2023/24

U.D.3. USO DE ESTILOS.

3.2 Selectores. Clases e identificadores.

ÍNDICE

1. Definición de selectores

Como se vio en el capítulo anterior, **CSS o lenguaje de hojas de estilo** es el lenguaje usado para definir el aspecto y la presentación de las páginas web, es decir, a través de él se consigue ofrecer una mejor experiencia de navegación al usuario. Las hojas de estilo se aplican sobre un documento estructurado escrito en lenguaje HTML.

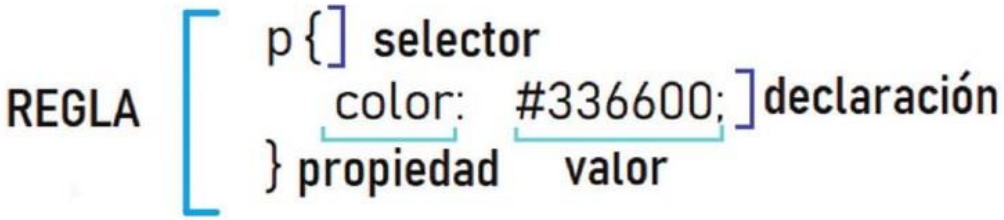
Para poder hacer el diseño de la página existen los mecanismos denominados **reglas, selectores y declaraciones**. Cada regla o conjunto de reglas consiste en uno o más selectores y un bloque de declaración o bloque de estilo. En el interior de estos bloques se definen elementos de aplicación, sobre los cuales se da valor a las propiedades deseadas.

Los estilos se aplican a los elementos del documento que cumplan con el selector que les precede. Cada bloque de estilos se define entre llaves, y está formado por una o varias declaraciones de estilo con la sintaxis.

Para crear diseños web profesionales es imprescindible conocer y dominar los selectores de CSS. Cada regla de CSS está formada por una parte llamada **selector** y otra parte llamada **declaración**.

La **declaración** indica "**lo que hay que hacer**", y el **selector** indica "**sobre qué elemento** hay que hacerlo". Por lo tanto, los selectores son imprescindibles para aplicar de forma correcta los estilos CSS en una página.

A un mismo elemento HTML se le pueden asignar infinitas reglas CSS, y cada regla CSS puede aplicarse a un número infinito de elementos. En otras palabras, una misma regla puede aplicarse sobre varios selectores y un mismo selector se puede utilizar en varias reglas.



En este capítulo veremos los selectores más importantes. La lista completa se puede ver en: https://www.w3schools.com/cssref/css_selectors.php

2. Selectores básicos

Existen múltiples tipos de selectores, en función del rango de acción sobre el que modifican las propiedades de los elementos que se definen en su declaración. Los tipos que se verán a continuación son selector universal, selector etiqueta, y selector descendente.

2.1 Selector universal (*)

Este selector permite **seleccionar todos los elementos** de una misma página, es decir, se da el mismo formato a todos los elementos recogidos en la página que esté utilizando esa hoja de estilo CSS. Su **sintaxis** es la siguiente:

```
* {
      propiedad: valor;
}
```

Un **ejemplo** del selector universal sería el siguiente:

```
* {
      font-size: 3em;
      color: red;
      text-align: left;
}
```

2.2 Selector etiqueta

El selector de tipo etiqueta permite seleccionar todos los elementos que contengan la etiqueta indicada en dicho selector. De esta forma, **todos aquellos elementos de un mismo tipo presentarán un mismo formato**, manteniendo una apariencia uniforme en todo el diseño del sitio web. Su **sintaxis** es la siguiente:

```
etiqueta HTML {
      propiedad: valor;
}
```

Un **ejemplo** código que incluye selectores etiqueta sería el siguiente:

```
h1 {
      font-size: 3em;
      color: red;
      text-align: left;
}

h2 {
      font-size: 2em;
      color: blue;
      text-align: center;
}
```

En este ejemplo se define un estilo diferenciado para los bloques determinados a través de las etiquetas de encabezado ***<h1>*** y ***<h2>***. Por ejemplo, los elementos de tipo h1 estarán alineados a la izquierda y serán de color rojo, mientras que a los de tipo h2 se les aplica una alineación central y son de color azul.

La creación de los selectores puede realizarse de varias **formas**:

1) **Selectores separados**, donde cada uno proporciona estilo de forma individual a un tipo de elemento.

2) **Selectores encadenados**, los cuales definen el mismo estilo para todos los elementos detallados en el selector.

3) **Selectores anidados o encadenados con particularidades**, similares al caso anterior, pero añadiendo características individuales para algunos elementos.

En los siguientes cuadros se recogen los tres tipos de presentación de selectores de etiquetas, con un ejemplo de uso de cada uno de ellos.

Selectores separados		

p{
…
}
	Selecciona todos los párrafos de la página	h1 {
		color: red;
		}

h1{
…
}
	Selecciona todos los títulos de tipo h1 de la página	h2 {
		color: blue;
		}

h2{
…
}
	Selecciona todos los títulos de tipo h2 de la página	p{
		color: black;
		}

Selectores encadenados

Se aplica el mismo estilo a todos los elementos representados por las etiquetas encadenadas. A continuación, se indica el formato que se le va a aplicar de la misma forma que en los selectores separados.

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

Selectores encadenados incorporando particularidades para cada selector

Al igual que en el caso anterior se aplica el mismo estilo a todos los elementos representados por las etiquetas encadenadas, ya continuación, se indica las particularidades de cada selector.

```
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;
}
```

2.3 Selector descendiente

Un elemento es **descendiente** de otro cuando se encuentra **entre las etiquetas de apertura y de cierre del elemento "padre"**. Un selector de tipo descendente seleccionará los elementos del tipo especificado que se encuentran dentro de otro elemento, también denotado en la sintaxis del selector. Es decir, las propiedades recogidas en la declaración se aplicarán sólo a los elementos de la etiqueta citada que estén dentro de otra etiqueta concreta.

La **sintaxis** del selector descendente se muestra a continuación: ***selector1 selector2 ... selectorN.***

Siendo el selector N el elemento sobre el que se aplica el estilo, el resto de selectores indican el lugar donde se encuentra el elemento indicado por el selector último.

- **Sintaxis** del selector descendiente:

```
selector1 selector2 ... selectorNhtml {
                        propiedad: valor;
}
```

La **combinación del selector descendiente con el selector universal** permite restringir el alcance del primero. En lugar de aplicar el estilo a todos los elementos que desciendan de la ruta marcada, sólo se le aplicará a aquellos que se encuentren dentro de cualquier otro elemento y no directamente del que aparece marcado con el símbolo del selector universal (*). Obsérvese el siguiente **ejemplo**:

- **Selector descendiente**: se aplica a los dos enlaces, puesto que ambos se localizan bajo un selector de tipo ***<p>*.**

```
p a {
   color: blue;
}
```

```
<p>
   <a href="#">Enlace uno</a>
</p>
<p>
   <span>
      <a href="#">Enlace dos</a>
   </span>
</p>
```

- **Selector descendente combinado con selector universal**: solo se muestra de color azul el segundo enlace, puesto que el selector universal indica que se aplique a todos aquellos elementos `<a>` que se encuentran bajo cualquier elemento que a su vez se encuentre dentro de un elemento de tipo `<p>`.

```
p * a {
    color: blue;
}
```

```
<p>
   <a href="#">Enlace uno</a>
</p>
<p>
   <span>
      <a href="#">Enlace dos</a>
   </span>
</p>
```

Es importante **no confundir el selector descendiente de este aparato con la combinación de selectores vista en el apartado anterior**. En el segundo caso, el estilo se aplica a todos los selectores mientras que con el selector descendente sólo se aplica al último de ellos; el resto indican el lugar exacto en el que debe encontrarse el selector que va a recibir el estilo.

- **Combinación de selectores**:

```
p, a, span, em {
     color: #001100;
}
```

- **Selector descendente**:

```
p a span em {
     color: #001100;
}
```

2.3.1 Selector hijo

Un **selector hijo** es un caso concreto de un selector descendiente; en él, **un selector está contenido directamente en otro, sin que existan niveles intermedios**.

Un selector hijo se escribe a continuación de su selector padre separándolo de él por el símbolo "mayor que" (**>**).

En el siguiente **ejemplo** se pone en gris el fondo del texto enfatizado, pero **sólo si es hijo directo** de un párrafo.

```
p > em {
      background-color: gray;
      }
```

2.3.2 Selector adyacente

El **selector adyacente** se utiliza para hacer referencia a un **elemento que sigue inmediatamente a otro** en el código, **con** el que comparte **el mismo elemento padre**. Un selector adyacente se escribe a continuación de otro selector separándolo de él por el símbolo de suma (**+**).

El siguiente **ejemplo** pondría en color azul el primer párrafo que sigue a una cabecera de primer nivel.

```
      h1 + p {
            color: blue;
      }
```

2.3.3 Selector de hermano general

El selector de hermano general se utiliza para seleccionar todos los elementos que son hermanos de un elemento determinado.

El combinador ~ separa dos selectores y **selecciona el segundo elemento sólo si está precedido por el primero (no tiene por qué ser inmediatamente) y ambos comparten un padre común**.

El siguiente **ejemplo** pondrá en color rojo a aquel atributo ****** que va precedido por un párrafo ***<p>***, sólo si ambos comparten el mismo padre.

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

2.4 Selector de atributos

Los **selectores de atributos** permiten elegir elementos HTML en función de sus atributos y/o **valores** de esos atributos.

Los **tipos de selectores de atributos** son los siguientes:

- **[attr]**, selecciona los elementos que tienen el atributo *attr*.

```
/* Elementos <a> con un atributo title */
a[title] {
  color: purple;
}
```

- **[attr=value]**, selecciona los elementos cuyo atributo *attr* tenga exactamente el valor *value*.

```
/* Elementos <a> con un href que coincida con
"https://example.org" */
a[href="https://example.org"] {
  color: green;
}
```

- **[attr~=value]**, selecciona los elementos que tienen establecido un atributo llamado attr y al menos uno de los valores del atributo es value. Es decir, selecciona aquellos atributos cuyo valor contenga la **palabra** especificada.

```
/* Elementos <a> cuyo atributo class contenga la palabra "logo" */
a[class~="logo"] {
  padding: 2px;
}
```

- **[attr|=value]**, selecciona los elementos cuyo atributo *attr* tenga exactamente el valor *value* o empiece por *value* seguido de un guión - . Se puede usar para coincidencias de subcódigos en otros idiomas.

```
/* selecciona el elemento lang que empieza con el valor "es"
    y después tenga un guión sin importar lo que haya detrás" */

h2[lang |= "es"]{
    color: blue;
}
```

- **[attr$=value]**, selecciona aquellas etiquetas cuyo atributo acabe en ese valor.

```
/* Elementos <a> con un href que termine en ".org" */
a[href$=".org"] {
  font-style: italic;
}
```

- **[attr^=value]**, selecciona aquellas etiquetas cuyo atributo comience por ese valor.

```
/* Elementos <a> con un href que comience con "#" */
a[href^="#"] {
  color: #001978;
}
```

- **[attr*=value]**, selecciona aquellas etiquetas cuyo atributo contenga el valor especificado.

```
/* Elementos <a> con un href que contenga "example" */
a[href*="example"] {
  font-size: 2em;
}
```

3. Selectores de clase e identificadores

Los selectores se utilizan para dar un determinado formato a un conjunto de elementos. Si no se indica lo contrario, se aplicará el mismo estilo a todos los elementos comunes, es decir, a todos los elementos tipo <h1>, <p> o cualquier otro que se seleccione. Ahora bien, hay ocasiones en las que, a pesar de pertenecer al mismo tipo de elementos, no es deseable que presenten el mismo estilo. Para ello se utilizan los selectores de tipo **clase** o tipo **identificador**. A grandes rasgos, ambos distinguen usando una etiqueta identificadora o de clase, a través de la cual se seleccionarán los elementos y se les aplicará el estilo oportuno.

3.1 Clases (Class)

Los selectores de clase modifican y **dan estilo solo a aquellos elementos agrupados** bajo un atributo de tipo 'class' en el código estructurado en HTML, y cuyo nombre de clase debe coincidir con el nombre de la clase definida bajo ese atributo. De esta forma, a **dos o más elementos HTML distintos**, definidos con la misma clase, se les aplicará los mismos estilos.

El nombre de la clase buscada debe ser exactamente el mismo nombre contenido en el atributo **'class'** en HTML. Se utiliza para agrupar a un conjunto de etiquetas que forman parte del mismo grupo o clase, aunque no sean del mismo tipo.

El nombre que se da a una clase debe empezar por el carácter punto '**.**' y es **sensible a mayúsculas.**

La **sintaxis** del selector basado en clase es la siguiente:

```
.clase{                                  <etiqueta class=clase>
      propiedad: valor;                      …
}                                        </etiqueta>
```

A continuación, se presenta un **ejemplo** sencillo:

CÓDIGO CSS:

```
.clase1 { background-color: blue; }
```

CÓDIGO HTML:

```
<h1 class="clase1"> Esta es la clase 1.
</h1>
```

En este ejemplo la clase queda definida de forma general para todas las etiquetas identificadas con la etiqueta *class="clase1*". Ahora bien, podría utilizarse una clase de forma

más concreta, es decir, que aplique sólo sobre las etiquetas de un determinado tipo. Para ello bastaría con añadir antes del símbolo "." el nombre de la etiqueta deseada.

```
p.clase1 {
      background-color: blue;
      }
```

3.2 Identificadores (ID)

Los selectores basados en identificadores seleccionan un atributo al que dar formato buscando en el contenido de los atributos 'id' del documento estructurado en HTML. La llamada al identificador desde la hoja de estilo se hace precedida del carácter '#'. Es **sensible a mayúsculas**.

Los selectores 'id' son únicos, solo puede existir uno con el mismo nombre de atributo para toda la página web.

La diferencia entre las clases y los identificadores es que un **identificador está asociado a un único elemento** de la página mientras que una clase puede estar asociada a varios elementos.

La **sintaxis** del selector basado en identificador es la siguiente:

```
#identificador{          ──────────►  <etiqueta id = identificador>
      propiedad: valor;                    …
}                                      </etiqueta>
```

A continuación, se presenta un **ejemplo** sencillo:

CÓDIGO CSS:

```
#anexos { font-size: 10px; text-align:
center; }
```

CÓDIGO HTML:

```
<h1 id="anexos"> Anexos 1
</h1>
<h2 id="textos"> Textos
</h2>
```

Podría utilizarse un identificador de forma más concreta, es decir, que aplique sólo sobre las etiquetas de un determinado tipo. Para ello bastaría con añadir antes del símbolo # el nombre de la etiqueta deseada. Este identificador recibe el nombre de **identificador dependiente**.

```
div#anexos {
     font-size: 10px; text-align: center;
     }
```

El selector 'id' prioriza siempre a las reglas globales; es decir, si se ha definido que todos los párrafos sean de color negro de forma general, y para el elemento señalado con 'id' deben ser de color verde, no importa si en la hoja de estilo se define antes un estilo o el otro, siempre tendrán prioridad sobre las reglas globales.

3.3 Combinación de clases e identificadores

Existe la posibilidad de aplicar estilos, al mismo tiempo, a elementos identificados mediante clases e identificadores. Para ello basta con añadir al elemento un parámetro de cada tipo, **'class'** e **'id'**, de tal forma que el elemento tendrá el estilo definido para la clase y para el identificador.

La **sintaxis** del selector basado en la combinación de identificador y clase sería la siguiente:

```
.clase{                          <etiqueta id=identificador class=clase>
     propiedad: valor;               …
}                                </etiqueta>

#identificador{
     propiedad: valor;
}
```

A continuación, se presenta un **ejemplo** sencillo:

CÓDIGO CSS:

```
h1.clase1 {
      background-color: blue;
      }

#destacado {
      font-size: 18px;
      text-align: center;
      }
```

CÓDIGO HTML:

```
<h1 id="destacado" class="clase1">
Titulo. </h1>
<h1 class="clase1"> Subtitulo. </h1>
```

En este ejemplo, el primer elemento queda definido por un identificador (id="destacado") y por una clase (class="clase1"). Por otro lado, el segundo solo está identificado por un atributo de tipo clase (class="clase1"). Desde la hoja de estilos, todos aquellos elementos de tipo h1 definidos bajo la clase1, aparecerán con un fondo de color azul, así mismo, aquellos elementos que, además, están identificados con el id="destacado", se mostrarán con el estilo marcado en la hoja de estilos CSS en el elemento #destacado; esto es, presentarán un tamaño de 18 px y estarán centrados.

4. Pseudoselectores: pseudoclases y pseudoelementos.

En este apartado se explica en qué consisten los pseudoselectores, en concreto, los dos tipos de estos que pueden encontrarse:

1) Las **pseudoclases**, utilizadas para modificar el estilo de un elemento que puede presentar varios estados.

2) Los **pseudoelementos**, los cuales modifican el estilo de determinados elementos especiales que no pueden ser modificados a través de otros selectores ni de pseudoclases.

Antes de decidir usar los pseudoselectores en tu código, **comprueba que son compatibles** con el tipo y versión de navegador web que va a utilizar los usuarios de tu página web.

4.1 Pseudoclases

Existen ciertos tipos de elementos que pueden presentar más de un estado. En el diseño de páginas web es interesante que, en función del estado en el que se encuentre el elemento, se presente un estilo u otro. Por ejemplo, en el caso de los enlaces, es común que estos elementos cambien su apariencia en función de si han sido visitados o no. Por lo tanto, podrían definirse pseudoclases como aquellos selectores utilizados para **definir las propiedades de elementos con diferentes estados**.

4.1.1 Pseudoclases de enlaces: link, visited, hover y active

Como ya se ha dicho, uno de los casos más usuales es la de los enlaces, que quedan definidos mediante la etiqueta `<a>`, la cual puede presentar diferentes **estados**, tal y como se muestran a continuación:

a:link	**Enlaces que no han sido visitados por el usuario.**
a:visited	Enlaces visitados al menos una vez por el usuario.
:hover	El elemento se activa, modifica su estilo, cuando cualquier elemento apuntador pasa por encima de dicho elemento. NO SE ACTIVA CON EL DEDO, por lo que no es funcional en dispositivos móviles sin elemento apuntador.

:active	Este estado se activa cuando el usuario activa un elemento, normalmente, al pulsar con el ratón sobre el elemento. Es decir, indica **cuando el clic se está ejecutando**.

Se debe tener en cuenta que **las pseudoclases deben aparecer siempre en un determinado orden**. Este orden es: ***:link***, ***:visited***, ***:hover*** y ***:active*****.**

Las pseudoclases ***:active*** *y* ***:hover*** ***se pueden usar con todos los elementos, no solo en links.***

En el siguiente **ejemplo** sencillo, el enlace va a cambiar su apariencia al pasar el ratón sobre él, gracias al estado **a:hover** aparecerá de color rojo. Cuando haya sido visitado al menos una vez, pasará a ser de color gris, tal y como se indica en **a:visited**. En el resto de casos, conservará el mismo estado que el resto de los elementos de la página, al menos lo que no queda definido en **a:link**.

CÓDIGO CSS:

```
a:link { font-family: sans-serif; }
a:hover { color:red; }
a:visited { color:gray; }
```

CÓDIGO HTML:

```
<a href="...">
     <h1> Enlace </h1>
</a>
```

4.1.2 focus

Otra de las pseudoclases utilizadas en aquellos elementos que aceptan eventos de teclado u otras entradas del usuario, como los **formularios**, es ***:focus***.

Utilizando la pseudoclase ***:focus***, el elemento **se activa cuando tiene el foco del navegador** sobre él, por ejemplo, al seleccionar un elemento en un formulario.

Un **ejemplo** sería:

```
input:focus {background-color: yellow;}
```

4.1.3 lang

:lang(idioma) **hace referencia al idioma** en el que está un determinado elemento.

En el siguiente **ejemplo** se aplica el estilo a cualquier párrafo que esté escrito en inglés:

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

4.1.4 Pseudoclases avanzadas

Existen más pseudoclases que las vistas hasta ahora. Algunas de las más comunes son las que se describen a continuación:

Pseudo-clase	Descripción	Ejemplo
:first-child	Selecciona el elemento que es el **primer hijo** de su padre.	***p:first-child*** Selecciona cada <p> que sea el primer hijo de su padre.
:last-child	Selecciona el elemento que es el **último hijo** de su padre.	***p:last-child*** Selecciona cada <p> que sea el ultimo hijo de su padre.
:first-of-type	Selecciona el selector especificado, solo si es el primer hijo de su padre. Es decir, selecciona el **primer hermano de su tipo**.	***p:first-of-type*** Selecciona cada <p> que sea el primer elemento hijo de tipo <p> de su padre.
:last-of-type	Selecciona el selector especificado, solo si es el último hijo de su padre. Es decir, selecciona el **último hermano de su tipo.**	***p:last-of-type*** Selecciona cada <p> que sea el último elemento hijo de tipo <p> de su padre.
:only-child	Selecciona al elemento si es el **único hijo** de su padre.	***p:only-child*** Selecciona cada <p> que sea el único hijo de su padre.
:only-of-type	Selecciona cada elemento que sea el **único hijo de ese tipo**, de ese padre.	***p:only-of-type*** Selecciona cada <p> que sea el único <p> de su padre.
:empty	Selecciona el elemento sólo si está vacío, es decir, **no tiene hijos**.	***p:empty*** { width: 100px; height: 20px; background: #ff0000;} Selecciona cada <p> que no tenga hijos.

:nth-child(n)	Selecciona el elemento, el cual es el **enésimo hijo dentro de su padre**. Es decir, n es el número que ocupa el hijo dentro del padre. n puede ser un número, una keyword (odd o even), o una fórmula (como an + b).	`p:nth-child(2)` Selecciona el segundo elemento dentro de un mismo padre, que tiene que ser de tipo <p>.
:nth-last-child(n)	Selecciona el **enésimo elemento dentro de su padre, independientemente del tipo, contando desde el último hijo.** Es decir, n es el número que ocupa el hijo dentro del padre, contando desde el último hijo. n puede ser un número, una keyword (odd o even), o una fórmula (como an + b).	`p:nth-last-child(2)` Selecciona el segundo elemento contando desde el último elemento contenido dentro de un mismo padre, y que tiene que ser de tipo <p>.
:nth-of-type(n)	Selecciona el n elemento hijo del tipo que indiquemos, dentro de su padre. Es decir, selecciona el **enésimo hermano de su tipo**. n puede ser un número, una keyword (odd o even), o una fórmula (como an + b).	`p:nth-of-type(2)` Selecciona cada <p> que sea el segundo elemento de tipo <p> de su padre.
:nth-last-of-type(n)	Selecciona el n elemento hijo del tipo que indiquemos, dentro de su padre, contando desde el último hijo. Es decir, selecciona el **enésimo hermano de su tipo comenzando desde el último**. n puede ser un número, una keyword (odd o even), o una fórmula (como an + b).	`p:nth-last-of-type(2)` Selecciona cada <p> que sea el segundo elemento de tipo <p> de su padre, contando desde el último hijo.
:not(selector)	Selecciona cada elemento que **no sea el selector que se ha indicado**.	`:not(p)` Selecciona cada elemento que no sea un elemento <p>
:root	Se selecciona el **elemento raíz** del documento.	`root` En HTML, el elemento raíz es siempre el element html.
:enabled	Selecciona cada elemento que esté **activo o habilitado**.	`input:enabled` Selecciona cada elemento <input> que esté activo.
:disabled	Selecciona cada elemento que esté **deshabilitado**.	`input:disabled` Selecciona cada elemento <input> que esté deshabilitado.
:checked	Selecciona los elementos <input> de radio buttons y checkboxes, así como los elementos <option>.	`input:checked` Selecciona cada elemento <input> de tipo check (radio bottons y checkboxes).

En el siguiente **ejemplo**, se ha creado una lista de cinco elementos mediante ***<ul>*** en código HTML. Desde la hoja de estilo CSS se da formato al primero (***first-child***) y al último elemento (***last-of-type***), que toman un estilo diferente al resto de elementos del documento HTML.

CÓDIGO CSS:

```
li:first-child {
      font-size: 22px;
      color: #333399;
      list-style-type: none;
      }

li:last-of-type {
      font-size: 16px;
      color: #333399;
      font-weight:bolder;
      }
```

CÓDIGO HTML:

```
<ul>
      <li>Nombre y
cantidad</li>
      <li>Pepe. 5</li>
      <li>Maria. 8</li>
      <li>Manuel. 3</li>
      <li>Total. 16</li>
</ul>
```

4.2 Pseudoelementos

Los **pseudoelementos** permiten modificar el estilo de determinados elementos especiales, que no pueden ser modificados a través de selectores ni de pseudoclases. Normalmente, no describen un estado, sino que **añaden estilos a una parte concreta del elemento al que están vinculados**. **Si el elemento se mueve, el pseudoelemento lo acompaña**.

Los pseudoelementos más comunes son ***first-line***, ***first-letter***, ***:before*** y ***:after***.

De forma general, la **sintaxis** de uso es:

```
selector::pseudo-elemento {
    propiedad:valor;
}
```

Los pseudo elementos **utilizan dos puntos dobles** (::) en vez de solo uno (:). Esto forma parte de CSS3 y de un intento para **distinguir pseudoelementos de pseudoclases**.

Algunas **características** principales de los pseudoelementos son:

- Todo pseudoelemento tiene que tener un **content**.

- Los pseudoelementos **no son seleccionables por el usuario**, ya que son código CSS y no HTML.
- Los pseudoelementos **no están pensados para poner letra**, si no que están pensados para "darle" algo más al elemento, que tenga que ver con su estilo o apariencia visual.
- **Su orden** en el código CSS **no es relevante**, es decir, el pseudoelemento se puede poner antes o después en el código.

A continuación, se describen los **pseudoelementos más comunes**:

⇒ **:first-line**

Este pseudoelemento **selecciona la primera línea** de texto de un elemento (calculada por el navegador, independientemente del tamaño que tenga). Sólo se emplea en **elementos de bloque.**

En el siguiente **ejemplo** en código CSS, la primera línea será de color rojo.

```
p::first-line {
      color:red;
}
```

⇒ **:first-letter**

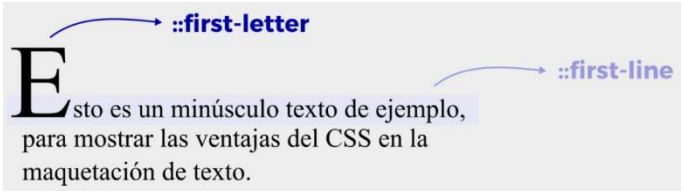
Selecciona **la primera letra de la primera línea** de texto de un elemento. En el ejemplo, la primera letra se mostrará de color verde. Sólo se emplea en **elementos de bloque.**

En el siguiente ejemplo en código CSS, la primera letra de la primera línea será de color verde.

```
p::first-letter {
      color:green;
}
```

Otro ejemplo de :first-letter y :first-line:

```css
p {
  font-family: Verdana, sans-serif;
  font-size: 16px;
  color: #333;
}

p::first-letter {
  font-family: 'Times New Roman', serif;
  font-size: 42px;
  color: black;
}
 p::first-line {
   color: #999;
 }
```



⇒ **:before y :after**

Estos dos pseudoelementos se utilizan para **añadir contenido** al documento original. En el caso de ***:before*** se añade al principio, y en el caso de ***:after***, después.

En la **sintaxis** de uso debe aparecer la propiedad '***content***', donde se indica el nuevo contenido que se va a incorporar.

- Ejemplo de ***:before***:

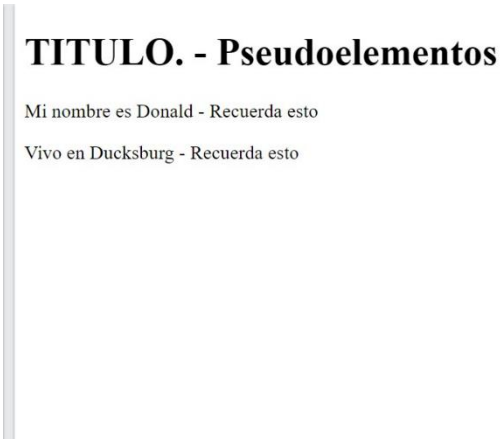
```css
h1::before {
      content: "INICIO.-";
}
```

- Ejemplo de ***:after***:

```css
h1::after {
      content: ".-FIN" ;
}
```

Otro ejemplo de :after y :before

```
<head>
<style>

h1::before {
content: "TITULO. - ";
}

p::after {
  content: " - Recuerda esto";
}

</style>
</head>
<body>

<h1>Pseudoelementos</h1>

<p>Mi nombre es Donald</p>
<p>Vivo en Ducksburg</p>

</body>
</html>
```

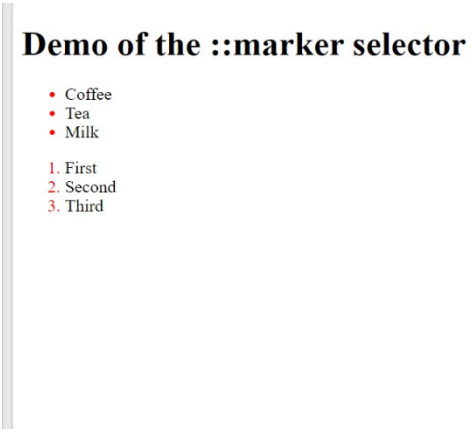


⇒ :marker

El pseudoelemento :marker aplica **estilos** a los elementos de cada ítem de una **lista**. Es decir, hace referencias a los signos o marcas de las listas (*<ol>* o *<ul>*), en el caso de que queramos que tengan, por ejemplo, un color diferente al del texto de la lista.

```
ul ::marker {
  color: green;
}
```

```
<head>
<style>
::marker {
  color: red;
}
</style>
</head>
<body>

<h1>Demo of the ::marker selector</h1>

<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>

<ol>
  <li>First</li>
  <li>Second</li>
  <li>Third</li>
</ol>
```

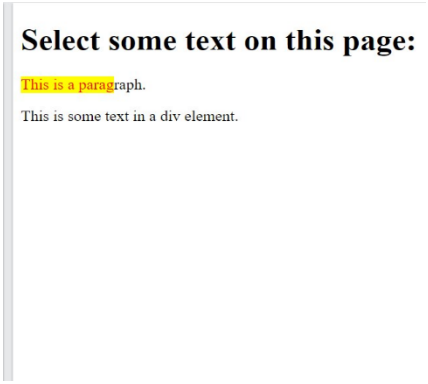


⇒ :selection

Este pseudoelemento aplica **estilos al fragmento de texto seleccionado** por el usuario.

Cuando marcamos un fragmento de texto, el navegador suele aplicar un color de fondo. Es posible que queramos aprovechar esto para definir un color que tenga sentido con los colores corporativos de la marca, página o similar, por lo que podríamos cambiarlo haciendo uso de pseudoelemento ***::selection***.

```
<!DOCTYPE html>
<html>
<head>
<style>
::selection {
  color: red;
  background: yellow;
}
</style>
</head>
<body>

<h1>Select some text on this page:</h1>

<p>This is a paragraph.</p>
<div>This is some text in a div element.</div>

</body>
</html>
```



5. Buenas prácticas CSS

Seguir unas pautas a la hora de crear estilos CSS hace que el código sea más claro y legible tanto para nosotros como para otros desarrolladores que deban trabajar en el proyecto.

Aunque no hay definido ningún estándar al respecto, es recomendable seguir las siguientes **recomendaciones** para conseguir un desarrollo lo más entendible posible.

1) Organizar la estructura de arriba hacia abajo

Con el fin de encontrar rápidamente cualquier parte de código, es recomendable organizar los estilos de arriba hacia abajo.

```
/****** generic classes*******/
styles goes here…

/****** header *******/
styles goes here…

/****** nav menu *******/
styles goes here…

/****** main content *******/
styles goes here…

/****** footer *******/
```

2) Nombrar correctamente los selectores

Para conseguir más claridad en el código y soporte en todos los navegadores conviene no comenzar el nombre de los selectores con mayúsculas, números ni caracteres especiales.

Evitar: ~~#1div~~, ~~.=div~~, ~~DivContent~~

Mejor utilizar: #div1, .div, divContent

3) Separar las palabras mediante guiones o mediante mayúsculas

Es recomendable escoger una única manera de escribir el nombre de los selectores. Además, es mejor no usar guiones bajos "_" u otros caracteres especiales ya que algunos navegadores no los soportan. Conviene seguir alguna de las siguientes opciones:

```
/* Opción 1: Palabras separadas por mayúsculas */
navMenu { padding: 2em; border: 2px solid green; }

/* Opción 2: Palabras separadas por guiones */
nav-menu { padding: 2em; border: 2px solid green; }
```

4) Legibilidad

Adopta una única forma de escribir tu código para que sea más fácil de mantener y de encontrar cualquier elemento.

```
/* Opción 1: Estilos en una línea */
.nav-menu { padding: 2em; border: 2px solid green; }

/* Opción 2: Cada estilo en una línea */
.nav-menu {
      padding: 2em;
      border: 2px solid green;
}
```

5) Combinar elementos

Cuando varios elementos disponen de las mismas propiedades es recomendable compartirlas en vez de volver a repetir el código. Para ello podemos utilizar selectores combinados, tal y como hemos visto anteriormente.

```
h1, h2, h3 { font-family: Arial; font-weight: 700;  }
```

6) Utilizar selectores descendientes

Es conveniente utilizar selectores descendientes siempre que sea posible antes de crear un selector clase o un selector identificador. De esta forma, el código estará más limpio, dispondrá de menos selectores clase e identificador y se comprenderá mucho mejor.

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

7) Utilizar propiedades abreviadas

Siempre que sea posible es recomendable utilizar clases abreviadas para conseguir una reducción de código.

```
/* Propiedades margin-left, margin-right y margin-top */
.nav-menu {margin-left: 5px; margin-right: 5px; margin-
top: 5px;}

/* Propiedad abreviada margin */
.nav-menu {margin: 5px 5px 0px 5px;}
```

8) Utilizar nombres descriptivos en los selectores

Mediante la utilización de nombres que permitan averiguar fácilmente a qué elemento le estamos dando estilo comprenderemos el código fácilmente.

```
.nav-button { background: blue; } /* Estilo del botón de
la navbar*/
```

9) Evitar utilizar como nombre de un selector una característica visual

Al utilizar en el nombre de un selector una característica visual como el color, el tamaño o la posición, si posteriormente modificamos esa característica, también deberíamos cambiar el nombre del selector. Esto complica mucho el código ya que tendríamos que actualizar todas las referencias a ese selector en el código HTML.

```
/* Selector con nombre que define la característica visual
del color */
.menu-red { background: red; }

/* Utilizar mejor: */
.nav-menu { background: red; }
```

10) Probar el diseño en los diferentes navegadores

Para descubrir si se producen errores de visualización en los navegadores lo recomendable es instalarlos todos en el equipo. Algunos de los más utilizados son los siguientes:

- Chrome
- Firefox
- Opera
- Safari
- Microsoft Edge

También puedes hacer uso de la aplicación browserling que te permite ver tu desarrollo en varias versiones diferentes de cada navegador.

11) Validar el código CSS

Detectar errores en el código es esencial y se puede hacer fácilmente mediante un validador CSS. El W3C proporciona una herramienta de validación de CSS gratuita para los documentos CSS.

12) Agregar los prefijos de los navegadores en propiedades que no sean estables

Para ahorrar tiempo y facilitarnos la tarea de incluir los prefijos de las propiedades CSS que todavía no son estables podemos hacer uso de la extensión “Autoprefixer” en Visual Studio Code.