IFC INFORMÁTICA Y COMUNICACIONES

IES Santiago Hernández

Zaragoza, España

DISEÑO DE INTERFACES WEB

Unidad 2: USOS DE ESTILOS. SELECTORES CSS

Departamento de Informática

Curso 2021/2022

AENOR Empresa Registrada UNE-EN ISO 9001

CERTIFIED IQNet MANAGEMENT SYSTEM

*

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

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 bien para pruebas rápidas, 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.

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

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.

#X

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

Nos permite seleccionar por **id**. Los **id** son únicos en el código HTML.

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.

.X

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

Éste es un selector de clase **class** La diferencia de los id, **class** permite seleccionar varios elementos. Usa classes cuando quieras que un estilo se aplique a un grupo de elementos.

X Y

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

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 link, sólo necesitas los links que están dentro de una lista li?. Aquí es cuando usarías específicamente un selector descendiente.

Si empleas selectores como X Y Z A B , lo estás haciendo mal. Siempre pregúntate si es absolutamente necesario aplicar todo ese *peso*.

X > Y

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

La diferencia entre **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 a los **ul** que sean hijos directos del **div** con **id container**.

X

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

¿Qué pasa si quieres seleccionar todos los elementos en una página, de acuerdo a su etiqueta (tag), en vez de un nombre de id o clase. Usa un selector de etiqueta.

X:visited and X:link

```
a:link { color: red; }
a:visted { 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, 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*.

X:hover

```
div:hover {
    background: #e3e3e3;
}
```

Aplica un estilo específico cuando un usuario pasa sobre un elemento.

X + Y

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

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.

X ~ Y

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

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**.

X[title]

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

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.

X[href="foo"]

```
a[href="http://net.tutsplus.com"] {
    color: #1f6053;
}
```

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.

Es un poco rígido. ¿Que tal si el enlace dirige ciertamente a **net.tutsplus**, 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.

X[href*="nettuts"]

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

El asterisco designa que el valor que 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*. Esta es una declaración abierta. Cuando necesites ser más específico, usa **^** y **$**, para hacer referencia al **principio** y **fin** de una cadena de texto.

X[href^="http"]

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

Afecta a todas las etiquetas anchor que tienen un href que comienza con http y desplegará un pequeño icono junto a los enlaces que son externos.

X[href$=".jpg"]

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

En éste caso, estamos buscando todos los anchor que enlacen a una imagen – o por lo menos a una url que termine con .jpg.

X[data-*="foo"]

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

¿como seleccionamos los diferentes tipos de imagen: png, jpeg,jpg, gif?

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

Otra posible solución es usar atributos personalizados. ¿Y si agregáramos nuestro propio atributo data-filetype para cada anchor que enlaza a una imagen?

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

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

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

X[foo~="bar"]

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

a[data-info~="image"] {
        border: 1px solid black;
}
```

El símbolo (**~**) nos permite afectar un atributo que tenga una lista de valores separados por espacios.

Podemos crear un atributo **data-info** que puede recibir una lista separada por espacios de lo que necesitemos.

**" Click Me, Fool **

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

a[data-info~="image"] {
        border: 1px solid black;
}
```

X:checked

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

Este 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).

X:after

La pseudo clases after genera contenido al final del elemento seleccionado.

```
.clearfix:after {
        content: “ ";
   }
```

X:before

Las pseudo clases before generac ontenido antes del elemento seleccionado.

```
.clearfix:before {
        content: “ ";
   }
```

X:not(selector)

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

Selecciona todos los divs, excepto por los que tienen un id de container.
Para seleccionar todos los elementos excepto por las etiquetas de párrafo, podríamos hacer:

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

X::pseudoElement

Primera línea de un Párrafo

```
p::first-line { /* primera línea */
        font-weight: bold;
        font-size: 1.2em;
}
```

Podemos usar pseudo elementos (designados por ::) para enfatizar fragmentos de un elemento, como la primera línea, o la primera letra.
Primera Letra de un Párrafo p::first-letter

X:nth-child(n)

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

Selecciona un elemento específico hijo.

nth-child acepta un número entero como parámetro (menos el cero)

Si quieres seleccionar 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.

```
li:nth-child(2n+1) { ... } /* selecciona todos los elementos impares de una lista */
li:nth-child(2n) { ... } /* selecciona todos los elementos pares de una lista */
```

X:nth-last-child(n)

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

idéntico al anterior pero el número indicado se empieza a contar desde el último hijo.

X:nth-of-type(n)

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

Habrá 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.

Selecciona el elemento indicado pero con la condición de que sea el enésimo elemento hermano de ese tipo.

X:nth-last-of-type(n)

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

idéntico al anterior pero el número indicado se empieza a contar desde el último hijo.

X:first-child y X:last-child

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

seleccionan los elementos indicados pero con la condición de que sean respectivamente los primeros o últimos hijos de su elemento padre.

Usarás esto frecuentemente para quitar 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 esta especificación, el primer y último elemento en ese conjunto variarán su aspecto.

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

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

X:only-child

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

Te permite modificar 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.
Veamos el siguiente código.

```
<div><p> Mi Párrafo. </p></div>

<div>
        <p> Dos Párrafos en Total. </p>
        <p> Dos Párrafos en Total. </p>
</div>
```

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.

X:only-of-type

```
li:only-of-type {
        font-weight: bold;
}
```

Afectará elementos que no tienen hermanos dentro de su contenedor padre. Como ejemplo, si queremos modificar todas las **uls**, que tengan sólo un elemento de lista.

Podríamos usar **ul li**, pero, esto afectaría a *todos* los elementos de lista.

La única solución es usar only-of-type.

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

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:

```
<div>
    <p> My paragraph here. </p>
        <ul>
            <li> List Item 1 </li>
            <li> List Item 2 </li>
        </ul>

        <ul>
            <li> List Item 3 </li>
            <li> List Item 4 </li>
        </ul>
</div>
```

Ahora, sin leer más adelante, trata de resolver como afectar sólo a *"List Item 2"*

X:first-of-type

Solución 1

Hay una variedad de formas de resolver ésta prueba. Comencemos usando first-of-type.

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

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.

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

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.

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

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

ENLACES DE INTERÉS

https://code.tutsplus.com/es/tutorials/the-30-css-selectors-you-must-memorize--net-16048
https://uniwebsidad.com/libros/css-avanzado/capitulo-3/selectores-de-css-3