Selectores CSS y CSS3

1. * (Selector “universal”)

```
* {
 margin: 0;
 padding: 0;
}
```

El asterisco selecciona todos los elementos. Muchos desarrolladores usarán este truco para poner a cero el `margin` y `padding`. .

El `*` puede ser usado también con selectores descendentes.

```
#container * {
 border: 1px solid black;
}
```

Esto se centrará en cada uno de los elementos que sea hijo del `#container`.

2. #X (Selector de id)

```
#miid {
 width: 960px;
 margin: auto;
}
```

Prefijar una almoadilla a un selector nos permite seleccionar por `id`. Éste es fácilmente el uso más común, sin embargo ten cuidado cuando uses selectores `id` .

> *Pregúntate a ti mismo: ¿necesito absolutamente aplicar un `id` a este elemento para afectarlo?*

Los selectores `id` son rígidos y no pueden ser re-utilizados ya que son “propiedad” de un sólo elemento.

Usa `id` para “encontrar una aguja en un pajar” y estilizar sólo ése elemento específicamente.

3. .X (Selector de clase)

```
.miclase {
color: red;
}
```

Éste es un selector de clase `class` La diferencia entre `id`s y `class` es que, con el último, puedes seleccionar varios elementos. Usa `class` cuando quieras aplicar un estilo a varios elementos.

Además no olvidemos que podemos aplicar varias clases a un solo elemento pudiendo, de esta manera, estilizarlo de una manera muy flexible.

4. X Y (Selector descendente)

```
li a {
text-decoration:none;
}
```

El siguiente selector más común es el selector `descendente`. Cuando necesites ser más específico con tus selectores, usas éstos. Por ejemplo, ¿que tal sí, en lugar de seleccionar todas las etiquetas <a>, sólo necesitas los <a> que están dentro de una ?. Aquí es cuando usarías específicamente un selector descendiente.

> *Consejo - Si tu selector luce como `X Y Z A B .miclase`, lo estás haciendo mal. Siempre pregúntate si es absolutamente necesario aplicar todo ese peso y complejidad.*

5. [ul, div etc.] (Selector de etiqueta)

```
a { color: red; }
ul { margin-left:0; }
```

¿Qué pasa si queremos seleccionar todos los elementos en una página, de acuerdo a su tipo (`type`), en vez de un nombre de `id` o `clase`?. Trataremos de hacerlo con simplicidad usando un selector de etiqueta. Por ejemplo, si necesitas seleccionar todas las <ul>, usa `ul {}`.

6. X:link, X:visited, X:hover, X:active

```
a:link { color: red; }
a:visited { color: purple;
}
a:hover { color: red; }
a:active { color: red; }
```

Usamos la pseudo clase `:link` para seleccionar todas las etiquetas <a> a las que aún no se les ha dado click.

De manera alternativa, también tenemos la pseudo-clase `:visited` que nos permite aplicar un estilo específico sólo a las etiquetas <a> que han sido visitadas.

Utilizaremos la pseudo-clase `:hover` cuando queremos aplicar un estilo específico a un elemento al pasar el ratón por encima.

La pseudo-clase `:active` se aplicará cuando pulsemos sobre un elemento hasta que lo soltemos.

> *Tip - Love & Hate. Debemos poner estas pseudo clases en este orden: link,visited,hover y active.*

7. X + Y (Selector adyacente)

```
ul + p {
 color:red;
}
```

Este se conoce como un selector adyacente. Seleccionará solamente el elemento que está inmedatamente después del primer elemento. En éste caso, solo el primer <p> después de cada <ul> tendrá texto rojo.

8. X > Y (Selector “hijos directos”)

```
div#container > ul {
border: 1px solid
black;
}
```

La diferencia entre el estándar `X Y` y `X > Y` es que el último sólo seleccionará hijos directos. Por ejemplo, considera el siguiente código:

```
<div id="container">
    <ul>
        <li> List Item
          <ul>
              <li> Child </li>
          </ul>
        </li>
        <li> List Item </li>
        <li> List Item </li>
        <li> List Item </li>
    </ul>
</div>
```

Un selector de `#container > ul` solo afectará los <ul> que sean hijos directos del con `id` de `container`. No afectará, por ejemplo, al <ul> que es hijo del primer <li>,

Es recomendable particularmente cuando se trabaja con motores de selectores CSS basados en JavaScript.

9. X ~ Y (Selector adyacente generalizado)

```
ul ~ p {
 color:
red;
}
```

Es similar a `X + Y`, pero menos estricto. Mientras que un selector adyacente (`ul + p`) sólo selecciona el primer <p> que está inmediatamente seguido de un <ul> éste seleccionará todos los <p> que estén inmediatamente seguidos de un <ul>,

10. X[nombre atributo] (Selector de atributo)

```
a[title] {
 color:green;
}
```

Denominado como un selector de atributos, en nuestro ejemplo de arriba, esto sólo seleccionará las etiquetas <a> que tengan un atributo `title`. Las etiquetas <a> que no lo tengan no recibirán éste estilo en particular.

11. X[href="valor"] (Selector de atributo especificando el valor de ese atributo)

```
a[href=""] {
color: #1f6053;
}
```

El fragmento de arriba dará estilo a todas las etiquetas <a> que enlacen a “http://pepito.com”. Las demás etiquetas <a> permanecerán sin cambio.

> *Ten en cuenta que estamos encerrando el valor entre comillas.*

12. X[href*="valor"] (Selector de atributo especificando una cadena que debe estar incluída en el valor de ese atributo)

```
a[href*="pepi"] {
color: #1f6053; /* nettuts green
*/
}
```

El asterisco designa que el valor especificado a continuación debe aparecer en algún lugar del valor del atributo. De esa manera, el ejemplo propuesto cubriría “http://pepito.com”, “http://pepito.com/clientes/murcia.html”... “http://pepito.com/productos.html”...

13. X[href^="valor"] (Selector de atributo especificando una cadena que debe estar al principio del valor de ese atributo)

```
a[href^="http"] {
 background: url(enlace externo/icon.png) norepeat;
 padding-left: 10px;
}
```

¿Alguna vez te has preguntado cómo es que algunos sitios son capaces de desplegar un pequeño icono junto a los enlaces que son externos? Estoy seguro de que los has visto antes; son buenos recordatorios de que el enlace te dirigirá a un sitio web totalmente diferente.

Esto es un juego de niños con el símbolo de intercalación. Es más comúnmente usado en expresiones regulares para designar el inicio de una cadena de texto. Si queremos afectar todas las etiquetas <a>que tienen un `href` que comienza con `http`, podríamos usar un selector similar al fragmento mostrado arriba.

> *Observa que no estamos buscando `http://`; eso es innecesario y no cuenta para las urls que comienzan con `https://`.*

14. X[href$="valor"] (Selector de atributo especificando una cadena que debe estar al final del valor de ese atributo)

```
a[href$=".com"]
{
 color: red;
}
```

De nuevo, usamos un símbolo de expresiones regulares, `$`, para referirnos al final de una cadena de texto. En éste caso, estamos buscando todos los <a> que enlacen a una url que termine en ".com".

15. X[data-*="valor"] (Selector de atributo personalizado: "data-nombreatributopersonalinventado=valorinventado")

```
a[data-imagenes="image"]
{
 color: red;
}
```

¿Cómo podríamos (por ejemplo) crear un selector para los diferentes tipos de imagen: `png`, `jpeg`,`jpg`, `gif`?... podríamos crear múltiples selectores, tal como:

```
a[href$=".jpg"],a[href$=".jpeg"],a[href$=".png"],
a[href$=".gif"] {
 color: red;
}
```

Pero, esto es un selector demasiado complicado e ineficiente. Otra posible solución es usar atributos personalizados. ¿Y si agregáramos nuestro propio atributo `data-imagenes` para cada <a> que enlaza a una imagen?

> *La sintáxis es muy simple: cualquier atributo en cualquier elemento que comience por "data-" será considerado como atributo. Esto se puede resumir en que tenemos la capacidad de crear nuestros propios atributos: data-imagenes, data-pepito etc.*

```
<a href="path/to/image.jpg" data-imagenes="imagen"> Image Link
</a>
```

Entonces, dicho esto, dicho esto, podemos usar un selector de atributos estándar para afectar sólo a esos <a>:

```
a[data-imagenes="imagen"]
{
 color: red;
}
```

16. X[nombreatributo~="valor"] (Selector de atributo, con varios valores, especificando uno de ellos)

Este selector es bastante especial. No mucha gente lo conoce. El símbolo (`~`) nos permite afectar un elemento que tenga una lista de valores separados por espacios especificando uno de ellos.

Siguiendo con nuestro atributo personalizado del número 15, arriba, podríamos crear un atributo `data-info` el cual puede recibir una lista separada por espacios de lo que sea que necesitemos especificar. En éste caso, especificaremos enlaces externos y enlaces a imágenes :

```
"<a href="path/to/image.jpg" data-info="external image"> Click Me, Fool
</a>
```

Con ése código en su lugar, ahora podemos seleccionar cualquier etiqueta que tenga cualquiera de esos valores, usando el truco de selector de atributos con ~.

```
/* Selecciona elementos cuyo atributo “data-info” lleva el valor
"external" */

a[data-info~="external"] {
 color: red;
}

/* Y los que contienen el valor "image" */
a[data-info~="image"] {
border: 1px solid black;
}
```

17. X:checked

```
input[type=radio]:checked
{
 border: 1px solid black;
}
```

Ésta pseudo clase sólo afectará a un elemento de interfaz de usuario (Formularios) que haya sido seleccionado como un botón de opción (radio button) o casilla de selección (checkbox). Tan sencillo como eso.

18. X:after, X:before ó X::after, X::before

Los pseudo elementos `before` y `after` son lo máximo. Todos los días, al parecer, la gente está encontrando nuevas y creativas formas para usarlas de manera efectiva. Éstas simplemente generan contenido alrededor del elemento seleccionado.

> *De acuerdo con las especificaciones de Selectores CSS3, técnicamente deberías usar la sintaxis del pseudo elemento compuesto de dos "::". Sin embargo, para permanecer compatible, el navegador aceptará un solo ":" también. De hecho, en éste punto, es más inteligente un solo ":"en tus proyectos.*

```
p::after {
 content: " - Recuerda ésto";
}
```

```
p::before {
 content: "Recuerda ésto: ";
}
```

Podemos utilizar ***https://fontawesome.com/*** para poner un icono delante o detrás de un elemento HTML:

```
p.email:before, p.mobile:before {
 font-family: FontAwesome;
 display: inline-block;
 padding-right: 6px;
 vertical-align: middle;
}
```

```
p.email:before {
 content: "\f095";
}
```

```
p.mobile:before {
 content: "\f003";
```

Este código irá en el head y lo hemos conseguido en ***https://fontawesome.com/***

<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.6.3/css/solid.css" integrity="sha384" crossorigin="anonymous">

```
<div class="container">
    <div class="row wrapper">
       <p class="mobile">Mobile</p>
       <p class="email">Email</p>
    </div>
</div>
```

19. X.miclase

Cuando queremos seleccionar, por ejemplo, a todos los <p> de una determinada clase (por ejemplo "miclase") utilizaremos este selector:

```
p.miclase {
 margin-right: 5px;;
}
```

Esto no afectaría, por ejemplo, a los de la clase "miclase".

Hay que notar la diferencia entre este selector y este otro:

```
p .miclase {
 margin-right: 5px;
}
```

Este último afectaría a todos los elementos cuya clase sea “miclase” que estén dentro de una etiqueta <p>. La diferencia que marca ese espacio es notable.

20. X:not(selector)

```
div:not(#container)
{
 color: blue;
}
```

La pseudo clase `not` es particularmente útil. Digamos que quiero seleccionar todos los excepto los que tienen un `id` de `container`. El fragmento de arriba manejará la tarea perfectamente.

O, si hubiera querido seleccionar todos los elementos (no aconsejable) excepto las etiquetas <p>, podríamos hacer:

```
*:not(p) {
color:
green;
}
```

21. X::pseudoElemento

```
p::first-line {
 font-weight: bold;
 font-size: 1.2em;
}
```

Podemos usar pseudo elementos (designados por `::`) para estilizar fragmentos de un elemento, como la primera línea, o la primera letra. Ten en mente que estos deben ser aplicados a elementos de bloque para que funcione.

> *Un pseudo-elemento está definido por dos puntos dobles : “`::`”*

Afecta La Primer Letra De Un Párrafo

```
p::first-letter {
 float: left;
 font-size: 2em;
 font-weight: bold;
 font-family: cursive;
 padding-right: 2px;
}
```

Este fragmento de código selecconará todos los párrafos en la página, y luego afectará sólo la primer letra de ése elemento.

Esto es más usado para crear estilos como el de los periódicos para la primer letra de un artículo.

Afecta La Primer Linea De Un Párrafo

```
p::first-line {
 font-weight:
bold;
 font-size: 1.2em;
}
```

Similarmente, el pseudo elemento `::first-line` estilizará, como es esperado, sólo la primer línea del elemento.

> *Para compatibilidad con hojas de estilo existentes, los navegadores deben aceptar también la notación previa de un “dos puntos” para pseudo-elementos introducidos en niveles 1 y 2 de CSS (es decir, :first-line, :first-letter, :before and :after).Esta compatibilidad no está permitida para los nuevos pseudo-elementos introducidos en ésta especificación CSS3.”*

22. X:nth-child(n)

```
li:nth-child(3)
{
 color: red;
}
```

¿Recuerdas los días cuando no teníamos manera de afectar elementos específicos en un montón? ¡La pseudo clase `nth-child` resuelve eso!

Observa que `nth-child` acepta un número entero como parámetro empezando por el 1. Si quieres afectar el segundo elemento de una lista, usa `li:nth-child(2)`.

También podemos introducir como parámetro “even” designando elementos pares y “odd” designando elementos impares. Esto es muy utilizado en tablas para dar colores alternativos a las filas <tr>.

```
tr:nth-child(even) {
 background-color: red;
}
```

23. X:last-child

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

Este selector colorearía de rojo a los elementos <p> que sean el último hijo de su parent.

24. X:first-child

```
ul li:first-child
{
```

```
  border-top: none;
}
```

Esta pseudo clase nos permite afectar sólo al primer hijo del padre del elemento. Lo usaremos frecuentemente, por ejemplo, para eliminar bordes de los primeros y últimos elementos de una lista.

Ejemplo

Contruyamos un ejemplo simple para demostrar uno posible uso de estas clases. Crearemos un elemento de lista estilizado.

Código

```
<ul>
    <li> List Item </li>
    <li> List Item </li>
    <li> List Item </li>
</ul>
```

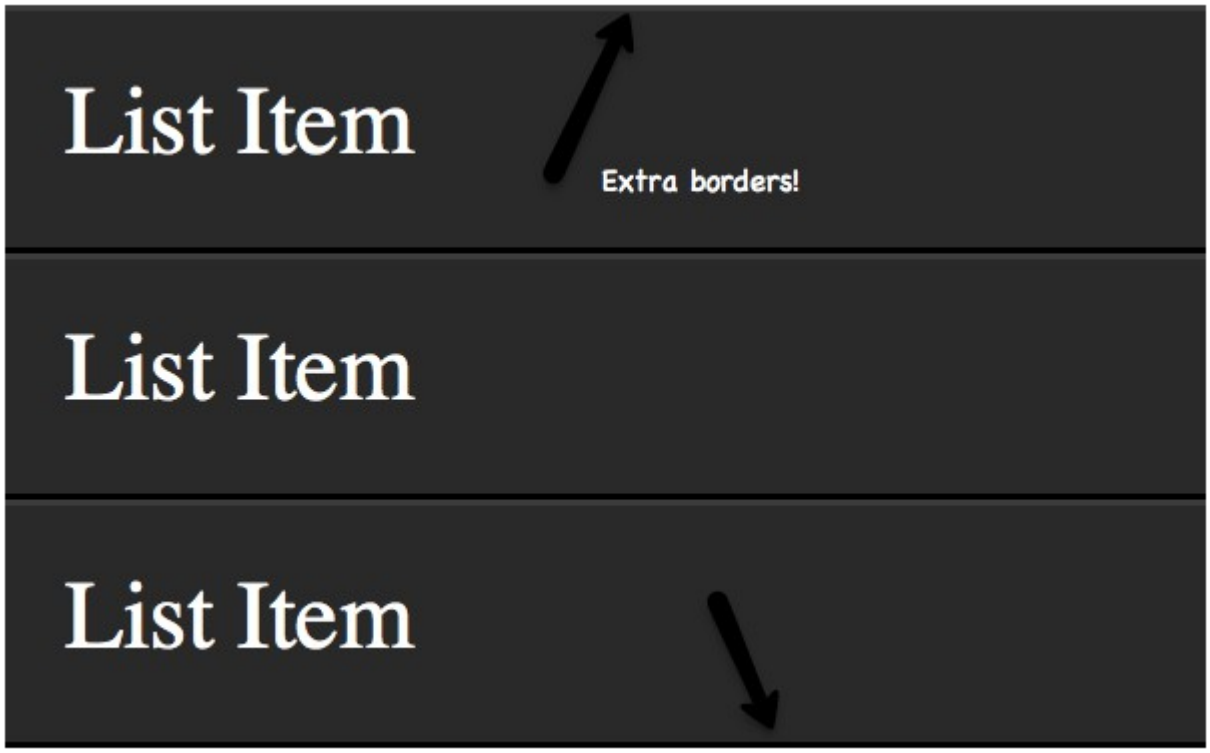
Nada especial aquí, sólo una simple lista.

CSS

```
ul {
  width: 200px;
  background: #292929;
  color: white;
  list-style: none;
  padding-left: 0;
}

li {
  padding: 10px;
  border-bottom: 1px solid black;
  border-top: 1px solid #3c3c3c;
}
```

Éste estilo fijará un fondo, eliminará el padding por defecto del navegador en losn y aplicará bordes en los <li> para conseguir un poco de profundidad.

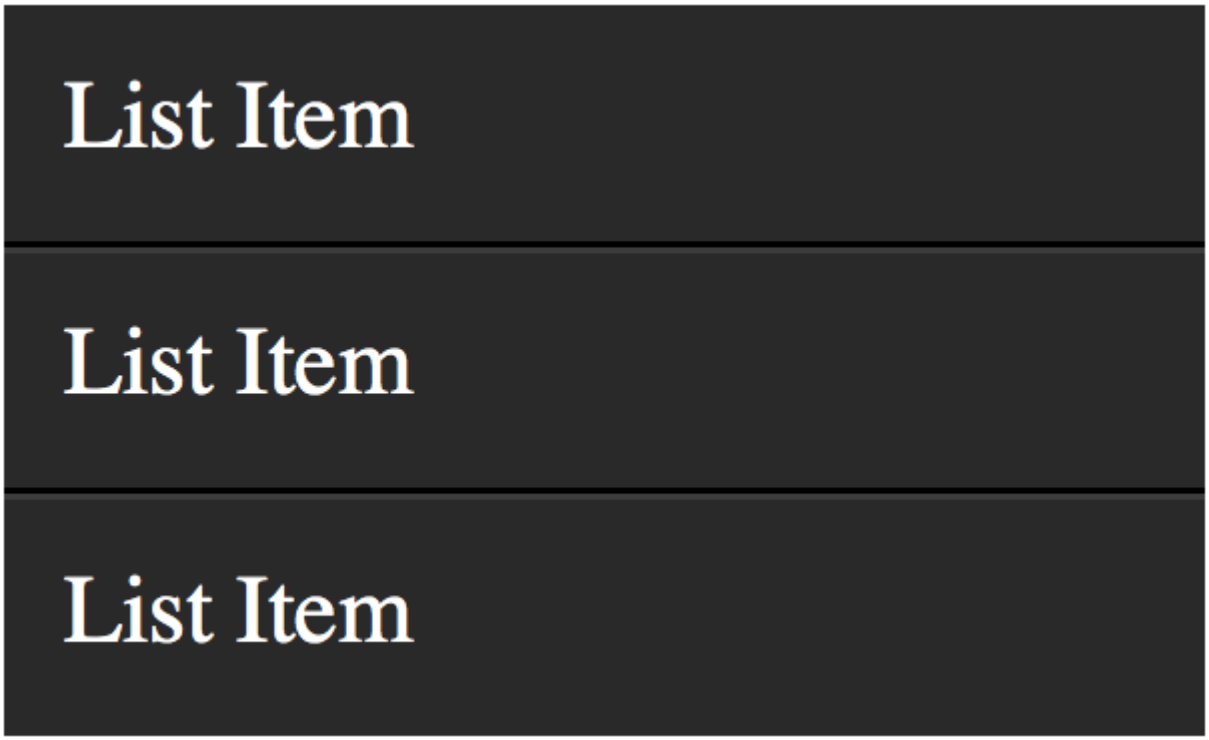


Para agregar profundidad a tus listas, aplica un borde inferior `border-bottom` *a cada que sea un tono o dos más oscuros que el color de fondo del<li>. Después, aplica un borde superior* `border-top` *que sea un par de tonos más claro.*

El único problema, como se muestra en la imagen superior, es que el borde será aplicado la parte superior e inferior de la lista lo cual luce un poco extraño. Usemos las pseudo clases `:first-child` y `:last-child` para arreglar esto.

```
li:first-child {
  border-top: none;
}

li:last-child {
 border-bottom: none;
}
```



¡Perfecto!. Eso lo arregla!