LMSGI – UD02
SELECTORES

Selectores básicos:

Selector universal:

Se utiliza para seleccionar todos los elementos de la página.

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

Selector de tipo o etiqueta:

Selecciona todos los elementos de la página cuya etiqueta XHTML coincide con el valor del selector.

```
h1 {
color: red;
}
h2 {
color: blue;
}
p {
color: black;
}
```

Se pueden agrupar:

```
h1, h2, h3 {
color: #8A8E27;
font-weight: normal;
font-family: Arial, Helvetica, sans-serif;
}

h1 { font-size: 2em; }
h2 { font-size: 1.5em; }
h3 { font-size: 1.2em; }
```

Selector descendente:

Selecciona los elementos que se encuentran dentro de otros elementos. Un elemento es descendiente de otro cuando se encuentra entre las etiquetas de apertura y de cierre del otro elemento.

La sintaxis formal del selector descendente se muestra a continuación:

```
elemento1 elemento2 elemento3 ... elementoN
```

Los selectores descendentes siempre están formados por dos o más partes separadas entre sí por espacios en blanco. La última parte indica el elemento sobre el que se aplican los estilos y todas las partes anteriores indican el lugar en el que se tiene que encontrar ese elemento para que los estilos se apliquen realmente.

Por ejemplo:

```
p span { color: red; }
```

selecciona todos los elementos de la página que se encuentren dentro de un elemento <p>. Si el código XHTML de la página es el siguiente:

```
<p>
...
<span>texto1</span>
...
<a href="">...<span>texto2</span></a>
...
</p>
```

El selector `p span` selecciona tanto `texto1` como `texto2`. El motivo es que en el selector descendente, un elemento no tiene que ser "hijo directo" de otro. La única condición es que un elemento debe estar dentro de otro elemento, sin importar lo profundo que se encuentre.

Al resto de elementos de la página que no están dentro de un elemento <p>, no se les aplica la regla CSS anterior.

Otro ejemplo:

```
p span { color: red; }
h1 span { color: blue; }
```

muestra, además, de color azul todo el texto de los contenidos dentro de un <h1>.

Otro ejemplo:

```
/* El estilo se aplica a todos los elementos "p", "a", "span" y "em" */
p, a, span, em { text-decoration: underline; }

/* El estilo se aplica solo a los elementos "em" que se
encuentran dentro de "p a span" */
p a span em { text-decoration: underline; }
```

Otro ejemplo:

```
p a { color: red; }
<p><a href="#">Enlace</a></p> /* Rojo */
<p><span><a href="#">Enlace</a></span></p> /* Rojo */
```

pero:

```
p * a { color: red; }
<p><a href="#">Enlace</a></p> /* NO Rojo */
<p><span><a href="#">Enlace</a></span></p> /* Rojo */
```

Selector de clase:

```
<body>
<p class="destacado">Lorem ipsum dolor sit amet...</p>
<p>Nunc sed lacus et <a href="#" class="destacado">est adipiscing</a>
accumsan...</p>
<p>Class aptent taciti <em class="destacado">sociosqu ad</em>
litora...</p>
</body>
```

A continuación, se crea en el archivo CSS una nueva regla llamada `destacado` con todos los estilos que se van a aplicar al elemento. Para que el navegador no confunda este selector con los otros tipos de selectores, se prefija el valor del atributo class con un punto (`.`):

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

`.destacado` ➔ "cualquier elemento de la página cuyo atributo `class` sea igual a `destacado`".

Combinando el selector de tipo y el selector de clase, se obtiene un selector mucho más específico:

```
p.destacado { color: red }
```

`p.destacado` ➔ "aquellos elementos de tipo <p> que dispongan de un atributo `class` con valor `destacado`".

`.destacado` es equivalente a `*.destacado`.

No debe confundirse el selector de clase con los selectores anteriores:

```
/* Todos los elementos de tipo "p" con atributo class="aviso" */
p.aviso { ... }

/* Todos los elementos con atributo class="aviso" que estén dentro
de cualquier elemento de tipo "p" */
p .aviso { ... }

/* Todos los elementos "p" de la página y todos los elementos con
atributo class="aviso" de la página */
p, .aviso { ... }
```

Es posible aplicar los estilos de varias clases CSS sobre un mismo elemento. La sintaxis es similar, pero los diferentes valores del atributo `class` se separan con espacios en blanco:

```
.error { color: red; }
.destacado { font-size: 15px; }
.especial { font-weight: bold; }
<p class="especial destacado error">Párrafo de texto...</p>
```

Si un elemento dispone de un atributo `class` con más de un valor, es posible utilizar un selector más avanzado:

```
.error { color: red; }
.error.destacado { color: blue; }
.destacado { font-size: 15px; }
.especial { font-weight: bold; }
<p class="especial destacado error">Párrafo de texto...</p> /* Azul */
```

`.error.destacado` ➔ "aquellos elementos de la página que dispongan de un atributo `class` con al menos los valores `error` y `destacado`".

Selectores de ID:

Permite seleccionar un elemento de la página a través del valor de su atributo `id`.

Este tipo de selectores sólo seleccionan un elemento de la página porque el valor del atributo `id` no se puede repetir en dos elementos diferentes de una misma página.

El atributo `class` no es obligatorio que sea único, de forma que muchos elementos XHTML diferentes pueden compartir el mismo valor para su atributo `class`.

Se utiliza el símbolo de la almohadilla (#) en vez del punto (`.`) como prefijo del nombre de la regla CSS:

```
#destacado { color: red; }
<p>Primer párrafo</p>
<p id="destacado">Segundo párrafo</p>
<p>Tercer párrafo</p>
```

Recomendación general:

- ⌦ Utilizar el selector de ID cuando se quiere aplicar un estilo a un solo elemento específico de la página.
- ⌦ Utilizar el selector de clase cuando se quiere aplicar un estilo a varios elementos diferentes de la página XHTML.

Combinación con otros selectores:

```
p#aviso { color: blue; }
```

Un selector de este tipo sólo tiene sentido cuando el archivo CSS se aplica sobre muchas páginas XHTML diferentes. En este caso, algunas páginas pueden disponer de elementos con un atributo `id` igual a `aviso` y que no sean párrafos, por lo que la regla anterior no se aplica sobre esos elementos.

No debe confundirse el selector de ID con los selectores anteriores:

```
/* Todos los elementos de tipo "p" con atributo id="aviso" */
p#aviso { ... }

/* Todos los elementos con atributo id="aviso" que estén dentro
de cualquier elemento de tipo "p" */
p #aviso { ... }

/* Todos los elementos "p" de la página y todos los elementos con
atributo id="aviso" de la página */
p, #aviso { ... }
```

Selectores avanzados:

Selector de hijos:

Para seleccionar un elemento que es hijo directo de otro elemento. Se indica mediante el "signo de mayor que" (>):

```
p > span { color: blue; }
<p><span>Texto1</span></p> // Azul
<p><a href="#"><span>Texto2</span></a></p> // No azul
```

`p > span` ➔ "cualquier elemento que sea hijo directo de un elemento <p>".

Selector adyacente:

Utiliza el signo + y su sintaxis es: `elemento1 + elemento2 { ... }`

Selecciona todos los elementos de tipo `elemento2` que cumplan las dos siguientes condiciones:

- `elemento1` y `elemento2` deben ser hermanos, por lo que su elemento “padre” debe ser el mismo.
- `elemento2` debe aparecer inmediatamente después de `elemento1` en el código XHTML de la página.

```
h1 + h2 { color: red }
<body>
<h1>Titulo1</h1>
<h2>Subtítulo</h2>  // Rojo
...
<h2>Otro subtítulo</h2>  // No Rojo
...
</body>
```

Selector de atributos:

Permiten seleccionar elementos XHTML en función de sus atributos y/o valores de esos atributos.

Cuatro tipos:

- `[nombre_atributo]`, selecciona los elementos que tienen establecido el atributo llamado `nombre_atributo`, independientemente de su valor.
- `[nombre_atributo=valor]`, selecciona los elementos que tienen establecido un atributo llamado `nombre_atributo` con un valor igual a valor.

- `[nombre_atributo~=valor]`, selecciona los elementos que tienen establecido un atributo llamado `nombre_atributo` y al menos uno de los valores del atributo es valor.
- `[nombre_atributo|=valor]`, selecciona los elementos que tienen establecido un atributo llamado `nombre_atributo` y cuyo valor es una serie de palabras separadas con guiones, pero que comienza con valor.

```
/* Se muestran de color azul todos los enlaces que tengan
un atributo "class", independientemente de su valor */
a[class] { color: blue; }

/* Se muestran de color azul todos los enlaces que tengan
un atributo "class" con el valor "externo" */
a[class="externo"] { color: blue; }

/* Se muestran de color azul todos los enlaces que apunten
al sitio "http://www.ejemplo.com" */
a[href="http://www.ejemplo.com"] { color: blue; }

/* Se muestran de color azul todos los enlaces que tengan
un atributo "class" en el que al menos uno de sus valores
sea "externo" */
a[class~="externo"] { color: blue; }

/* Selecciona todos los elementos de la página cuyo atributo
"lang" sea igual a "en", es decir, todos los elementos en inglés */
*[lang=en] { ... }

/* Selecciona todos los elementos de la página cuyo atributo
"lang" empiece por "es", es decir, "es", "es-ES", "es-AR", etc. */
*[lang|="es"] { color : red }
```

Agrupación de reglas:

```
h1 { color: red; }
...
h1 { font-size: 2em; }
...
h1 { font-family: Verdana; }
```

```
h1 {
color: red;
font-size: 2em;
font-family: Verdana;
}
```

Herencia:

Cuando se establece el valor de alguna propiedad en un elemento, todos sus descendientes heredan inicialmente ese mismo valor.

La herencia de estilos no funciona en todas las propiedades CSS, por lo que se debe estudiar cada propiedad de forma individual.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"
/>
<title>Ejemplo de herencia de estilos</title>

<style type="text/css">
body { font-family: Arial; color: black; }
h1 { font-family: Verdana; }
p { color: red; }
</style>

</head>

<body>
<h1>Titular de la página</h1>
<p>Un párrafo de texto no muy largo.</p>
</body>

</html>
```

Colisiones de estilos:

```
p { color: red; }
p { color: blue; }
<p>...</p>
```

CSS tiene un mecanismo de resolución de colisiones muy complejo y que tiene en cuenta el tipo de hoja de estilo que se trate (de navegador, de usuario o de diseñador), la importancia de cada regla y lo específico que sea el selector.

Método genérico seguido por CSS para resolver las colisiones:

1. Determinar todas las declaraciones que se aplican al elemento para el medio CSS seleccionado.
2. Ordenar las declaraciones según su origen (CSS de navegador, de usuario o de diseñador) y su importancia (palabra clave `!important`).
3. Ordenar las declaraciones según lo específico que sea el selector. Cuanto más genérico es un selector, menos importancia tienen sus declaraciones.
4. Si después de aplicar las normas anteriores existen dos o más reglas con la misma prioridad, se aplica la que se indicó en último lugar.

Norma que se puede seguir ➔ "especificidad" del selector:

1. Cuanto más específico sea un selector, más importancia tiene su regla asociada.
2. A igual especificidad, se considera la última regla indicada.

Ejemplo anterior ➔ los dos selectores son idénticos, las dos reglas tienen la misma prioridad y prevalece la que se indicó en último lugar, por lo que el párrafo se muestra de color azul.

```
p { color: red; }
p#especial { color: green; }
* { color: blue; }
<p id="especial">...</p>
```

Ejemplo actual ➔ Como el selector `p#especial` es el más específico, su declaración es la que se tiene en cuenta y por tanto el párrafo se muestra de color verde.