

Unidad 5

Selectores avanzados

*“Somos lo que elegimos ser,
siempre podemos hacer lo correcto”*

Peter Parker

Aunque hemos dicho que prácticamente cualquier página web puede hacerse con los cinco selectores básicos que ya conocemos, existen algunos que se llaman avanzados que nos van a permitir, bien simplificar algunas tareas de aplicación de estilos o bien permitirnos algunos efectos imposibles de conseguir de ninguna otra forma como es el caso de la utilidad proporcionada por alguna de las pseudo-classes. Los presentaremos en las siguientes páginas.

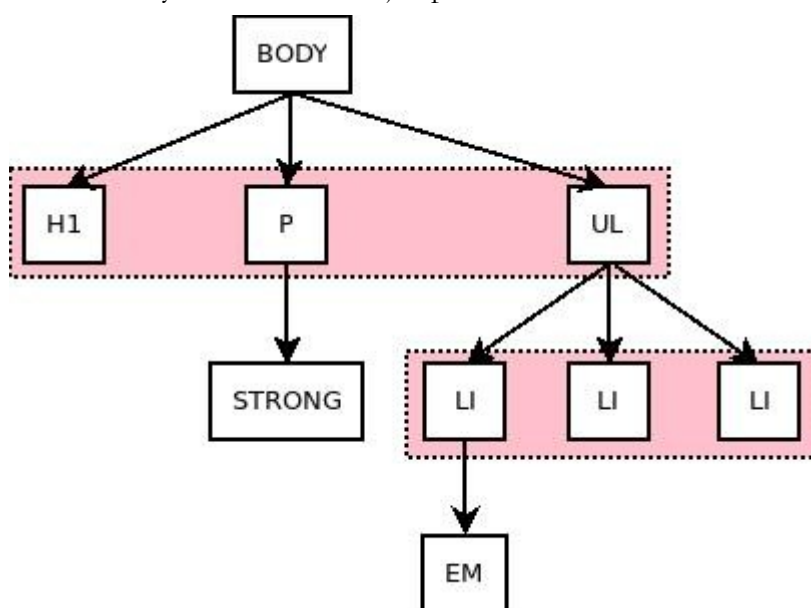
Selector de hijos

El selector de hijos es muy similar al selector descendiente pero es mucho más específico. Cuando uso un selector descendiente, por ejemplo `p.destacado` sólo me importa que el elemento perteneciente a la clase destacado se encuentre dentro del elemento `p`, sin importarme cuanto “profundo” se encuentre dentro del árbol jerárquico de etiquetas del documento. Si uso el selector de hijo es porque quiero que el elemento con la clase destacado sea hijo directo del elemento `p` y no un descendiente cualquiera. Y se simboliza así:

```
p > .destacado {color: purple; }
```

Selector de hermanos adyacentes (o adyacente, sin más)

El selector de hermanos es aquel que nos permite seleccionar elementos que comparten el mismo padre y están adyacentes uno al otro. En el siguiente grafico podemos ver destacados los hermanos de un HTML sencillo ya visto en otros ejemplos:



Para aplicar un determinado estilo al primer párrafo después de un titular de nivel 1, usaríamos la siguiente sintaxis:

```
h1 + p {text-indent: 2.5em; }
```

El selector de hermanos es una forma muy útil de crear estilos para ciertas combinaciones de elementos muy utilizados: el primer párrafo de un capítulo, el primer párrafo posterior a una tabla, etc. También es útil para seleccionar todos los elementos menos el primero en una secuencia determinada. Por ejemplo:

```
p + p { text-indent: 1.5em; }
```

Selector de atributos

Los selectores de atributos nos permiten aplicar estilos a determinados elementos en función de los atributos que acompañen a la etiqueta HTML o de los valores que tengan dichos atributos. Si nos paramos a pensar, en realidad los selectores de clase o de ID son una especie de selectores de atributo porque aplican estilos en función del valor de un atributo (class o ID, respectivamente), así que no vamos a ver nada demasiado nuevo en este apartado.

Existen cuatro tipos diferentes de selectores de atributo. Los primeros seleccionan los elementos en función de un determinado atributo independientemente de su valor:

```
td[rowspan] {background-color: black; color: white; }
```

Los segundos seleccionan un elemento en función de un atributo con un valor determinado:

```
a[href="http://www.google.es/"] {color: red; text-weight: bold; }
```

La tercera clase de selectores de atributos realiza su selección en función del atributo y de la existencia de un determinado valor pero que no tiene porqué ser unico, sino simplemente aparecer en una lista separada por espacios:

```
div[class~="alerta"] {color: red;}
```

Podemos correr el riesgo de pensar que simplemente es una forma de hacer más complicado lo que ya sabemos hacer con otros selectores más simples. Por ejemplo, estas definiciones son casi idénticas:

```
div.alerta {color: red;}  
div[class="alerta"] {color: red;}  
div[class~="alerta"] {color: red;}
```

Pero también podemos pensar en otras posibilidades más interesantes:

```
img[alt~="ilustración"] {border: 2px solid;}
table[summary~="2010"] {background-color: green;}
*[title~="importante"] {color: red; cursor: crosshair; }
```

El cuarto tipo de selector de atributo apenas tiene utilidad. Permite seleccionar elementos con cierto atributo y que incluyen un determinado valor como el primer elemento de una lista separada por guiones.

```
*[lang|="es"] { color: blue; }
```

El anterior ejemplo seleccionaría a todos los elementos con un atributo lang que empiece por es- (por ejemplo es-ES, es-MX, es-AR, etc.) De hecho esta es su principal (y casi única) utilidad: la selección de elementos según los códigos estándares de idiomas.

Pseudo-clases y pseudo-elementos

Existen ciertos estados o características de nuestro documento HTML a los que es útil y apropiado aplicarle un estilo diferenciado: un enlace ya visitado, la primera letra de un párrafo, etc. Esto se consigue mediante las pseudoclasas. Para identificar que hablamos de una pseudo-clase en una regla CSS usamos los dos puntos como signo diferenciador y estas pueden aplicarse sobre selectores de tipo, de clase, etc. Veremos ejemplos enseguida.

Y una nota: en algunos textos se hace una separación entre pseudo-clases y pseudo-elementos mientras que en otras se engloban todas bajo la misma categoría. Aquí haremos la diferenciación clásica porque, si bien no existen grandes diferencias, si que es necesario hacer la distinción a la hora de calcular la especificidad de las reglas CSS.

IMPORTANTE: Además, para reforzar la diferencia entre ambas, en CSS3 se ha cambiado la forma de referirnos a los pseudo-elementos. Ahora se usan dos dobles puntos :: y no uno sólo : como se hace con las pseudo-clases

Pseudo-clases

Las pseudo-clases son aquellas que se aplican fundamentalmente sobre los selectores de enlace (aunque alguna de ellas puede aplicarse también sobre otros elementos). Se trata de `:link`, `:visited`, `:hover`, `:active`, `:focus`, `:first-child` y `:lang`.

La pseudo-clase `:link` hace referencia a un enlace que aún no ha sido visitado, mientras que `:visited` hace referencia a lo opuesto, es decir, los enlaces que ya han sido visitados por el usuario. Dos ejemplos:

```
a:link {color: green; background-color: yellow;}  
a:visited {color: yellow; background-color: green; }
```

La pseudo-clase `:hover` nos dicta los estilos a aplicar al pasar el ratón por encima del elemento al que el selector hace referencia. Se suele usar fundamentalmente con enlaces pero es aplicable a cualquier otro elemento:

```
a:hover, p.destacado:hover{font-weight: bold; }  
em:hover {color: blue;}
```

`:active` es una pseudo-clase cuyos estilos se aplican al hacer click sobre un elemento o activarlo mediante la tecla Intro cuando tiene el “foco”:

```
h1:active {color: orange; background-color: black;}
```

Como imaginamos, muchas de estas propiedades pueden activarse simultáneamente: un enlace puede haber sido visitado y, podemos pasar el ratón por encima de él, etc. EL orden en que se colocan en la hoja de estilos es por ello fundamental para aplicarlos correctamente cuando queremos aplicar reglas distintivas a todos ellos:

```
a:link {color: blue;}  
a:visited {color: green;}  
a:hover {color: orange;}  
a:active {color: red;}
```

Podemos aplicar también reglas a dos pseudo-clases simultáneamente. Por ejemplo, la siguiente regla se aplicaría sobre un enlace visitado sobre el que hacemos click:

```
a:visited:active {color: black; background-color: orange;}
```

Por último, :focus es una regla que se aplica cuando el elemento tiene el “foco”, es decir, cuando se activaría si pulsamos la tecla Intro. Para poder usarla debemos de definir que elementos tienen el foco en nuestra página y el orden en el que lo reciben (cada vez que pulsamos secuencialmente el tabulador). Esto se hace mediante el atributo tabindex que podemos usar en las etiquetas a (enlaces) de nuestra web posibilitando así la navegación sin necesidad de ratón. Por ejemplo:

```
<p>Un enlace a <a href="http://www.google.es/" alt="Google" tabindex="1">Google</a>, otro a <a href="http://www.bing.es" alt="Bing" tabindex="2">Bing</a> y un tercero a <a href="http://www.yahoo.es" alt="Yahoo" tabindex="3">Yahoo</a>.</p>
```

Y ahora ya si podemos ver los efectos de la pseudo-clase :focus

```
a:focus {background-color: black; color: white; font-size: 1.8em;}
```

La pseudo-clase :first-child nos permite seleccionar un determinado elemento siempre y cuando sea el primer hijo de su padre. Por ejemplo, veamos las reglas siguiente:

```
p:first-child {color: blue; }  
p:first-child em {background: yellow; }
```

La primera selecciona y aplica estilo a un párrafo siempre y cuando sea el primer hijo de su padre, mientras que la segunda hace lo mismo sobre las cursivas de ese mismo tipo de párrafos.

Existe una última pseudo-clase de aplicación muy limitada llamada :lang que nos permite aplicar estilos diferenciados en función del idioma en que esté escrito el elemento. Por ejemplo así:

```
p:lang(es) {color: red; }
```

Aunque tienen un uso muy similar al visto en uno de los selectores de atributo, existe una ligera diferencia. El selector de atributo es aplicable sólo a los contenidos que usen el atributo lang, mientras que esta pseudo-clase sería aplicable también en otras circunstancias, como por ejemplo si el propio navegador fuese capaz de analizar el contenido y detectar elementos escritos en diferentes idiomas (cosa que, por el momento, no hacen demasiado bien...).

Pseudo-elementos

Los dos primeros pseudo-elementos que veremos son aplicables a la decoración de párrafos de texto. Se llaman `::first-letter` y `::first-line` y sus nombres son autoexplicativos

```
p.decorado::first-letter {margin-right: 4px; font-size: 4em; color:  
red; font-weight: bolder;}  
p.decorado::first-line {font-variant: small-caps; font-weight: bold;}
```

Existen otros dos pseudo-elementos llamados `::before` y `::after` que usados junto con la propiedad `content` nos permiten insertar contenido (texto, imágenes, etc.) antes o después de un elemento.

```
p#insert1::after{ content: " (TM – Marca Registrada) "};  
p.insert2::before{ content: url(icon_new.gif);}
```

En los anteriores ejemplos, el primero inserta el texto que aparece entre comillas al final de cada párrafo identificado con el selector de ID `insert1`, mientras que el segundo inserta el gif animado `icon_new.gif` delante de cada párrafo identificado como perteneciente a la clase `insert2`.

Los nuevos selectores de CSS3

En CSS3 se añade un nuevo pseudo-elemento llamado `::selection` que permite aplicar estilos diferenciados al texto que el usuario ha seleccionado mediante el ratón o el teclado.

Pero el principal cambio en los selectores de CSS3 se encuentra en las pseudo-classes donde se han añadido 14 selectores nuevos

El selector `:root` representa al elemento raíz del documento HTML que siempre ha de ser la propia etiqueta `html`. No existe ninguna diferencia apreciable entre usar este o la etiqueta `body`, salvo que `:root` tendría una mayor especificidad.

`:empty` es un selector que demarca a todos los elementos vacíos de contenido y sin hijos. Por ejemplo, tengamos el siguiente HTML:

```
<p>A paragraph.</p>
<p></p>
<p>Another paragraph.</p>
```

El párrafo central, vacío y sin hijos, sería el único que respondería a este selector:

```
p:empty{ width:200px; height: 40px; background: red; }
```

El selector `:target` sirve para seleccionar el elemento activo, o sea, aquél enlace interno realizado mediante la etiqueta `a` y que acabas de visitar. Veamos este código:

```
<p><a href="#enlace1">Ir al punto 1</a></p>
<p><a href="#enlace2">Ir al punto 2</a></p>
<p><a name="enlace1"></a>La, la, la...</p>
<p><a name="enlace2"></a>Bra, bra, bra...</p>
```

Y usemos esta regla combinada con el selector `:before`

```
:target:before{
    content: "Estás aquí -> ";
}
```

:last-child viene a acompañar al selector :first-child que ya conocíamos y selecciona a un determinado elemento siempre y cuando sea el último hijo de un padre cualquiera. Por ejemplo, supongamos el siguiente código:

```
<body>
  <p>Primer párrafo del body</p>
  <h1>Titular</h1>
  <div>
    <p>Primer párrafo del div</p>
    <p>Segundo párrafo del div</p>
    <p>Tercer párrafo del div</p>
  </div>
</body>
```

El selector p:first-child seleccionaría a los párrafos marcados en azul (el primer hijo del body y el primer hijo del div), mientras que el selector p:last-child seleccionaría al párrafo marcado en rojo (el último hijo del div).

Siguiendo en esta línea, :only-child seleccionaría a un elemento que fuese hijo único de su padre. :nth-child(n) seleccionaría al elemento que está en la posición n dentro de la descendencia de su padre. El primer elemento sería el 1 y, por lo tanto, p:nth-child(1) sería equivalente a p:first-child. Por último, :nth-last-child(n) seleccionaría al elemento que está en la posición n pero empezando a contar por el último y no por el primero.

:nth-child(n) permite un uso avanzado configurando adecuadamente su argumento único. Podemos cambiar la letra n por las palabras claves odd o even y, en ese caso, seleccionaríamos a todos los elementos impares o pares, respectivamente. También podemos usar expresiones como 3n (los elementos múltiplos de 3) o usar tres reglas diferentes (3n+1, 3n+2 y 3n+3) para alternar entre tres estilos diferentes en una lista de elementos iguales.

Similares a estos, tenemos un conjunto de selectores que se comportan de igual forma pero con los elementos de un determinado tipo. Son :first-of-type, :last-of-type, only-of-type, :nth-of-type(n) y :nth-last-of-type(n).

Nuevos selectores para uso con formularios

Existen, además, tres pseudo-clases nuevas específicamente pensadas para su uso con formularios que son `:enabled`, `:disabled` y `:checked`. Como su nombre indica, aplican a los elementos de tipo `input` que cumplen con lo especificado en el nombre del selector.

El selector `:not()`

Existe una última pseudo-clase que trataremos aparte por su especial importancia. Se trata del selector `:not()` que sirve para aplicar estilos a aquellos elementos que no cumplen lo especificado ente paréntesis. Veámoslo con algunos ejemplos. La siguiente regla se aplicaría al contenido de cualquier `div` que no fuese de la clase “comun”

```
div:not(.comun){color: red; }
```

La siguiente regla aplicaría a todos los elementos `strong` que están dentro de un `div` pero que no están dentro de un `p`

```
div *:not(p) strong{color: red; }
```

En la siguiente, lo combinamos con un selector de atributo para seleccionar todos los `input` salvo aquellos de tipo `password`.

```
input:not([type="password"]){color: red; }
```

Podemos usar también una lista como argumento del `not`. Lo siguiente se aplicará a todos los titulares de nivel `h2` salvo a aquellos que pertenecen a las clases “política” y “economía”:

```
h2:not(.politica, .economia){color: red; }
```

Y, por supuesto, se puede combinar con cualquier otro selector que hemos visto antes: descendientes, etc.

Nuevos selectores de atributos

CSS3 añade, además, tres nuevos selectores de atributos que nos permiten un control más fino. El primer selector que veremos, que usa el símbolo distintivo ^ sirve para seleccionar a etiquetas con un atributo cuyo valor empieza por una cadena. En el ejemplo, a aquellas etiquetas a con un atributo href cuyo valor empiece por mailto:

```
a[href^="mailto:"]{ color: blue; }
```

El segundo selector de atributo, que usa el símbolo distintivo \$ selecciona a aquellas etiquetas con un atributo concreto cuyo valor termina en determinada cadena. Por ejemplo, las etiquetas a cuyo atributo href terminan en html:

```
a[href$=".html"] { color: yellow; }
```

Por último, se incluye un selector que usa el símbolo * que selecciona a aquellas etiquetas con un atributo cuyo valor contenga en cualquier posición la cadena especificada. En el siguiente ejemplo, seleccionaría a cualquier imagen cuyo título contenga la palabra importante.

```
img[title*="importante"] {border: 3px red dashed; }
```

Todos los selectores de atributo pueden combinarse entre si y, por supuesto, con otros selectores como siempre. Por ejemplo, el siguiente selector cuadraría con las imágenes cuyo título empiece por Estadísticas y contenga la palabra importante:

```
img[title^="Estadísticas"][title*="importante"] {border: 3px red dashed; }
```

Selector general de elementos hermanos

El último de los nuevos selectores de CSS3 amplía las posibilidades del selector de hermanos existente en la versión anterior. Recordemos

que el selector de hermanos (que usaba el símbolo +) sólo se “activaba” cuando los dos elementos de la regla eran hermanos e iban uno a continuación de otro de forma inmediata. El nuevo selector usa el símbolo ~ y se activa cuando ambos elementos son hermanos y el segundo va después del primero pero sin importar que sea de forma inmediata. Veamos un ejemplo sencillo:

```
<h1>Titular 1</h1>
<h2>Titular 2</h2>
<p>Párrafo 1</p>
<div>
    <h2>Titular 3</h2>
</div>
<h2>Titular 4</h2>
```

Una regla que use el selector de hermanos “clásico” (por ej. h1+h2 {color: red;}) sólo seleccionaría el Titular 2, mientras que usando el selector general h1 ~ h2 {color: red; } seleccionaría el Titular 2 y el Titular 4.

Especificidad

A nadie se le escapa a estas alturas que a medida que vayamos haciendo más complejas nuestras hojas de estilo, más difícil será saber en algunas ocasiones que regla prevalecerá sobre otra. Ya hemos dicho que, en igualdad de condiciones (por ejemplo, dos reglas que hacen referencia a estilos contradictorios para la etiqueta p), se le dará preferencia a la que está más cercana al contenido si las reglas se encuentran en diferentes sitios (las más prioritarias serían las consignadas a través del atributo style en el propio HTML, las siguientes las que escribimos directamente en la sección head del mismo y, por último, aquellas que van en un fichero externo). Ante dos reglas que estén en el mismo sitio (por ejemplo, ambas en dos líneas diferentes de un mismo archivo externo CSS) se le dará preferencia a la que esté escrita en último lugar.

Estas reglas nos ayudan a solucionar los casos más simples de conflicto pero, ¿que ocurre cuando el problema lo plantean dos reglas

totalmente diferentes que se deberían de aplicar al mismo elemento? Para eso surge lo que llamamos especificidad.

Imaginemos que nuestro CSS tiene las siguientes reglas:

```
ul ol li.alerta {color: red; }  
.alerta{color: blue;}  
li.alerta{color: yellow; background-color: pink;}  
ul .alerta{color: green;}
```

Y en nuestro HTML aparece algo como esto:

```
<ul>  
  <li>Uno  
    <ol>  
      <li>Dos</li>  
      <li class="alerta">TRES</li>  
    </ol>  
  </li>  
  <li>Cuatro</li>  
</ul>
```

¿Qué regla o reglas se aplicarían al elemento cuyo atributo class corresponde con alerta, correspondiendo este con las cuatro reglas que aparecen en el CSS? Puedes probar que cualquiera de ellas se corresponde con el elemento poniéndolas una a una. Si las pones todas juntas, independientemente del orden en que estén observarás que el texto aparece con color rojo y el fondo con color amarillo, es decir, se está aplicando la regla 1 y, parcialmente, la regla 3.

El hecho de que el fondo aparezca amarillo es bien sencillo: aparece así porque, no entra en conflicto con ninguna propiedad de otra regla. De hecho, podemos ver que si movemos esa propiedad a cualquiera de las otras reglas el fondo seguirá apareciendo amarillo. Esto es porque en realidad las estamos aplicando todas y sólo en el caso de que exista un conflicto entre propiedades en diferentes reglas tenemos que decidir cual es la que hay que usar. Esto se arbitra calculando la especificidad de la regla.

La forma más ampliamente aceptada de calcular la especificidad es mediante un conjunto de cuatro números separados por comas (por ejemplo 0,1,0,5). Los números a la izquierda tienen un rango mayor que a la derecha y, por tanto, definen a un selector más específico y que por tanto prevalece. Ante empate en un nivel, miramos en el siguiente. En caso de empate en todos los niveles prevalece la regla escrita en última posición.

EJEMPLOS: entre una regla con valor 0,0,1,0 y otra con valor 0,0,0,13 prevalece la primera y entre una regla con valor 0,1,3,1 y otra con valor 0,1,3,2 prevalecería la segunda.

¿Cómo se calculan estos números? El primero es fácil y se usa cuando el estilo se aplica en el propio HTML mediante el atributo style. Este número sólo puede valer 0 o 1 (puesto que no podemos duplicar atributos en una misma etiqueta) y tiene siempre la máxima prioridad. Los otros tres se calculan sumando 1 según los tipos de selectores que aparezcan en la regla:

- El segundo número se calcula sumando uno por cada selector de id que aparezca en la regla
- El tercer número se calcula sumando 1 por cada selector de clase, selector de atributo o pseudo-clase que aparezca en la regla
- El cuarto número, el de menor prevalencia, se calcula sumando 1 por cada selector de etiqueta o pseudo-elemento que aparezca en la regla.

Así, si calculamos la especificidad de las cuatro reglas anteriores:

```
ul ol li#alerta {color: red; }           /* 0,1,0,3 */
.alerta{color: blue;}                   /* 0,0,1,0 */
li.alerta{color: yellow; background-color: yellow;} /* 0,0,1,1 */
ul .alerta{color: green;}                /* 0,0,1,1 */
```

Lo que nos confirma que, efectivamente, el color del texto debería de salir rojo independientemente del orden en que se situen las reglas. Si eliminamos esa regla, el texto saldrá en verde, pero puesto que hay un

empate entre esta regla y la que dicta que debería de salir amarillo y prevalece la escrita en último lugar.

Los selectores de hijos y hermanos no aportan una mayor especificidad a las reglas.

La clausula !important

La clausula !important se puede aplicar a cualquier regla aumentando así su nivel de prioridad de forma que prevalezca sobre definiciones posteriores del mismo selector e, incluso, sobre las valoraciones de especificidad. Ante las siguientes reglas, un texto etiquetado como strong, dentro de un párrafo (selector descendiente) y con el id minegrilla aparecería con color rojo:

```
strong {color: red !important;}  
p strong#minegrilla {color: blue; }
```