Capítulo 2. Selectores

Conocer y dominar todos los selectores de CSS es imprescindible para crear diseños web profesionales.

Utilizando solamente los cinco selectores básicos de CSS (universal, de tipo, descendente, de clase y de id) es posible diseñar cualquier página web. No obstante, los selectores avanzados de CSS3 permiten simplificar las reglas CSS y también el código HTML.

2.1.1 Selector, hijos

Se trata de un selector similar al selector descendente, pero muy diferente en su funcionamiento. Se utiliza para seleccionar un elemento que es *hijo* de otro elemento y se indica mediante el *"signo de mayor que"* (>).

Mientras que en el selector descendente sólo importa que un elemento esté dentro de otro, independientemente de lo profundo que se encuentre, en el selector de hijos el elemento debe ser *hijo directo* de otro elemento.

```
span { color: blue; }

<p>
   <span>Texto1</span>
</p>
<p>
    <a href="#">
       <span>Texto2</span>
    </a>
</p>
```

En el ejemplo anterior, el selector `p > span` se interpreta como *"cualquier elemento `<span>` que sea hijo directo de un elemento `<p>`"*, por lo que el primer elemento `<span>` cumple la condición del selector. Sin embargo, el segundo elemento `<span>` no la cumple porque es descendiente pero no es hijo directo de un elemento `<p>`.

Utilizando el mismo ejemplo anterior se pueden comparar las diferencias entre el selector descendente y el selector de hijos:

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

```
<p>
    <a href="#">Enlace1</a>
</p>
<p>
  <span>
     <a href="#">Enlace2</a>
  </span>
</p>
```

El primer selector es de tipo descendente (`p a`) y por tanto se aplica a todos los elementos `<a>` que se encuentran dentro de elementos `<p>`. En este caso, los estilos de este selector se aplican a los dos enlaces.

El segundo selector es de hijos (`p > a`) por lo que obliga a que el elemento `<a>` sea hijo directo de un elemento `<p>`. Por tanto, los estilos del selector `p > a` no se aplican al segundo enlace del ejemplo anterior.

2.1.2 Selector adyacente

El selector adyacente se emplea para seleccionar elementos que son *hermanos* (su elemento padre es el mismo) y están seguidos en el código HTML. Este selector emplea en su sintaxis el símbolo +. Si se considera el siguiente ejemplo:

```
h1 + h2 { color: red }
<body>
  <h1>Título1</h1>
  <h2>Subtítulo</h2>
  <h2>Otro subtítulo</h2>
</body>
```

Los estilos del selector `h1 + h2` se aplican al primer elemento `<h2>` de la página, pero no al segundo `<h2>`, ya que:

- El elemento padre de `<h1>` es `<body>`, el mismo padre que el de los dos elementos `<h2>`. Así, los dos elementos `<h2>` cumplen la primera condición del selector adyacente.
- El primer elemento `<h2>` aparece en el código HTML justo después del elemento `<h1>`, por lo que este elemento `<h2>` también cumple la segunda condición del selector adyacente.
- Por el contrario, el segundo elemento `<h2>` no aparece justo después del elemento `<h1>`, por lo que no cumple la segunda condición del selector adyacente y por tanto no se le

aplican los estilos de `h1 + h2`.

El siguiente ejemplo puede ser útil para los textos que se muestran como libros: `p + p { text-indent: 1.5em; }`

En muchos libros es habitual que la primera línea de todos los párrafos esté indentada, salvo la primera línea del primer párrafo. El selector `p + p` selecciona todos los párrafos que están dentro de un mismo elemento padre y que estén precedidos por otro párrafo. En otras palabras, el selector `p + p` selecciona todos los párrafos de un elemento salvo el primer párrafo.

El selector adyacente requiere que los dos elementos sean *hermanos*, por lo que su elemento *padre* debe ser el mismo. Si se considera el siguiente ejemplo:

```
p + p { color: red; }
<p>Lorem ipsum dolor sit amet...</p>
<p>Lorem ipsum dolor sit amet...</p>
<div>
  <p>Lorem ipsum dolor sit amet...</p>
</div>
```

En el ejemplo anterior, solamente el segundo párrafo se ve de color rojo, ya que:

- El primer párrafo no va precedido de ningún otro párrafo, por lo que no cumple una de las condiciones de `p + p`
- El segundo párrafo va precedido de otro párrafo y los dos comparten el mismo padre, por lo que se cumplen las dos condiciones del selector `p + p` y el párrafo muestra su texto de color rojo.
- El tercer párrafo se encuentra dentro de un elemento `<div>`, por lo que no se cumple ninguna condición del selector `p + p` ya que ni va precedido de un párrafo ni comparte padre con ningún otro párrafo.

2.1.3 Selector, general de elementos hermanos,

Generaliza el selector adyacente. Su sintaxis es `elemento1 ~ elemento2` y selecciona el `elemento2` que es hermano de `elemento1` y se encuentra detrás en el código HTML. En el selector adyacente la condición adicional era que los dos elementos debían estar uno detrás de otro en el código HTML, mientras que ahora la única condición es que uno esté detrás de otro.
Si se considera el siguiente ejemplo:

```
h1 + h2 { ... } /* selector adyacente */
```

```
h1 ~ h2 { ... } /* selector general de hermanos */
<h1>...</h1>
<h2>...</h2>
<p>...</p>
<div>
   <h2>...</h2>
</div>
<h2>...</h2>
```

El primer selector (`h1 + h2`) sólo selecciona el primer elemento `<h2>` de la página, ya que es el único que cumple que es hermano de `<h1>` y se encuentra justo detrás en el código HTML. Por su parte, el segundo selector (`h1 ~ h2`) selecciona todos los elementos `<h2>` de la página salvo el segundo. Aunque el segundo `<h2>` se encuentra detrás de `<h1>` en el código HTML, no son elementos hermanos porque no tienen el mismo elemento padre.

2.1.4. Selector, atributos

El último tipo de selectores avanzados lo forman los selectores de atributos, que permiten seleccionar elementos HTML en función de sus atributos y/o valores de esos atributos.

Los cuatro tipos de selectores de atributos son:

- `[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 cuyo valor es una lista de palabras separadas por espacios en blanco en la que al menos una de ellas es exactamente igual a `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`. Este tipo de selector sólo es útil para los atributos de tipo `lang` que indican el idioma del contenido del elemento.
- `elemento[atributo^="valor"]`, selecciona todos los elementos que disponen de ese atributo y cuyo valor comienza exactamente por la cadena de texto indicada. (Acento circunflejo, Alt +94).
- `elemento[atributo$="valor"]`, selecciona todos los elementos que disponen de ese atributo

y cuyo valor termina exactamente por la cadena de texto indicada.

- `elemento[atributo*="valor"]`, selecciona todos los elementos que disponen de ese atributo y cuyo valor contiene la cadena de texto indicada.

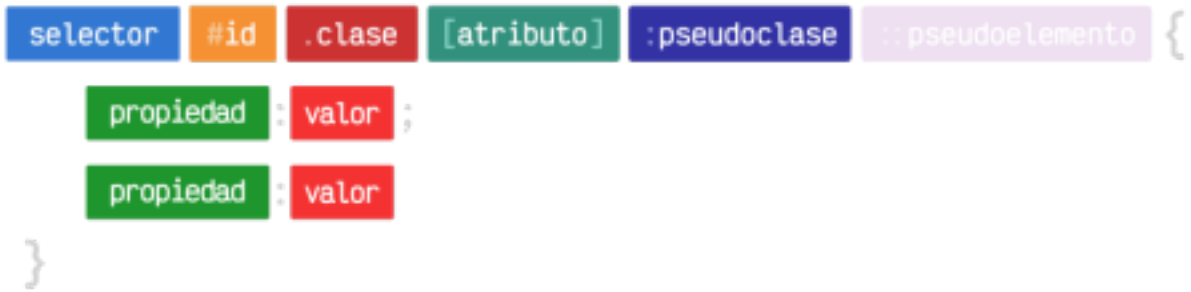
A continuación, se muestran algunos ejemplos de estos tipos de selectores:

```
/* 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 }

/* Selecciona todos los enlaces que apuntan a una dirección de correo
electrónico */

a[href^="mailto:"] { ... }
```

```
/* Selecciona todos los enlaces que apuntan a una página HTML */

a[href$=".html"] { ... }

/* Selecciona todos los títulos h1 cuyo atributo title contenga la palabra
"capítulo" */

h1[title*="capítulo"] { ... }
```

2.2. Pseudo-clases

Las **pseudoclases** se utilizan para hacer referencia a elementos HTML que tengan un **cierto comportamiento concreto**. Podremos seleccionar elementos que en principio parecen iguales, pero tienen diferentes características de comportamiento.



2.2.1. Pseudoclase de estructura

:root

La pseudoclase **:root** hace referencia al elemento raíz del documento HTML, o lo que es lo mismo, la etiqueta <html>. Sin embargo, en muchas ocasiones veremos que en lugar de utilizar directamente la etiqueta, se utiliza la pseudoclase :root. Al ser una pseudoclase, tiene una especificidad CSS más alta (0,1,0) que el elemento html, el cuál, al ser una etiqueta HTML, tiene una especificidad más baja (0,0,1):

```
:root {
 background: black;
}
```

:root se usa habitualmente en CSS para definir variables CSS globales (también llamadas custom properties) que se pueden reutilizar en todo el documento.

```
:root {
 --primary-color: #007bff;
 --secondary-color: #6c757d;
}
body {
 font-size: var(--font-size-base);
 font-family: var(--font-family);
 }
```

:first-child

La pseudo-clase `:first-child` selecciona el primer elemento hijo de un elemento. Si se considera el siguiente ejemplo:

```
 p em:first-child {
color: red;
}
<p>Lorem <span><em>ipsum dolor</em></span> sit amet, consectetuer
adipiscing elit. Praesent odio sem, tempor quis, <em>auctor eu</em>, tempus
at, enim. Praesent nulla ante, <em>ultricies</em> id, porttitor ut,
pulvinar quis, dui.
</p>
```

El selector `p em:first-child` selecciona el primer elemento `<em>` que sea hijo de un elemento y que se encuentre dentro de un elemento `<p>`. Por tanto, en el ejemplo anterior sólo el primer `<em>` se ve de color rojo.

La pseudo-clase `:first-child` también se puede utilizar en los selectores simples, como se muestra a continuación: `p:first-child { ... }`

La regla CSS anterior aplica sus estilos al primer párrafo de cualquier elemento. Si se modifica el ejemplo anterior y se utiliza un selector compuesto:

```
p:first-child em {
            color: red;
          }
```

```
<body>

  <p>Lorem <span><em>ipsum dolor</em></span> sit amet, consectetuer
adipiscing elit. Praesent odio sem, tempor quis, <em>auctor eu</em>, tempus
at, enim.</p>

  <p>Lorem <span><em>ipsum dolor</em></span> sit amet, consectetuer
adipiscing elit. Praesent odio sem, tempor quis, <em>auctor eu</em>, tempus
at, enim.</p>

  <div> <p>Lorem <span><em>ipsum dolor</em></span> sit amet, consectetuer
adipiscing elit.
Praesent odio sem, tempor quis, <span><em>auctor eu</em> </span>, tempus
at, enim.</p>
  </div>

</body>
```

El selector `p:first-child em` selecciona todos aquellos elementos `<em>` que se encuentren dentro de un elemento `<p>` que sea el primer hijo de cualquier otro elemento.

El primer párrafo del ejemplo anterior es el primer hijo de `<body>`, por lo que sus `<em>` se ven de color rojo. El segundo párrafo de la página no es el primer hijo de ningún elemento, por lo que sus elementos `<em>` interiores no se ven afectados. Por último, el tercer párrafo de la página es el primer hijo del elemento `<div>`, por lo que sus elementos `<em>` se ven de la misma forma que los del primer párrafo.

:nth-child

- `elemento:nth-child(número)`, selecciona el elemento indicado pero con la condición de que sea el hijo enésimo de su padre. Este selector es útil para seleccionar el segundo párrafo de un elemento, el quinto elemento de una lista, etc.

  Podemos seleccionar los hijos impares (odd) o pares (even) utilizando las palabras clave odd o even.

- `elemento:nth-last-child(número)`, idéntico al anterior pero el número indicado se empieza a contar desde el último hijo.
- `elemento:empty`, selecciona el elemento indicado pero con la condición de que no tenga ningún hijo. La condición implica que tampoco puede tener ningún contenido de texto.

Ej: Se podría utilizar para aplicar tramas a celdas vacías en una tabla:

Table td:empty{ background-image:url(images/pattern.png);}

- `elemento:first-child` y `elemento:last-child`, seleccionan los elementos indicados pero con la condición de que sean respectivamente los primeros o últimos hijos de su elemento padre.
- `elemento:nth-of-type(número)`, selecciona el elemento indicado pero con la condición de que sea el enésimo elemento hermano de ese tipo.
- `elemento:nth-last-of-type(número)`, idéntico al anterior pero el número indicado se empieza a contar desde el último hijo.

Algunas pseudo-clases como `:nth-child(número)` permiten el uso de expresiones complejas para realizar selecciones avanzadas:

```
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 */

/* Las siguientes reglas alternan cuatro estilos diferentes para los párrafos */

p:nth-child(4n+1) { ... }
p:nth-child(4n+2) { ... }
p:nthchild(4n+3) { ... }
p:nth-child(4n+4) { ... }
```

Empleando la pseudo-clase `:nth-of-type(número)` se pueden crear reglas CSS que alternan la posición de las imágenes en función de la posición de la imagen anterior:

```
img:nth-of-type(2n+1){ float: right; }

img:nth-of-type(2n) { float: left; }
```

`:not()`, se puede utilizar para seleccionar todos los elementos que no cumplen con la

condición de un selector:

```
:not(p) { ... } /* selecciona todos los elementos de la página que no sean párrafos */

:not(#especial) { ... } /* selecciona cualquier elemento cuyo atributo id no sea "especial" */
```

2.2.2. Pseudoclases de interacción

:link y :visited

Las pseudo-clases `:link` y `:visited` se pueden utilizar para aplicar diferentes estilos a los enlaces de una misma página:

- La pseudo-clase `:link` se aplica a todos los enlaces que todavía no han sido visitados por el usuario.
- La pseudo-clase `:visited` se aplica a todos los enlaces que han sido visitados al menos una vez por el usuario.

El navegador gestiona de forma automática el cambio de enlace no visitado a enlace visitado. Aunque el usuario puede borrar la caché y el historial de navegación de forma explícita, los navegadores también borran de forma periódica la lista de enlaces visitados.

Por su propia definición, las pseudo-clases `:link` y `:visited` son mutuamente excluyentes, de forma que un mismo enlace no puede estar en los dos estados de forma simultánea.

> Como los navegadores muestran por defecto los enlaces de color azul y los enlaces visitados de color morado, es habitual modificar los estilos para adaptarlos a la guía de estilo del sitio web:

```
a:link { color: red; } a:visited { color: green; }
```

:hover, :active y :focus

Las pseudo-clases `:hover`, `:active` y `:focus` permiten al diseñador web variar los estilos de un elemento en respuesta a las acciones del usuario. Al contrario que las pseudo-clases `:link` y `:visited` que sólo se pueden aplicar a los enlaces, estas pseudo-clases se pueden aplicar a cualquier elemento.

A continuación se indican las acciones del usuario que activan cada pseudo-clase:

- `:hover`, se activa cuando el usuario pasa el ratón o cualquier otro elemento apuntador por encima de un elemento.

- `:active`, se activa cuando el usuario activa un elemento, por ejemplo cuando pulsa con el ratón sobre un elemento. El estilo se aplica durante un espacio de tiempo prácticamente imperceptible, ya que sólo dura desde que el usuario pulsa el botón del ratón hasta que lo suelta.
- `:focus`, se activa cuando el elemento tiene el foco del navegador, es decir, cuando el elemento está seleccionado. Normalmente se aplica a los elementos `<input>` de los formularios cuando están activados y por tanto, se puede escribir directamente en esos campos.

De las definiciones anteriores se desprende que un mismo elemento puede verse afectado por varias pseudo-clases diferentes de forma simultánea. Cuando se pulsa por ejemplo un enlace que fue visitado previamente, al enlace le afectan las pseudo-clases `:visited`, `:hover` y `:active`.

Debido a esta característica y al comportamiento en cascada de los estilos CSS, es importante cuidar el orden en el que se establecen las diferentes pseudo-clases. El siguiente ejemplo muestra el único orden correcto para establecer las cuatro pseudo-clases principales en un enlace:

```
a:link { ... }
a:visited { ... }
 a:hover { ... }
a:active { ... }
```

Por último, también es posible aplicar estilos combinando varias pseudo-clases compatibles entre sí. La siguiente regla CSS por ejemplo sólo se aplica a aquellos enlaces que están seleccionados y en los que el usuario pasa el ratón por encima: `a:focus:hover { ... }`

2.2.3. Pseudoclases de formularios.

En HTML5 es posible dotar de capacidades de validación a los campos de un formulario, pudiendo interactuar desde Javascript o incluso desde CSS. Con estas validaciones podemos asegurarnos de que el usuario escribe en un campo de un formulario el valor esperado que debería.

Existen **dos categorías** de Pseudoclases:

- De **estado en formularios**: Seleccionar elementos cuando se encuentran en un estado concreto.
- De **validación**: Seleccionar elementos si cumplen o no un cierto criterio de validación.

2.2.3.1 Pseudoclases de estados

Por norma general, los elementos de un formulario HTML están por defecto activados, aunque se pueden desactivar añadiendo el atributo disabled (es un atributo booleano, no lleva valor específico). Esto es una práctica muy utilizada para impedir al usuario escribir en cierta parte de un formulario porque, por ejemplo, no es aplicable.

Existen varias pseudoclases para detectar si un campo de un formulario está activado o desactivado, o cierta información relacionada con su estado:

:checked Selecciona el elemento cuando el campo está seleccionado.

```
input:checked + span {
  color: green;
}
```

:enabled Podemos seleccionar elementos que se encuentren activados (comportamiento por defecto):

```
button:enabled {
 background-color: green;
}
```

:disabled Se seleccionan los elementos a los que se le ha añadido el atributo disabled:

```
<button disabled>Botón desactivado</button>
button:disabled {
        background-color: grey;
```

```
cursor: not-allowed;}
```

:read-only Selecciona aquellos elementos <input> de un formulario que están marcados con el atributo de sólo lectura readonly. La diferencia entre un campo con atributo disabled y un campo con atributo readonly es que la información del campo con readonly se enviará a través del formulario, mientras que la del campo con disabled no se enviará.

```
input:read-only {
 background-color: darkred;
 color: white
}
```

:placeholder-shown Se aplica al input cuando el texto del placeholder se está mostrando

```
input:placeholder-shown {
  background: #555;
  padding: 5px;
  border: 2px solid #000;
  color: white; }
```

2.2.3.2 Pseudoclases de validación

Con estas validaciones podemos asegurarnos de que el usuario escribe en un campo de un formulario el valor esperado.

Existen algunas pseudoclases útiles para las validaciones, como por ejemplo las siguientes:

:required Cuando el campo es obligatorio, o sea, tiene el atributo required.

:optional Cuando el campo es opcional (por defecto, todos los campos).

:invalid Cuando los campos no cumplen la validación HTML5.

:valid Cuando los campos cumplen la validación HTML5.

:out-of-rang Cuando los campos numéricos están fuera del rango.

:in-range Cuando los campos numéricos están dentro del rango.

En un formulario HTML es posible establecer un campo obligatorio que será necesario rellenar para enviar el formulario. Por ejemplo, el DNI de una persona que va a matricularse en un curso, o el nombre de usuario de alta en una plataforma web para identificarse. Campos que son absolutamente necesarios. HTML Input Attributes

Para hacer obligatorios dichos campos, tenemos que indicar en el HTML el atributo **required**, al cuál será posible darle estilo mediante la pseudoclase **:required**:

```
Nombre: <input type="text" required>

input:required {
 border: 2px solid blue;
}
```

Por otra parte, los campos opcionales (no obligatorios, sin el atributo required) pueden seleccionarse con la pseudoclase **:optional**:

```
input:optional {
  border: 2px solid grey;
}
```

HTML5 brinda un excelente soporte de validaciones desde el lado del cliente, pudiendo comprobar si los datos especificados son correctos o no antes de realizar las validaciones en el lado del servidor, y evitando la latencia de enviar la información al servidor y recibirla de vuelta.

Ojo: Ten en cuenta que la validación de cliente es apropiada sólo para reducir la latencia de envío/recepción al servidor, pero nunca como estrategia para evitar problemas de seguridad o similares, para la cuál se debe tener validación en el servidor siempre.

Imaginemos un campo de entrada en el que queremos obtener la edad del usuario. Nuestra intención es que solo se puedan introducir números. Para ello hacemos uso de la expresión regular [0-9]+, que significa «una o más cifras del 0 al 9»:

```
<input type="number" name="age" pattern="[0-9]+" />
```

Sin embargo, el atributo pattern permite **expresiones regulares** realmente complejas, como por ejemplo, una expresión regular para validar el formato de un DNI, ya sea en el formato nacional de España (12345678L) o en formato NIE (X1234567L), aceptando guiones si se indican:

```
<input type="text" name="dni" pattern="^([0-9]{8}[A-Z])|([XYZ][0-9]{7}[A-Z])$" />
```

Se pueden aplicar ciertos estilos dependiendo de si se cumple o no el patrón de validación, utilizando las siguientes pseudoclases:

```
input:invalid {
 background-color:darkred;
 color: white;
}

input:valid {
 background-color: green;
 color: white;
}
```

Lo ideal sería establecer un rango, algo que se suele hacer muy a menudo si tenemos campos numéricos de formulario:

```
<input type="number" name="age" min="18" max="100" />
```

Este campo permite al usuario especificar su edad, utilizando los atributos de validación min y max, que sólo permiten valores entre 18 y 100 años. Los valores fuera de este rango, no serán válidos.

De la misma forma que antes, es posible aplicar estilos para los valores fuera de rango, como dentro de rango:

```
input:out-of-range {
 background-color: darkred;
 color: white;
}

input:in-range {
 background-color: green;
 color: white;
}
```

Ejemplo de uso:

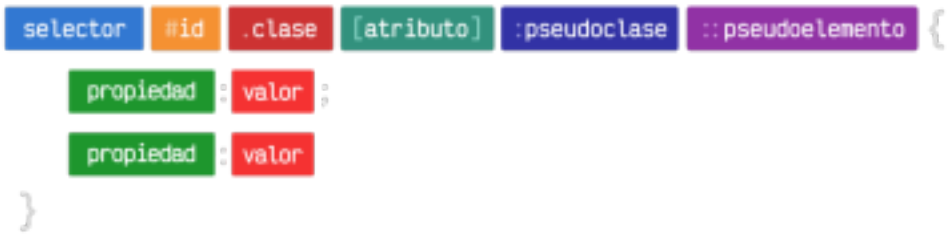
```
:required:placeholder-shown {
      border: 1px solid #888;
}
:required:not(:placeholder-shown):invalid {
      border: 1px solid red;
      box-shadow: 0 0 0 1px;
```

```
}
:required:not(:placeholder-shown):valid {
      border: 1px solid green;
      box-shadow: 0 0 0 1px;
}
:required:focus:invalid {
      border: 1px solid red;
      box-shadow: 0 0 0 1px;
}
```

2.3. Pseudo-elementos

Los selectores de CSS, las pseudo-clases y todos los elementos HTML no son suficientes para poder aplicar estilos a algunos elementos especiales. Si se desea por ejemplo cambiar el estilo de la primera línea de texto de un elemento, no es posible hacerlo con las utilidades anteriores.

La única forma de poder seleccionar estos elementos especiales es mediante los pseudo-elementos definidos por CSS para este propósito.



Los **pseudoelementos** permiten **seleccionar** y dar estilo a **elementos que no existen en el HTML**, o que no son un simple elemento en sí. La **sintaxis** de los pseudoelementos está **precedida de dos puntos dobles (::)** para diferenciarlos de las pseudoclases, las cuales sólo tienen dos puntos (:).

2.3.1. El pseudo-elemento ::first-line

El pseudo-elemento `::first-line` permite seleccionar la primera línea de texto de un elemento. Así, la siguiente regla CSS muestra en mayúsculas la primera línea de cada párrafo:

```
p::firstline { text-transform: uppercase; }
```

Este pseudo-elemento sólo se puede utilizar con los elementos de bloque y las celdas de datos de las tablas.

Se pueden combinar varios pseudo-elementos de tipo `::first-line` para crear efectos avanzados:

```
div::first-line { color: red; } p::first-line { text-transform: uppercase;
}
<div>
 <p>Lorem ipsum dolor sit amet...</p>
 <p>Lorem ipsum dolor sit amet...</p>
 <p>Lorem ipsum dolor sit amet...</p>
</div>
```

En el ejemplo anterior, la primera línea del primer párrafo también es la primera línea del elemento `<div>`, por lo que se le aplican las dos reglas CSS y su texto se ve en mayúsculas y de color rojo.

2.3.2. El pseudo-elemento ::first-letter

El pseudo-elemento `::first-letter` permite seleccionar la primera letra de la primera línea de texto de un elemento. De esta forma, la siguiente regla CSS muestra en mayúsculas la primera letra del texto de cada párrafo:

```
p::first-letter { text-transform: uppercase; }
```

Los signos de puntuación y los caracteres como las comillas que se encuentran antes y después de la primera letra también se ven afectados por este pseudo-elemento.

Este pseudo-elemento sólo se puede utilizar con los elementos de bloque y las celdas de datos de las tablas.

2.3.3. Los pseudo-elementos ::before y ::after

Los pseudo-elementos `::before` y `::after` se utilizan en combinación con la propiedad **`content`** de CSS para añadir contenidos antes o después del contenido original de un elemento.

Las siguientes reglas CSS añaden el texto `Capítulo -` delante de cada título de sección `<h1>` y el carácter `.` detrás de cada párrafo de la página:

```
h1::before { content: "Capítulo - "; }
p::after { content: "."; }
```

El contenido insertado mediante los pseudo-elementos `::before` y `::after` se tiene en cuenta en los otros pseudo-elementos `::first-line` y `::first-letter`.

2.3.4 Pseudo-elemento ::selection

- **`::selection`**, selecciona el texto que ha seleccionado un usuario con su ratón o teclado.

```
h1::selection { background-color: yellow; }
```

2.3.5 El selector ::target

Todos sabemos cómo enlazar a un elemento con un id especificado dentro de una página:

```
<p id="arriba">Arriba</p> . . . .
<a href="#arriba">Ir arriba</a>
```

El elemento enlazado es un elemento de destino, o un elemento "target". Podemos dar formato al elemento de destino utilizando el selector :target de CSS3.

```
::target {
    border: 1px solid #d9d9d9;
    background-color: #FFC;
}
```

::target representa el único elemento, si existe alguno, con un id coincidente con el identificador de fragmentos de la URL del documento.

Considera una URL como ésta: http://ejemplo.com/ejemplo.html#arriba

Si un usuario accede a la url anterior, el elemento que tiene la referencia de Id=”arriba” será el target, por lo que los estilos definidos anteriormente se aplican a ese elemento.

Ejemplo creado con grid y :target link

2.4 Agrupación de selectores:

- **Combinador :is()**

Reescribir de forma más compacta y sencilla los selectores múltiples combinados

```
.container .item,
.container .parent,
.container .element {
 /* ... */
}
```

Con el combinador :is() quedaría de la siguiente manera:

```
.container :is(.item, .parent, .element) {
 /* ... */
}
```

- **Combinador :where()**

Funciona exactamente igual que el combinador :is(). La única diferencia que tiene es en cuanto a la especificidad CSS.

Mientras que con el combinador **:is()**, la especificidad es el valor más alto de la lista de parámetros de :is(), en el caso de **:where()** la especificidad CSS es siempre cero.

```
.container :is(.list, .element, .menu) { /* Especificidad (0,2,0) */
 /* ... */
}
```

```
.container :where(.list, .element, .menu) { /* Especificidad (0,1,0) */
  /* ... */
}
```

- **Selector de estados :has()**

Permite seleccionar un elemento contenedor, siempre y cuando sus elementos hijos (descendientes) cumplan los criterios indicados por los parámetros de :has(), lo que comúnmente siempre se ha denominado el **selector padre**.

```
.contenedor:has(.contenido) { border: 2px solid blue; }
/* Este código aplicará un borde al .contenedor sólo si contiene un elemento con la clase
.contenido. */
```

:has() permite cambiar estilos en función de estados internos, lo cual ayuda a crear interactividad sin necesidad de JavaScript. Por ejemplo, al estilo de un elemento padre si su hijo está en un estado específico, como checked.

```
label:has(input:checked) { font-weight: bold; }
/* Esto aplicará un estilo en label si contiene un input que está marcado (checked).*/
```

● Nesting CSS

Anidar código CSS**,** crear componentes CSS nativos autocontenidos dentro de otros

```
.container {
  width: 800px;
  height: 300px;
  background: grey;

  .item {
    height: 150px;
    background: orangered;
  }
}
```

Dentro del CSS Nesting tenemos el **operador &,** que nos permite hacer referencia al selector inmediatamente padre dentro del anidamiento. Esto puede resultarnos muy útil en algunos casos diferentes al anterior. Observa el siguiente ejemplo:

```
.container {
  background: grey;

  &:hover {
    background: indigo;
  }
```

```
}
```

Si omitimos el **selector &**, el código CSS será válido, pero no equivalente. Omitiendo el selector & estaríamos obteniendo el selector equivalente **.container :hover** en lugar de **.container:hover.** Es decir, estaríamos aplicando estilos :hover a los elementos dentro de .container en lugar de al propio .container.