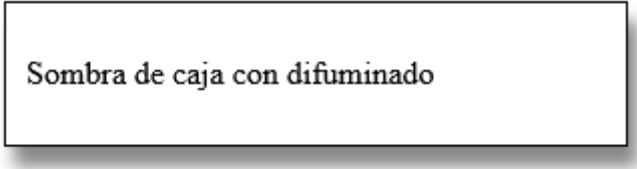
```
h1 {  
    text-shadow: 5px 5px 2px tomato,  
                5px 3px 4px wheat;  
}
```

13.2. Sombra de caja

La propiedad *box-shadow* es la sombra de la caja y se aplica a los elementos en sí, respetando su tamaño y redondez de bordes.

Su uso es similar al de texto, pero tiene varios valores opcionales a mayores.

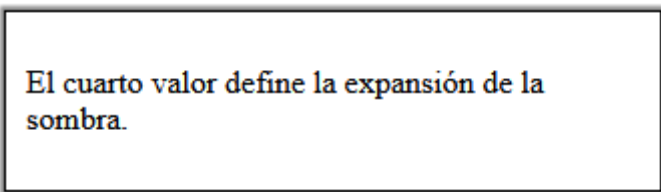
```
#shadow {  
    box-shadow: 5px 10px 8px #888888;  
}
```



Sombra de caja con difuminado

Esta propiedad dispone de un valor *spread*, colocado en cuarta posición, que estipula lo mucho que se aumenta o reduce la sombra con respecto al tamaño original del elemento.

```
#dilatada {  
    box-shadow: 0px 0px 1px 2px #888888;  
}
```



El cuarto valor define la expansión de la sombra.

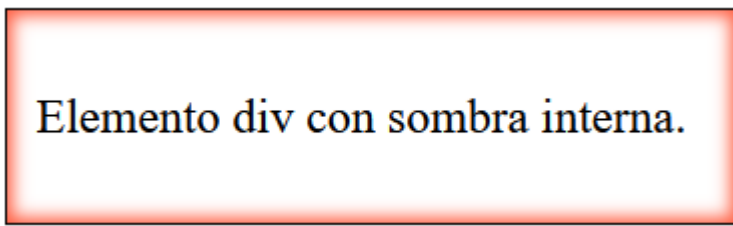
En este ejemplo, la sombra está en la posición 0, pero ocupa 2px más.

13.2.1. Sombra interna

La sombra de caja permite establecerse de forma interna, es decir, dentro de la caja en vez de fuera.

Para ello, se le pone el valor *inset* al principio o al final de los valores de la propiedad.

```
#sombra {  
    box-shadow: inset 0px 0px 5px 2px tomato;  
}
```



Elemento div con sombra interna.

Hay que tener en cuenta que, por defecto, su tamaño es igual que el elemento, por lo que no se verá sombra alguna si no se desplaza. Para crear una sombra uniforme por todos lados, será necesario estipular un *blur* y/o un *spread*.

13.3. Otros usos de las sombras

Las sombras pueden utilizarse para generar todo tipo de efectos visuales, como efecto 3D, neón o grabados, entre otros muchos.