

# Los 30 selectores CSS que debes memorizar

*by Jeffrey Way* 9 Jun 2011

20-26 minutes

---

¿Así que aprendiste los selectores básicos `id`, `class`, y `descendant` – y ahora te retiras? Si es así, te estás perdiendo un enorme nivel de flexibilidad. Mientras que muchos de los selectores mencionados en éste artículo son parte de CSS3, y están por tanto, sólo disponibles en navegadores modernos, te debes a ti mismo el compromiso de memorizarlos.

## 1. \*

```
1 * {  
2   margin: 0;  
3   padding: 0;  
4 }
```

Comencemos con los más obvios, para los principiantes, antes de continuar con selectores más avanzados.

El asterisco se centrará en cada uno de los elementos en la

página. Muchos desarrolladores usarán éste truco para poner a cero el `margin` y `padding`. . Mientras que esto está ciertamente bien para pruebas rápidas, yo te recomendaría que nunca uses esto en tu código final. Esto añade mucho *peso* en el navegador y es innecesario.

El `*` puede ser usado también con selectores de hijos.

```
1 #container * {  
2   border: 1px solid black;  
3 }
```

Esto se centrará en cada uno de los elementos que sea hijo del `#container` `div`. . De nuevo, trata de no usar mucho ésta técnica, si no es que nunca.

[Ver Demo](#)

## Compatibilidad

- IE6+
- Firefox
- Chrome
- Safari
- Opera

## 2. #X

```
1 #container {  
2   width: 960px;
```

```
3 margin: auto;
```

```
4 }
```

Prefijar el símbolo numérico a un selector nos permite seleccionar por `id`. T. Éste es fácilmente el uso más común, sin embargo te 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. De ser posible, primero trata de usar un tag name, uno de los nuevos elementos en HTML5, incluso una pseudo-clase.

[Ver Demo](#)

## Compatibilidad

- IE6+
- Firefox
- Chrome
- Safari
- Opera

## 3. .X

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

aplique a un grupo de elementos. De manera alternativa, usa `ids` para encontrar una aguja en un pajar y estilizar sólo ése elemento en específico.

[Ver Demo](#)

## Compatibilidad

- IE6+
- Firefox
- Chrome
- Safari
- Opera

## 4. X Y

```
1 li a {  
2 text-decoration: none;  
3 }
```

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

**Pro-tip** - Si tu selector luce como `X Y Z A B.error`, you're doing it wrong. lo estás haciendo mal. Siempre

pregúntate si es absolutamente necesario aplicar todo ese peso.

[Ver Demo](#)

## Compatibilidad

- IE6+
- Firefox
- Chrome
- Safari
- Opera

## 5. X

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

```
2 ul { margin-left: 0; }
```

¿Qué pasa si quieres seleccionar todos los elementos en una página, de acuerdo a su tipo ( `type`), en vez de un nombre de `id` o `clase`. Mantenlo simple, y usa un selector de tipo. Si necesitas seleccionar todas las unordered lists, usa `ul {}`.

[Ver Demo](#)

## Compatibilidad

- IE6+
- Firefox
- Chrome

- Safari
- Opera

## 6. X:visited and X:link

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

```
2 a:visited { color: purple; }
```

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

De manera alternativa, también tenemos la pseudo-clase `:visited`, la cual, tal como esperabas, nos permite aplicar un estilo específico sólo a las etiquetas anchor en la página *a las que se les ha dado click*, o han sido *visitadas*.

[Ver Demo](#)

### Compatibilidad

- IE7+
- Firefox
- Chrome
- Safari
- Opera

## 7. X + Y

Este se conoce como un selector adyacente. Seleccionará *solamente* el elemento que es inmediatamente precedido por el primer elemento. En éste caso, solo el primer párrafo

después de cada `ul` tendrá texto rojo.

[Ver Demo](#)

## Compatibilidad

- IE7+
- Firefox
- Chrome
- Safari
- Opera

## 8. X > Y

```
1 div#container > ul {  
2 border: 1px solid black;  
3 }
```

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

```
01 <div id="container">  
02     <ul>  
03         <li> List Item  
04             <ul>  
05                 <li> Child </li>  
06             </ul>
```

```
07         </li>
08         <li> List Item </li>
09         <li> List Item </li>
10         <li> List Item </li>
11     </ul>
12</div>
```

Un selector de `#container > ul` solo afectará los `ul`s que sean hijos directos del `div` con `id` de `container`. No afectará, por ejemplo, el `ul` que sea hijo del primer `li`.

Por ésta razón, hay beneficios de desempeño por usar el elemento de hijo. De hecho, es recomendable particularmente cuando se trabaja con motores de selectores CSS basados en JavaScript.

[Ver Demo](#)

## Compatibilidad

- IE7+
- Firefox
- Chrome
- Safari
- Opera

## 9. $X \sim Y$

Este elemento “hermano” es similar a  $X + Y$ , sin embargo,



es menos estricto. Mientras que un selector adyacente (`ul + p`) sólo selecciona el primer elemento que es *inmediatamente* precedido por el primer selector, éste es más generalizado. Seleccionará, refiriéndonos al ejemplo de arriba, cualquier elemento `p`, siempre y cuando estén después de un `ul`.

[Ver Demo](#)

## Compatibilidad

- IE7+
- Firefox
- Chrome
- Safari
- Opera

## 10. `X[title]`

```
1 a[title] {  
2   color: green;  
3 }
```

Denominado como un *selector de atributos*, en nuestro ejemplo de arriba, esto sólo seleccionará las etiquetas `anchor` que tengan un atributo `title`. Las etiquetas `anchor` que no lo tengan no recibirán éste estilo en particular. Pero, ¿y qué si necesitas ser más específico? Bueno...

[Ver Demo](#)

## Compatibilidad

- IE7+
- Firefox
- Chrome
- Safari
- Opera

## 11. X[href="foo"]

El fragmento de arriba dará estilo a todas las etiquetas anchor que enlacen a *http://net.tutsplus.com*; estas recibirán nuestro característico color verde. Las demás etiquetas anchor permanecerán sin cambio.

Ten en cuenta que estamos encerrando el valor entre comillas. Recuerda hacer esto también cuando uses un motor de selectores CSS JavaScript. Cuando sea posible, siempre usa selectores CSS3 en vez de métodos no oficiales.

esto funciona bien, aunque, es un poco rígido. ¿Que tal si el enlace dirige ciertamente a Nettuts+, pero, tal vez, la dirección es *nettuts.com* en lugar de la url completa? En esos casos, podemos usar un poco de sintaxis de expresiones regulares.

[Ver Demo](#)

## Compatibilidad

- IE7+

- Firefox
- Chrome
- Safari
- Opera

## 12. X[href\*="nettuts"]

```
1 a[href*="tuts"] {  
2 color: #1f6053; /* nettuts green */  
3 }
```

Aquí vamos; eso es lo que necesitamos. El asterisco designa que el valor a continuación debe aparecer *en algún lugar* del valor del atributo. De esa manera, esto cubre *nettuts.com*, *net.tutsplus.com*, e incluso *tutsplus.com*.

Ten en mente que esta es una declaración abierta. ¿Qué si la etiqueta anchor enlazara a algún sitio no perteneciente a Envato con la palabra *tuts* en la url? Cuando necesitas ser más específico, usa `^` y `$`, para hacer referencia al principio y fin de una cadena de texto, respectivamente.

[Ver Demo](#)

### Compatibilidad

- IE7+
- Firefox
- Chrome

- Safari
- Opera

### 13. X[href^="http"]

```
1 a[href^="http"] {  
2   background: url(path/to/external  
3   /icon.png) no-repeat;  
4   padding-left: 10px;  
5 }
```

¿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 anchor 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://.

Ahora, ¿si en vez de eso quisiéramos aplicar estilo a todas las etiquetas anchor que enlacen a, por decir, una foto? En

esos casos, busquemos el *final* de la cadena de texto.

[Ver Demo](#)

### Compatibilidad

- IE7+
- Firefox
- Chrome
- Safari
- Opera

## 14. X[href\$=".jpg"]

```
1 a[href$=".jpg"] {  
2   color: red;  
3 }
```

De nuevo, usamos un símbolo de expresiones regulares, \$, para referirnos al final de una cadena de texto. En éste caso, estamos buscando todos los anchor que enlacen a una imagen – o por lo menos a una url que termine con .jpg. Ten en mente que esto seguramente no funcionará para gifs ni pngs.

[Ver Demo](#)

### Compatibilidad

- IE7+

- Firefox
- Chrome
- Safari
- Opera

## 15. X[data-\*= "foo"]

```
1 a[data-filetype="image"] {
2   color: red;
3 }
```

Regresemos al número ocho; ¿como compensamos para los diferentes tipos de imagen: png, jpeg, jpg, gif?

Bueno, podríamos crear múltiples selectores, tal como:

```
1 a[href$=".jpg"],
2 a[href$=".jpeg"],
3 a[href$=".png"],
4 a[href$=".gif"] {
5   color: red;
6 }
```

Pero, eso es un dolor en el trasero y es ineficiente. Otra posible solución es usar atributos personalizados. ¿Y si agregáramos nuestro propio atributo data-filetype para cada anchor que enlaza a una imagen?

```
1 <a href="path/to/image.jpg" data-
```

```
filetype="image"> Image Link </a>
```

Entonces, *dicho esto*, dicho esto, podemos usar un selector de atributos estándar para afectar sólo a esos anchors.

```
1 a[data-filetype="image"] {  
2   color: red;  
3 }
```

[Ver Demo](#)

## Compatibilidad

- IE7+
- Firefox
- Chrome
- Safari
- Opera

## 16. X[foo~="bar"]

```
1 a[data-info~="external"] {  
2   color: red;  
3 }  
4 a[data-info~="image"] {  
5   border: 1px solid black;  
6 }  
7
```

Aquí hay uno especial que impresionará a tus amigos. No mucha gente sabe de éste truco. El símbolo tilde (~) nos permite afectar un atributo que tenga una lista de valores separados por espacios.

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 tomar nota. En éste caso, haremos nota de enlaces externos y enlaces a imágenes – justo para el ejemplo.

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

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

```
1 /* Target data-info attr that contains the  
2 value "external" */  
3 a[data-info~="external"] {  
4   color: red;  
5 }  
6 /* And which contain the value "image" */  
7 a[data-info~="image"] {  
8   border: 1px solid black;  
9 }
```



Bastante ingenioso, ¿eh?

[Ver Demo](#)

### Compatibilidad

- IE7+
- Firefox
- Chrome
- Safari
- Opera

## 17. X:checked

```
1 input[type=radio]:checked {  
2   border: 1px solid black;  
3 }
```

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

[Ver Demo](#)

### Compatibilidad

- IE9+
- Firefox
- Chrome

- Safari
- Opera

## 18. X:after

Las pseudo clases 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.

Muchos conocieron primero éstas clases cuando encontraron el hack de clear-fix.

```
01 .clearfix:after {
02     content: "";
03     display: block;
04     clear: both;
05     visibility: hidden;
06     font-size: 0;
07     height: 0;
08 }
09 .clearfix {
10     *display: inline-block;
11     _height: 1%;
12 }
13 }
```

Éste *hack* usa la pseudo clase `:after` para agregar un espacio después del elemento, y luego borrarlo. Es un excelente truco para tener en tu bolsa de herramientas, particularmente en los casos cuando el método `overflow: hidden;` no es posible.

Para otro uso creativo de esto, [dirígete a mi consejo rápido para crear sombras](#).

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

## Compatibilidad

- IE8+
- Firefox
- Chrome
- Safari
- Opera

## 19. X: hover

```
1 div: hover {  
2 background: #e3e3e3;  
3 }
```

O vamos. Tú te sabes este. El término oficial para este es pseudo clase de acción de usuario. Suena confuso, pero en realidad no lo es. ¿Quieres aplicar un estilo específico cuando un usuario pasa sobre un elemento? ¡Esto hará el trabajo!

Ten en cuenta que versiones más viejas de Internet Explorer no responden cuando la pseudo clase `:hover` es aplicada a cualquier otra cosa que no sea una etiqueta `anchor`.

Usarás más seguido éste selector cuando apliques, por ejemplo, un `border-bottom` a etiquetas `anchor`, cuando se pase por encima.

```
1 a:hover {  
2   border-bottom: 1px solid black;  
3 }
```

**Pro-tip** - `border-bottom: 1px solid black;` luce mejor que `text-decoration: underline;`.

## Compatibilidad

- IE6+ (In IE6, `:hover` must be applied to an anchor element)
- Firefox
- Chrome
- Safari
- Opera

## 20. X:not(selector)

```
1 div:not(#container) {  
2   color: blue;  
3 }
```

La pseudo clase negación es particularmente útil.

Digamos que quiero seleccionar todos los divs, excepto por 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 por las etiquetas de párrafo, podríamos hacer:

[Ver Demo](#)

### Compatibilidad

- IE9+
- Firefox
- Chrome
- Safari
- Opera

## 21. X::pseudoElement

```
1 p::first-line {  
2   font-weight: bold;  
3 }
```

```
    font-size: 1.2em;
4  }
```

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 nivel bloque para que funcione.

Un pseudo-elemento está compuesto de dos “dos puntos”  
::

### **Afecta La Primer Letra De Un Párrafo**

```
1 p::first-letter {
2   float: left;
3   font-size: 2em;
4   font-weight: bold;
5   font-family: cursive;
6   padding-right: 2px;
7 }
```

Este fragmento de código es una abstracción que encontrará 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**

```
1 p::first-line {  
2   font-weight: bold;  
3   font-size: 1.2em;  
4 }
```

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

“Para compatibilidad con hojas de estilo existentes, los agentes de usuario 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.” - [Fuente](#)

[Ver Demo](#)

## Compatibilidad

- IE6+
- Firefox
- Chrome
- Safari
- Opera

## 22. X:nth-child(n)

```
1 li:nth-child(3) {
```

```
2 color: red;
3 }
```

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

Por favor observa que `nth-child` acepta un número entero como parámetro, sin embargo, éste no es basado en cero. Si quieres afectar el segundo elemento de una lista, usa `li:nth-child(2)`.

Podemos incluso usar esto para seleccionar un conjunto variable de hijos. Por ejemplo, podríamos usar `li:nth-child(4n)` para seleccionar todos los cuartos elementos de una lista.

[Ver Demo](#)

## Compatibilidad

- IE9+
- Firefox 3.5+
- Chrome
- Safari

## 23. `X:nth-last-child(n)`

```
1 li:nth-last-child(2) {
2   color: red;
3 }
```



```
}
```

¿Y si tuvieras una inmensa lista de elementos en un `ul`, y solo necesitaras acceder, digamos, del tercer al último elemento? En vez de usar `li:nth-child(397)`, en su lugar podrías usar la pseudo clase `nth-last-child`.

Ésta técnica funciona casi igual que el número dieciséis, sin embargo, la diferencia es que comienza al final del grupo y de ahí hacia atrás.

[Ver Demo](#)

## Compatibilidad

- IE9+
- Firefox 3.5+
- Chrome
- Safari
- Opera

## 24. X:nth-of-type(n)

```
1 ul:nth-of-type(3) {  
2   border: 1px solid black;  
3 }
```

Habrás veces cuando, en vez de seleccionar un hijo, necesites seleccionar de acuerdo al tipo (`type`) de elemento.

Imagina un código que contiene cinco listas sin orden. Si quisieras estilizar sólo la tercera `ul`, y no tuvieras un `id` único al cual engancharla, podrías usar la pseudo clase `nth-of-type(n)` En el fragmento de arriba, sólo el tercer `ul` tendrá un borde a su alrededor.

[Ver Demo](#)

## Compatibilidad

- IE9+
- Firefox 3.5+
- Chrome
- Safari

## 25. X:nth-last-of-type(n)

```
1 ul:nth-last-of-type(3) {  
2   border: 1px solid black;  
3 }
```

Y sí, para permanecer consistente, también podemos usar `nth-last-of-type` para comenzar al final de la lista del selector y partir hacia atrás para afectar al elemento deseado.

## Compatibilidad

- IE9+
- Firefox 3.5+

- Chrome
- Safari
- Opera

## 26. X:first-child

```
1 ul li:first-child {  
2   border-top: none;  
3 }
```

Esta pseudo clase estructural nos permite afectar sólo al primer hijo del padre del elemento. Usarás esto frecuentemente para remover bordes de los primeros y últimos elementos de una lista.

Por ejemplo, digamos que tienes una lista de filas, y cada una tiene un borde superior `border-top` y un borde inferior `border-bottom`. Bueno, con ése arreglo, el primer y último elemento en ese conjunto lucirán un poco raro.

Muchos diseñadores aplican clases de `first` y `last` para compensar esto. En lugar de eso, puedes usar éstas pseudo clases.

[Ver Demo](#)

### Compatibilidad

- IE7+
- Firefox
- Chrome

- Safari
- Opera

## 27. X:last-child

```
1 ul > li:last-child {  
2   color: green;  
3 }
```

Al contrario de `first-child`, `last-child` afectará el último elemento del padre el elemento.

### Ejemplo

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

### Código

```
1 <ul>  
2   <li> List Item </li>  
3   <li> List Item </li>  
4   <li> List Item </li>  
5 </ul>
```


Nada especial aquí, sólo una simple lista.

### CSS

```
01 ul {
02   width: 200px;
03   background: #292929;
04   color: white;
05   list-style: none;
06   padding-left: 0;
07 }
08 li {
09   padding: 10px;
10   border-bottom: 1px solid black;
11   border-top: 1px solid #3c3c3c;
12 }
13 }
```

Éste estilo fijará un fondo, removerá el padding por defecto del navegador en la `ul`, y aplicará bordes para cada `li` para proveer un poco de profundidad.





Para agregar profundidad a tus listas, aplica un borde inferior `border-bottom` a cada `li` que es 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 sin orden – lo cual luce un poco extraño. Usemos las pseudo clases `:first-child` y `:last-child` para arreglar esto.

```
1 li:first-child {  
2     border-top: none;  
3 }  
4 li:last-child {  
5     border-bottom: none;  
6 }  
7 }
```



List Item

List Item

# List Item

¡Ahí vamos; eso lo arregla!

[Ver Demo](#)

## Compatibilidad

- IE9+
- Firefox
- Chrome
- Safari
- Opera

*Sip – IE8 soporta :first-child, pero :last-child.  
Imagínate.*

## 28. X:only-child

```
1 div p:only-child {  
2   color: red;  
3 }
```

A decir verdad, probablemente no te veas usando la pseudo clase `only-child` muy seguido. No obstante, está disponible, en caso de que la necesites.

Te permite afectar elementos que sean los *únicos* hijos de su padre. Por ejemplo, haciendo referencia al fragmento de

código de arriba, sólo el párrafo que es el único hijo del `div` será coloreado rojo.

Asumamos el siguiente código.

```
1 <div><p> Mi Párrafo. </p></div>
2 <div>
3   <p> Dos Párrafos en Total. </p>
4   <p> Dos Párrafos en Total. </p>
5 </div>
6
```

En este caso, el párrafo del segundo `div` no será afectado; sólo el primer `div`. Tan pronto como apliques más de un hijo a un elemento, la pseudo clase `only-child` deja de tener efecto.

[Ver Demo](#)

## Compatibilidad

- IE9+
- Firefox
- Chrome
- Safari
- Opera

## 29. X:only-of-type

```
1 li:only-of-type {
```



```
2 font-weight: bold;
3 }
```

Esta pseudo clase estructural puede ser usada en maneras inteligentes. Afectará elementos que no tienen hermanos dentro de su contenedor padre. Como ejemplo, afectemos a todos los `ul`s, que tengan sólo un elemento de lista.

Primero, pregúntate cómo lograrías esta tarea. Podrías usar `ul li`, pero, esto afectaría a *todos* los elementos de lista. La única solución es usar `only-of-type`.

```
1 ul > li:only-of-type {
2   font-weight: bold;
3 }
```

[Ver Demo](#)

## Compatibilidad

- IE9+
- Firefox 3.5+
- Chrome
- Safari
- Opera

## 30. X:first-of-type

La pseudo clase `first-of-type` te permite seleccionar los primeros hermanos de su tipo.

## Una Prueba

Para entender mejor esto, hagamos una prueba. Copia el siguiente código en tu editor de texto:

```
01 <div>
02   <p> My paragraph here. </p>
03   <ul>
04       <li> List Item 1 </li>
05       <li> List Item 2 </li>
06   </ul>
07   <ul>
08       <li> List Item 3 </li>
09       <li> List Item 4 </li>
10   </ul>
11 </div>
12
```

Ahora, sin leer más adelante, trata de resolver como afectar sólo a *"List Item 2"*. Cuando lo soluciones (o te rindas), continúa leyendo.

## Solución 1

Hay una variedad de formas de resolver ésta prueba. Revisaremos un puñado de ellas. Comencemos usando `first-of-type`.

```
1 ul:first-of-type > li:nth-child(2) {
```

El código en esencia dice, “encuentra la primer lista sin orden en la página, después encuentra sólo el hijo inmediato, que son elementos de lista. Después, fíltralo a sólo el segundo elemento de la lista en ese conjunto.

## **Solución 2**

Otra opción es usar el selector adyacente.

```
1 p + ul li:last-child {  
2 font-weight: bold;  
3 }
```

En éste escenario, encontramos el `ul` que inmediatamente procede a la etiqueta `p` y luego encontramos el último hijo del elemento.

## **Solución 3**

Podemos ser tan odiosos o traviesos como queramos con éstos selectores.

```
1 ul:first-of-type li:nth-last-child(1) {  
2 font-weight: bold;  
3 }
```

Ésta vez, tomamos el primer `ul` en la página y después encontramos el primer elemento, ¡pero empezando desde el fondo! :)

[Ver Demo](#)

## Compatibilidad

- IE9+
- Firefox 3.5+
- Chrome
- Safari
- Opera

## Conclusión

Si estás compensando para navegadores más antiguos, como Internet Explorer 6, tienes que seguir teniendo cuidado cuando uses éstos nuevos selectores. Pero, por favor no dejes que eso te impida aprenderlos. Estarías haciendo un gran deservicio a tu persona. Asegúrate de [entrar aquí para una lista de navegadores compatibles](#). De manera alternativa, puedes usar [El excelente script IE9.js de Dean Edward](#) para soportar estos selectores en navegadores más antiguos.

En segundo lugar, cuando trabajes con librerías JavaScript, como la popular JQuery, siempre trata siempre de usar éstos selectores nativos de CSS3 en vez de los métodos y selectores personalizados de la librería, cuando sea posible. Esto [hará to código más rápido](#), ya que el motor de

selectores puede usar el analizador nativo del navegador, en vez del propio.

¡Gracias por leer, y espero que hayas tomado un truco o dos!

¡Sé el primero en conocer las nuevas traducciones—sigue [@tutsplus\\_es](https://twitter.com/tutsplus_es) en Twitter!