INDICE DE PSEUDO-CLASES – solo de ayuda para estudio. Documento con contenido extraído de Internet para estudiantes IFCD0110.

Las pseudoclases se utilizan para hacer referencia a elementos HTML que tengan un cierto comportamiento concreto. Volvamos a recordar el esquema general de sintaxis de CSS, donde ahora añadiremos las pseudoclases, que se definen añadiendo dos puntos antes del nombre de la pseudoclase concreta, de la siguiente forma:

```css
a {
  padding: 5px;
}

a:hover {
  background: red;
  color: white;
  text-decoration: none;
}
```

En este caso hacemos lo siguiente:

El enlace siempre tendrá un padding de 5 píxeles.

El enlace tendrá color rojo de fondo, color blanco de texto y no tendrá subrayado cuando el usuario mueva el ratón por encima.

Existen multitud de pseudoclases CSS.

Pseudoclase	Significado
Interacción	Pseudoclases relacionadas con acciones de usuario.
	:hover, :active, :focus, :focus-within, :focus-visible
Ubicación	Pseudoclases relacionadas con enlaces o ubicaciones.
	:any-link, :link, :visited, :target
Idioma	Pseudoclases relacionadas con idiomas.
	:lang(), :dir()
Estructura	Pseudoclases de estructura de documentos HTML
	:root, :host, :defined, :empty :first-child, :last-child, :only-child :first-of-type, :last-of-type, :only-of-type :nth-child(), :nth-last-child(), :nth-of-type(), :nth-last-of-type()
Formulario	Pseudoclases de formularios HTML
	:checked, :indeterminate :enabled, :disabled, :read-only, :read-write, :placeholder-shown, :default :required, :optional, :valid, :invalid, :user-valid, :user-invalid :in-range, :out-of-range

Estado	Pseudoclases relacionadas con el estado de **modales** o similares.
`:fullscreen`, `:modal`	
Paginado	Pseudoclases de paginado de documentos HTML
`:first`, `:left`, `:right`, `:blank`	

PSEUDOCLASES DE INTERACCIÓN

Pseudoclase	Descripción
`:hover`	Selecciona el elemento si el usuario pasa el ratón sobre dicho elemento.
`:active`	Selecciona el elemento si el usuario se encuentra pulsando dicho elemento.
`:focus`	Selecciona el elemento cuando tiene el foco (está en primer plano).
`:focus-within`	Selecciona el elemento si uno de sus miembros hijos ha ganado el foco.
`:focus-visible`	Selecciona el elemento cuando tiene el foco sólo de forma visible (TAB , por ejemplo).

La pseudoclase :hover

La primera de ellas, :hover, es muy útil e interesante, ya que permite aplicar estilos a un elemento justo cuando el usuario pasa el ratón (o más concretamente, un dispositivo apuntador) sobre él. Es una de las pseudoclases más utilizadas:

```css
/* Usuario mueve el ratón sobre un enlace */
a:hover {
  background-color: cyan;
  padding: 2px
}

/* Usuario mueve el ratón sobre un div y resalta todos los enlaces que contiene */
div:hover a {
  background-color: steelblue;
  color: white;
}
```

Podemos realizar acciones un poco más específicas, como el segundo ejemplo anterior, donde al movernos sobre un elemento div (div:hover), aplicaremos los estilos a los enlaces (a) que están dentro del mencionado div.

La pseudoclase :active

Por otro lado, la segunda pseudoclase, :active, permite resaltar los elementos que se encuentran activos, o lo que es lo mismo, elementos que están siendo pulsados en ese instante con el ratón por el usuario:

```
a:active {
  border: 2px solid #FF0000;
  padding: 2px
}
```

Es importante destacar que esta pseudoclase no puede encargarse de detectar pulsación con persistencia, es decir, no guarda el estado de «ha sido pulsado», sino que sólo es capaz de detectar este comportamiento en el momento justo de ser pulsado. Si necesitamos persistencia, tendríamos que hacer uso de Javascript.

Nota: Aunque estas pseudoclases se inventaron para interactuar con un ratón en un sistema de escritorio, pueden llegar a funcionar en dispositivos táctiles. Aún así, ten en cuenta que el :hover no tiene demasiado sentido en el contexto de dispositivos móviles, ya que un usuario no navega por móvil arrastrando el dedo por la pantalla continuamente.

La pseudoclase :focus

Cuando estamos escribiendo en un campo de texto de un formulario de una página web, generalmente pulsamos TAB para cambiar al siguiente campo y SHIFT + TAB para volver al anterior. Cuando estamos posicionados en un elemento, se dice **que ese elemento tiene el foco,** mientras que al pulsar TAB y saltar a otro, solemos decir que pierde el foco. También es posible ganar o perder el foco pulsando con el ratón en un elemento.

El comportamiento de «ganar el foco» puede gestionarse mediante la pseudoclase :focus:

```
/* El campo ha ganado el foco */
input:focus {
  border: 2px dotted #444
}
```

Estas pseudoclases suelen utilizarse con elementos de formularios como <input>, <textarea> o similares, pero también pueden utilizarse con otros elementos, como por ejemplo enlaces <a>.

Esta es una excelente oportunidad para personalizar el estilo de los campos de texto de un formulario (<input> y <textarea>) para que cambien cuando el usuario escribe y se mueve por ellos.

La pseudoclase :focus tiene dos variaciones concretas, las explicaremos a continuación.

La pseudoclase :focus-within

La pseudoclase :focus-within permite darle estilo no sólo al elemento que tiene el foco, sino también a los elementos contenedores relacionados con el elemento que gana el foco.

En este ejemplo, :focus-within permite que cuando uno de los campos <input> del formulario gane el foco, podamos iluminar también el elemento label, que es su contenedor:

```
form :focus-within {
  background: yellow;
}
```

La pseudoclase :focus-visible

En el caso de la pseudoclase :focus-visible es prácticamente idéntico a :focus, solo que podemos aplicar estilos al elemento que gana el foco, pero sólo cuando se ha ganado el foco exclusivamente de forma visible, como por ejemplo, pulsando la tecla TAB y accediendo al elemento.

Esto puede resultar muy útil cuando quieres que el foco coincida con un tema visual para la página.

PSEUDOCLASES DE UBICACIÓN

Existen algunas pseudoclases orientadas a los enlaces o hipervínculos. En este caso, permiten cambiar los estilos dependiendo del comportamiento del enlace. Entre ellas, se encuentran las siguientes:

Pseudoclase	Descripción
:any-link	Selecciona un elemento que es un enlace.
:link	Selecciona un elemento que es un enlace no visitado aún.
:visited	Selecciona un elemento que es un enlace visitado anteriormente.
:target	Selecciona un elemento que coincide con el ancla de la URL actual.

La pseudoclase :any-link

Con la pseudoclase :any-link se puede hacer referencia, como dice su nombre, a elementos que sean cualquier tipo de enlaces.

En este caso, y si no se delimitan de alguna forma**, incluye etiquetas <a> y <area>,** ya que ambas se consideran enlaces.

```
:any-link {
  background: indigo;
  color: white;
  padding: 5px;
}
```

La pseudoclase :link

Por otro lado, la pseudoclase :link permite seleccionar enlaces a páginas que aún no han sido visitadas por el navegador del usuario, lo que puede ser interesante para personalizar el color de este tipo de enlaces. Por defecto, estos enlaces sin visitar suelen ser de color azul.

Veamos un ejemplo donde los cambiamos:

```
a:link {
  color: green;
  font-weight: bold
}
```

La pseudoclase :visited

También tenemos la pseudoclase :visited, que se utiliza para seleccionar y dar estilo a los enlaces que hayan sido visitados previamente en el navegador del usuario. Por defecto, estos enlaces suelen ser de color violeta.

```
a:visited {
  color: purple;
  font-weight: bold;
}
```

La pseudoclase :target

Con la pseudoclase :target podemos seleccionar un elemento HTML donde su **id (ancla)** coincida con el ancla que tenemos actualmente en la URL de navegación.

Es decir, si en nuestra URL tenemos el ancla **#section1, entonces se seleccionará el elemento con id="section1":**

```css
:target {
  background: gold;
  color: #333;
  padding: 5px;
}
```

Ideal para seleccionar cabeceras de secciones a modo de ancla en un documento HTML.

PSEUDOCLASES DE IDIOMA

En CSS podemos encontrarnos con varias pseudoclases relacionadas con el idioma utilizado en la página o en los elementos HTML en cuestión.

Veamos una pequeña tabla de resumen con ellos:

Pseudoclase	Significado	Más información
`:lang(es)`	Selecciona elementos con el idioma español, es decir, atributo `lang="es"`.	
`:dir(value)`	Selecciona elementos con la dirección indicada (`ltr` o `rtl`).	

La pseudoclase :lang()

El atributo HTML lang permite indicar en una etiqueta HTML el idioma en el que está el contenido de sus elementos hijos. De esta forma, un atributo lang="es" indica que el contenido de esa etiqueta se encuentra generalmente en español.

La pseudoclase :lang() acepta por parámetro un idioma (o una lista de ellos separados por comas) para seleccionar el elemento HTML que coincida con uno de ellos:

```
/* Selecciona el elemento que tenga el idioma español */
:lang(es) {
  /* ... */
}

/* Selecciona elementos que estén en español o inglés (*y relacionados*) */
:lang(es, es-*, en, en-*) {
  /* ... */
}
```

Además, como se puede ver en el ejemplo anterior, también se pueden utilizar asteriscos a modo de comodín, lo que lo diferencia considerablemente de seleccionar mediante un selector de atributos [lang="es"], ya que es más potente.

La pseudoclase :dir()

La pseudoclase :dir() permite seleccionar elementos dependiendo de la dirección que tienen establecida.

Generalmente, la direccionalidad del texto se establece con el atributo dir, indicándole el valor ltr (left to right) o el valor rtl (right to left), sin embargo, con **valores como auto** o sin valores indicados, también tendrán una direccionalidad dependiendo del lenguaje establecido en el documento o fragmento.

```
:dir(ltr) {
  border-left: 4px solid darkred;
}

:dir(rtl) {
  border-right: 4px solid darkred;
}
```

Ten en cuenta que la diferencia entre :dir(ltr) y [dir="ltr"] es que en el primer caso se selecciona al elemento tanto si tiene el valor ltr como si no lo tiene, pero lo hereda por contexto.

Sin embargo, con [dir="ltr"] sólo se selecciona si está indicado explícitamente.

PSEUDOCLASES DE ESTRUCTURA

Existe una amplia gama de pseudoclases que permiten seleccionar elementos de un documento HTML según su posición y/o estructura en el documento. Vamos a dividirlas en varias categorías y analizarlas detalladamente.

Elementos raíz

Existen algunas pseudoclases con la que podemos hacer referencia al elemento padre raíz del documento donde estamos trabajando.

En el caso de estar trabajando en un documento HTML, el elemento raíz es <html>, mientras que, en el caso de estar trabajando en un componente, el elemento raíz es el propio componente.

Las pseudoclases que podemos usar en cada caso son las siguientes:

La pseudoclase :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;
}

html {
  background: red;
}
```

En ambos casos, estamos dando un color de fondo al elemento <html>, sin embargo, si especificamos ambos, la pseudoclase :root sale vencedora, y sobreescribe el color de fondo indicado en html.

Aunque no es estrictamente necesario, generalmente, la pseudoclase :root suele utilizarse para establecer variables CSS «globales» que afectan a todo el documento HTML.

El primer y último hijo

A continuación, muestro un pequeño resumen de estas pseudoclases:

Pseudoclase	Descripción
:first-child	Primer elemento hijo (de cualquier tipo).
:last-child	Último elemento hijo (de cualquier tipo).

La pseudoclase :first-child

Con la pseudoclase :first-child podemos seleccionar el primer elemento (o primeros elementos) de un grupo de elementos al mismo nivel.

De esta forma, si indicamos .container :first-child buscará todos los primeros elementos que encuentre:

```
<div class="container">
  <div class="elements">
    <div class="element">Element</div>
    <div class="element">Element</div>
    <div class="element">Element</div>
  </div>
  <div class="elements">
    <div class="element">Element</div>
    <div class="element">Element</div>
    <div class="element">Element</div>
  </div>
</div>
```

En este ejemplo, se seleccionarían 3 elementos:

- El primer con clase .elements.
- El primer con clase .element dentro del primer .elements.
- El primer con clase .element dentro del segundo .elements.

La pseudoclase :last-child

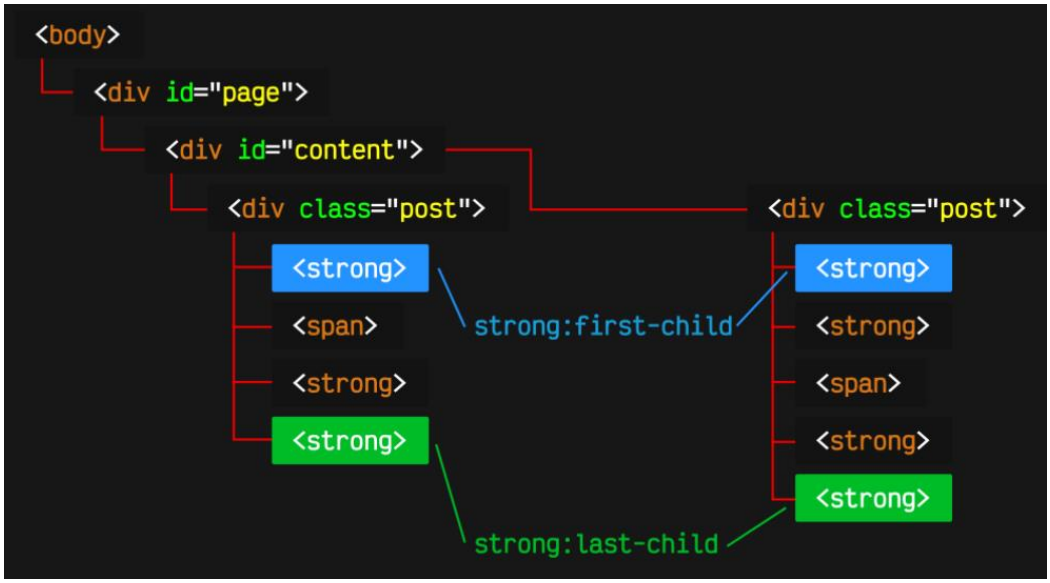
De la misma forma, con la pseudoclase :last-child podemos seleccionar el último elemento (o últimos elementos).

Funciona exactamente igual que :first-child pero haciendo referencia al último elemento en lugar del primero.

```
strong:first-child {
  background-color: cyan;
}

strong:last-child {
  background-color: green;
}
```

Veamos esto con un gráfico en forma de árbol que hará que el ejemplo sea más visual:



Observa que en este caso estamos seleccionando sólo los elementos porque así se lo hemos dicho en el selector del código CSS, donde prefijamos con strong justo antes de :first-child o :last-child. Por lo tanto, solo debe seleccionar el elemento y aplicarle los estilos si además de ser el primero (o último) es un elemento .

Hijos específicos

Sin embargo, con las pseudoclases anteriores sólo podemos seleccionar los primeros y últimos elementos, y podríamos necesitar un elemento específico, como el tercero o el quinto, por ejemplo.

Para ello, podemos utilizar pseudoclases funcionales como **:nth-child() o :nth-last-child().**

Pseudoclase	Descripción
:nth-child(n)	Elemento hijo número n (de cualquier tipo).
:nth-last-child(n)	Elemento hijo número n empezando desde el final (de cualquier tipo).

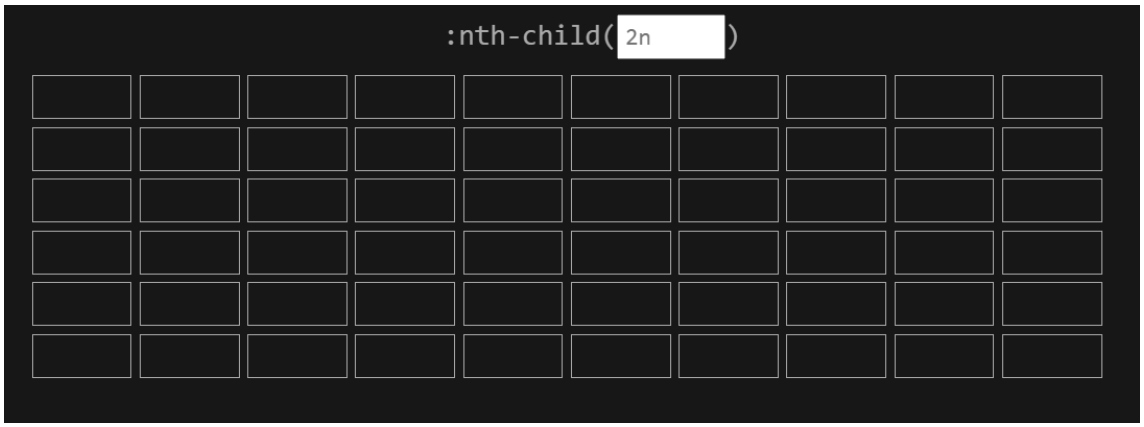
La pseudoclase :nth-child()

La pseudoclase :nth-child(A) permite especificar el elemento hijo deseado, simplemente estableciendo su número en el parámetro A.

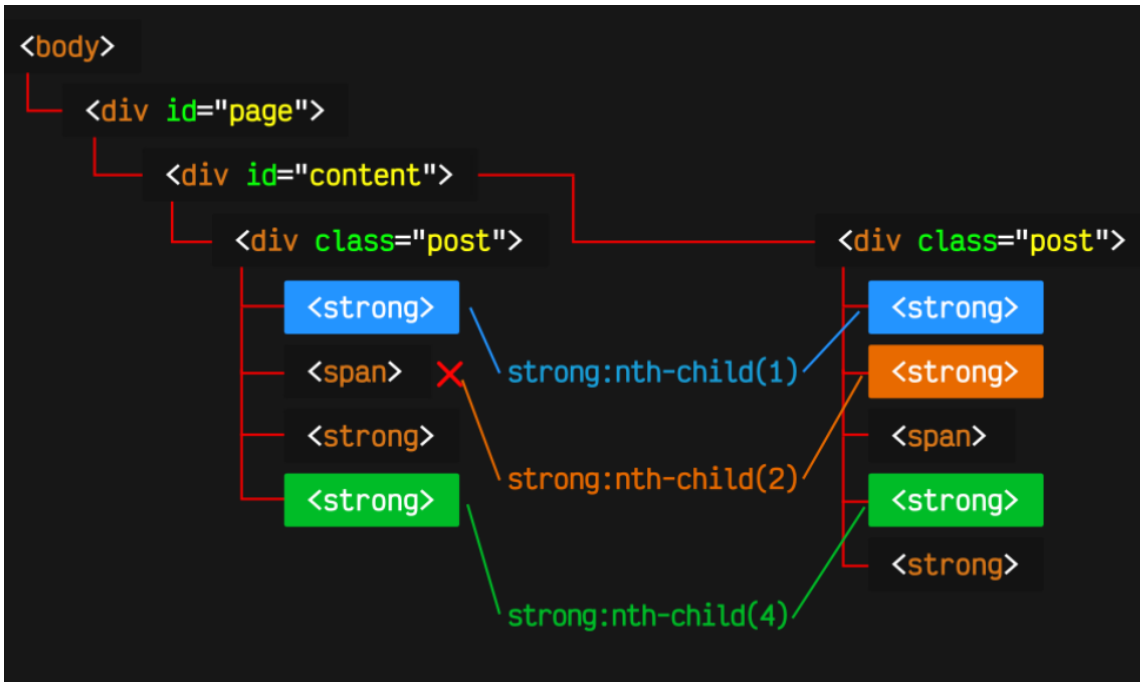
No obstante, hay que tener en cuenta que el parámetro A no es sólo un número, sino que es posible escribir ciertas expresiones:

Selector	Equivalente a...	Significado
strong:nth-child(1)	strong:first-child {}	Primer elemento hijo, que además es un
strong:nth-child(2)		Segundo elemento hijo, que además es un
strong:nth-child(3)		Tercer elemento hijo, que además es un
strong:nth-child(n)	strong	Todos los elementos hijos que son
strong:nth-child(2n)		Todos los elementos hijos pares
strong:nth-child(2n-1)		Todos los elementos hijos impares

A continuación puedes comprobar los valores de la tabla anterior de un modo más visual y práctico, estableciendo en el campo de texto valores como n, n+2, 2n, 2n-1 (o similares) y observando los resultados.



Veamos además un ejemplo gráfico:



Como se aprecia en el ejemplo, en el caso :nth-child(2) se puede ver como el segundo elemento lo ocupa un elemento span, por lo que sólo se selecciona el elemento strong del segundo caso, donde si existe.

La pseudoclase :nth-last-child()

La pseudoclase funcional :nth-last-child(A) funciona de forma muy similar a la anterior, permitiendo también indicarle un parámetro A donde específicar una expresión o número para indicar el hijo concreto.

La diferencia respecto a la anterior, es que comenzamos a contar desde el final, de modo que el elemento número 2, sería el segundo empezando a contar desde el final.

La pseudoclase :nth-child(A of B)

Tanto la pseudoclase :nth-child() como la pseudoclase :nth-last-child() tienen una nueva sintaxis que permiten hacer búsquedas mucho más específicas.

Para ello, en los paréntesis podemos indicar la sintaxis A of B, donde A es un número como en los ejemplos anteriores, pero B es un selector CSS adicional para delimitar más aún los elementos.

Veamos algún ejemplo:

```
<div class="parent">
  <div class="box">Yeah!</div>
  <div class="box special">Yeah!</div>
  <div class="box">Yeah!</div>
  <div class="box special">Yeah!</div>
  <div class="box">Yeah!</div>
  <div class="box special">Yeah!</div>
  <div class="box">Yeah!</div>
</div>
```

```
.parent :nth-child(2 of .special) {
  background: red;
}
```

Yeah!
Yeah!
Yeah!
Yeah!
Yeah!
Yeah!
Yeah!

Observa que en el ejemplo anterior, estamos seleccionando con .parent :nth-child() los hijos del elemento .parent. De esos elementos, hacemos una nueva selección con 2 of .special, con la que nos quedamos con el segundo elemento de los elementos con clase .special.

Hijos (del mismo tipo)

En los casos anteriores, seleccionamos elementos independientemente de que tipo de elemento sea. Simplemente, hacemos caso a la posición donde está ubicado. Y en algún caso, si no coincide la posición con el tipo de elemento especificado en el selector, simplemente no lo selecciona.

Una forma de actuar, quizás, más predecible para nosotros, es que queramos hacer referencia sólo a elementos del mismo tipo, ignorando el resto.

Para ello, utilizaremos los selectores siguientes, análogos a los que ya hemos visto, pero haciendo referencia sólo a elementos del mismo tipo:

Pseudoclase	Descripción
:first-of-type	Primer elemento hijo (de su mismo tipo).
:last-of-type	Último elemento hijo (de su mismo tipo).

La pseudoclase :first-of-type

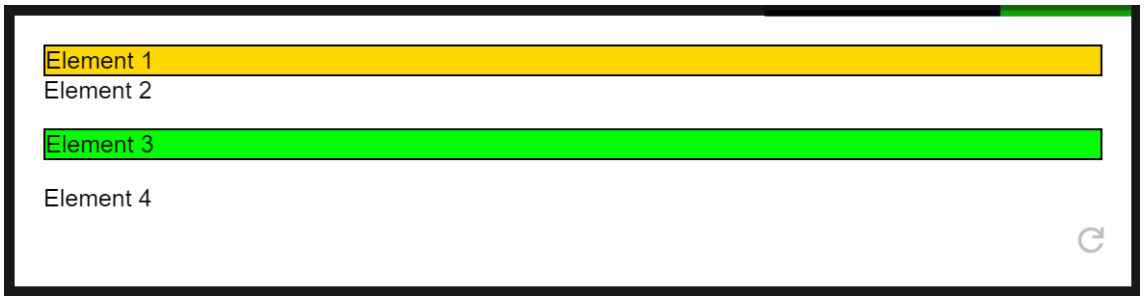
Por ejemplo, la pseudoclase :first-of-type es la análoga a :first-child, sólo que tendrá en cuenta sólo elementos de su mismo tipo. Observa el siguiente ejemplo donde no sólo tenemos , sino que también tenemos un <p>:

```
<div class="container">
  <div class="element">Element 1</div>
  <div class="element">Element 2</div>
  <p class="element">Element 3</p>
  <div class="element">Element 4</div>
</div>
```

```
/* Selecciona "Element 1" */
.container div:first-of-type {
  background: gold;
}

/* Selecciona "Element 3" */
.container p:first-of-type {
  background: lime;
}

/* Selecciona los dos anteriores */
.container :first-of-type {
  border: 2px solid black;
}
```



De esta forma, nos puede resultar mucho más sencillo trabajar con elementos de diferente tipo y darle estilo.

La pseudoclase :last-of-type

De la misma forma, la pseudoclase :last-of-type es la análoga a :last-child, que selecciona el último elemento, pero igual que :first-of-type sólo teniendo en cuenta elementos del mismo tipo.

Hijos específicos (del mismo tipo)

Ahora que estamos en la categoría en la que queremos seleccionar elementos del mismo tipo, también nos puede interesar seleccionar elementos específicos.

Para ello, tenemos también dos pseudoclases análogas a las anteriores:

Pseudoclase	Descripción
:nth-of-type(n)	Elemento hijo número n (de su mismo tipo).
:nth-last-of-type(n)	Elemento hijo número n empezando desde el final (de su mismo tipo).

La pseudoclase :nth-of-type()

La pseudoclase :nth-of-type(A) es la análoga a :nth-child(A). Se trata de una pseudoclase funcional que admite pasar parámetros, donde le podemos indicar un número (o cierta expresión) para ser mucho más específicos a la hora de seleccionar elementos del mismo tipo.

```
<div class="container">
  <div class="element">Element 1</div>
  <div class="element">Element 2</div>
  <p class="element">Element 3</p>
  <div class="element">Element 4</div>
</div>
```

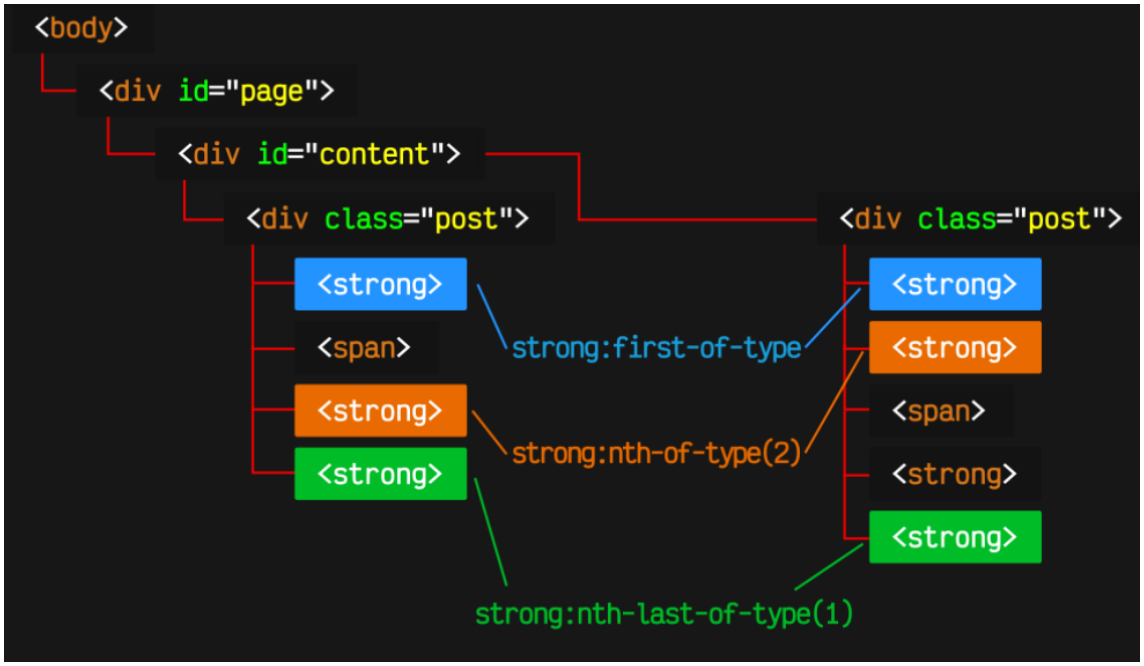
```css
/* Seleccionamos sólo el "Element 2", ya que no hay un segundo <p> */
.container :nth-of-type(2) {
  background: gold;
}
```

Element 1
Element 2
Element 3
Element 4

La pseudoclase :nth-last-of-type()

La pseudoclase :nth-last-of-type(A) es la análoga a :nth-last-child(A).

Veamos un nuevo ejemplo sobre el ejercicio anterior, utilizando ahora estas últimas pseudoclases que hemos visto:



En este gráfico, se puede ver como strong:nth-of-type(2) selecciona el segundo elemento strong en ambos casos, a pesar de que en el primero ocupa la tercera posición. En este caso se selecciona porque es el segundo elemento de su mismo tipo ().

Por otro lado, strong:nth-last-of-type(1) hace una selección de forma inversa, empezando por el último elemento, por lo que elige el último elemento.

Elementos únicos o sin hijos

Existen también varias pseudoclases para la gestión de hijos únicos. Son las siguientes:

Pseudoclase	Descripción
:only-child	Elemento que es hijo único (de cualquier tipo).
:only-of-type	Elemento que es hijo único (de su mismo tipo).
:empty	Elemento vacío (sin hijos, ni texto).

La pseudoclase :only-child

La propiedad :only-child nos proporciona una forma de seleccionar los elementos que sean el único hijo de su elemento padre.

Por lo tanto, si un contenedor tiene en su interior un sólo elemento hijo, podremos seleccionar y aplicar estilos.

```
.container :only-child {
  /* Selecciona el hijo de un padre que sólo tiene un elemento hijo */
}
```

La pseudoclase :only-of-type

Además, como ha ocurrido anteriormente, también existe la pseudoclase :only-of-type que es la análoga a la anterior, pero sólo para elementos del mismo tipo.

En este caso, podríamos tener un contenedor que contiene varios elementos, pero todos son únicos en su tipo, por lo tanto, podrían ser seleccionados.

```
<div class="container">
  <strong>Hi</strong>
  <p>Ho</p>
</div>

<div class="container">
  <strong>Hi</strong>
</div>
```

```css
/* Selecciona los hijos de un padre que sólo tiene un elemento hijo de su mismo tipo
/* En este caso, seleccionaría todos */
.container :only-of-type {
  background: gold;
}
```

La pseudoclase :empty

Por último, la pseudoclase :empty permite seleccionar los elementos que estén vacíos. Observa los siguientes ejemplos, que podríamos considerar que están vacíos, aunque con ciertos matices:

```html
<div class="empty container"></div>

<div class="empty ghost container">

</div>

<div class="empty comment container"><!-- lala --></div>
```

```css
div {
  background: grey;
  width: 100px;
  height: 50px;
  margin: 10px;
}

:empty {
  border: 3px solid red;
}
```

Ojo con esto, ya que el navegador no considerará vacío el elemento que contiene espacios en blanco (segundo ejemplo).

El primer elemento, efectivamente está vacío.

El segundo elemento .ghost, puede parecer vacío pero tiene espacios en blanco.

El tercer elemento .comment, está vacío sólo que contiene un comentario HTML.

PSEUDOCLASES DE FORMULARIOS

Existe una serie de pseudoclases en CSS que pueden ser utilizadas para usar en formularios de una página web.

Estas pseudoclases permiten seleccionar elementos para darle estilo dependiendo de temas relacionados con los formularios o los campos que están en el interior de un formulario.

Esencialmente, existen tres categorías de Pseudoclases:

- **De interacción en formularios**: Seleccionar elementos cuando cambia el estado de un elemento.
- **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.

Veamos cada una de ellas y las pseudoclases que las componen.

Pseudoclases de interacción

Las siguientes pseudoclases están orientadas a un estado específico de ciertos campos de un formulario que el usuario puede modificar. Estos campos pueden ser los siguientes elementos:

Elementos de tipo radio <input type="radio">

Elementos de casilla de verificación <input type="checkbox">

Las pseudoclases que podemos utilizar son las siguientes:

Pseudoclase	Descripción
:checked	Selecciona el elemento cuando el campo está seleccionado.
:indeterminate	Selecciona el elemento cuando la casilla está en un estado indeterminado.

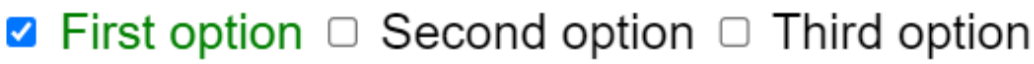
La pseudoclase :checked

La pseudoclase :checked permite seleccionar un elemento que ha sido marcado o seleccionado y entonces aplicarle estilo CSS.

Por ejemplo, se podría utilizar el siguiente fragmento de código:

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

```
<input type="checkbox" checked> <span>First option</span>
<input type="checkbox"> <span>Second option</span>
<input type="checkbox"> <span>Third option</span>
```



Este ejemplo está **utilizando el combinador hermano adyacente + para darle formato al ** que se encuentra a continuación de la casilla <input> seleccionada, ya sea de tipo radio o de tipo checkbox. De esta forma, los textos que acompañan al campo del formulario que hayan sido seleccionados, se mostrarán en verde.

Como detalle adicional, también pueden seleccionar los elementos de opción <option> de una lista seleccionable <select> que se encuentren seleccionados.

La pseudoclase :indeterminate

La pseudoclase :indeterminate se utiliza para **seleccionar elementos que tienen un estado indeterminado** donde no se sabe exactamente su estado**. Hay tres situaciones donde esto puede ocurrir:**

Un <input type="checkbox"> donde mediante Javascript se marca la propiedad .indeterminate a true.

Varios <input type="radio"> con el mismo atributo name y sin definir el atributo checked en ninguno de ellos.

Un elemento <progress> donde no se define su valor y generalmente se muestra moviéndose de un lado a otro.

Un ejemplo en acción:

```
:indeterminate + span {
  background: black;
  color: white;
  padding: 5px;
}
```

```
<input type="checkbox"> <span>First option (Indeterminated)</span>
<input type="checkbox"> <span>Second option</span>
<input type="checkbox"> <span>Third option</span>
```

```
const inputs = document.querySelectorAll("input");
inputs[0].indeterminate = true;
```

En este caso, de forma similar al anterior, cambiamos el texto que está dentro de un elemento a continuación de uno de los elementos anteriores de formulario en estado indeterminado.

Pseudoclases de estado

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:

Pseudoclase	Descripción
:enabled	Selecciona cuando el campo del formulario está activado.
:disabled	Selecciona cuando el campo del formulario está desactivado.
:read-only	Selecciona cuando el campo es de sólo lectura.
:read-write	Selecciona cuando el campo es editable por el usuario.
:placeholder-shown	Selecciona cuando el campo está mostrando un placeholder.
:default	Selecciona cuando el elemento tiene el valor por defecto.

La pseudoclase :enabled

Utilizando la autoexplicativa pseudoclase :enabled, podemos seleccionar elementos que se encuentren activados (comportamiento por defecto):

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

```
<button>Botón activado</button>
<button disabled>Botón desactivado</button>
```

En este caso restringimos a los elementos <input> que se encuentren desactivados.

La pseudoclase :disabled

Sin embargo, lo más interesante de este tema viene al poder darle estilo a elementos desactivados con la **pseudoclase :disabled,** donde se seleccionan los elementos a los que se le ha añadido el atributo disabled:

```
button:disabled {
  background-color: grey;
  cursor: not-allowed;
}
```

```
<button>Botón activado</button>
<button disabled>Botón desactivado</button>
```

Al margen de tener la funcionalidad desactivada, también podremos darle estilos visuales que lo informen.

La pseudoclase :read-only

La pseudoclase :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.

Lo que tienen en común es que ambas están bloqueadas y no permiten modificar su valor, por lo que se suelen percibir como algo equivalente.

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

Importante: Ten en cuenta que :read-only aplicará los estilos a todos los elementos HTML que no puedan ser modificados por el usuario. Incluso un o un <p> lo identificará como un elemento de solo lectura, siempre y cuando no lleven el atributo contenteditable.

La pseudoclase :read-write

Por otro lado, la pseudoclase :read-write es muy útil para dar estilos a todos aquellos elementos que son editables por el usuario, sean campos de texto <input> o <textarea>.

```
input:read-write {
  background-color: green;
  color: white
}
```

De esta forma, todos los elementos <input> que sean de lectura y escritura (editables) se seleccionarán y será posible aplicarles estilo. **Recuerda que la pseudoclase read-write también da estilos a elementos HTML que contengan el atributo contenteditable**, como por ejemplo un párrafo editable por el usuario con dicho atributo:

```
<p contenteditable>Mensaje editable</p>
```

```
:read-write {
  background-color: cyan;
  color: black;
  padding: 5px;
}
```

Mensaje editable

La pseudoclase :placeholder-shown

Con la pseudoclase :placeholder-shown **se nos permite seleccionar y dar estilo a los elementos que están actualmente mostrando un placeholder**. Si no lo conoces, el término placeholder es un texto o imagen de muestra que se suele colocar para que el usuario conozca un ejemplo de la información que debe ir en esa zona.

En nuestro caso, se puede hacer con el atributo placeholder en elementos <input>:

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

```html
<input type="text" placeholder="usuario@gmail.com">
```

Recuerda que en los placeholder no se debe colocar texto para indicar que campo es (nick, email...), sino una sugerencia de lo que podrías escribir (prueba, prueba@email.com...). Para el primer caso es más apropiado utilizar la etiqueta <label>.

La pseudoclase :default

Con la pseudoclase :default podemos seleccionar los elementos de un formulario que se consideran que tienen, de alguna forma, un valor por defecto. Es decir:

- Elementos <input type="checkbox"> o <input type="radio"> que tienen el atributo checked.
- Elementos <selected> donde una de sus opciones tiene el atributo selected.
- Elementos <button> o <input type="submit"> que son el botón por defecto del <form>.

Los elementos que coincidan con uno de estos casos, se podrían seleccionar con el siguiente código:

```css
:default {
  background: red;
  color: white;
  border: 3px solid red;
  accent-color: red;
}
```

Pseudoclases de validación

En HTML5 es posible dotar de **capacidades de validación a los campos de un formulario, pudiendo interactuar con ellos 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.

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

Pseudoclase	¿Cuándo aplica estilos?
`:required`	Cuando el campo es obligatorio, o sea, tiene el atributo `required`.
`:optional`	Cuando el campo es opcional (por defecto, todos los campos).
`:valid`	Cuando los campos cumplen la validación HTML5.
`:invalid`	Cuando los campos no cumplen la validación HTML5.
`:user-valid`	Idem a `:valid`, pero cuando el usuario ha interactuado.
`:user-invalid`	Idem a `:invalid`, pero cuando el usuario ha interactuado.
`:in-range`	Cuando los campos numéricos están dentro del rango.
`:out-of-range`	Cuando los campos numéricos están fuera del rango.

La pseudoclase :required

En un formulario HTML es posible establecer un campo obligatorio que será necesario rellenar para enviarlo. 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.

Por lo general, los campos de un formulario son siempre opcionales. **Para hacer obligatorio un campo, tenemos que indicar en el elemento HTML el atributo required, al cuál será posible darle estilo mediante la pseudoclase :required:**

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

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

Nombre: Apellidos: Nickname:

De esta forma, todos los campos <input> obligatorios aparecerán con un borde rojo.

La pseudoclase :optional

Por otra parte, los campos opcionales son todos aquellos que no tienen el atributo required. Pueden seleccionarse con la pseudoclase :optional:

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

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

Nombre: Apellidos: Nickname:

La pseudoclase :valid

Las validaciones en formularios HTML siempre han sido un proceso tedioso, al menos hasta la llegada de HTML5. Actualmente, desde HTML5 se brinda un excelente soporte de validaciones desde el lado del cliente sin necesidad de Javascript, 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.

Importante: Ten en cuenta que la validación de cliente es apropiada solo para reducir la latencia de envío/recepción al servidor, pero nunca como estrategia para evitar problemas de seguridad o similares, para la cuál SIEMPRE se debe tener validación en el servidor. Las validaciones utilizadas en frontend, es posible falsearlas o saltárselas.

Para validar la información del campo de formulario utilizamos el atributo pattern y un mecanismo para detectar **coincidencias llamado expresiones regulares.**

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="text" 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.

Además, una vez establecida la validación mediante el atributo pattern, ahora, mediante la pseudoclase :valid podemos aplicar un estilo cuando el texto escrito en el campo pase correctamente la validación:

```
<input type="text" name="dni" required
       pattern="(([X-Z]{1}-?[0-9]{7})|([0-9]{8}))[A-Z]{1}" />
```

```
input:invalid {
  background: red;
}

input:valid {
  background-color: green;
}

input {
  color: white;
}
```

La pseudoclase :invalid

De forma opuesta, se pueden seleccionar elementos y aplicar ciertos estilos si no se cumple el patrón de validación, utilizando la pseudoclase :invalid:

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

La pseudoclase :user-valid

La pseudoclase :user-valid es una versión particular de :valid, donde el elemento se selecciona si, además de comprobar que la validación es correcta y se cumple, el usuario ha interactuado con anterioridad con el campo en cuestión.

La pseudoclase :user-invalid

De la misma forma, :user-invalid es una versión particular correspondiente a la pseudoclase :invalid. Mientras que :invalid permite seleccionar los elementos que no cumplen la restricción de la validación, :user-invalid permite seleccionar los elementos que no cumplen la validación pero que además el usuario ha interactuado anteriormente con ellos.

La pseudoclase :in-range

Pero ten en cuenta que en una validación numérica, donde un usuario podría escribir 500 en el campo de edad, sería un resultado que no nos gustaría aceptar. En el patrón de validación del atributo pattern indicamos «una o más cifras del 0 al 9», pero no establecemos unos límites.

Lo ideal sería establecer un rango, algo que se suele hacer muy a menudo si tenemos campos numéricos de formulario mediante los atributos min y max:

```
<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. De esta forma, si escribimos la pseudoclase :in-range podremos seleccionar los elementos que indican un valor y cumplen la validación:

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

La pseudoclase :out-of-range

La pseudoclase :out-of-range, por otro lado, permite seleccionar y dar estilo a los elementos que tienen valores fuera del rango definido, y por lo tanto, no son válidos.

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

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

PSEUDOCLASES DE ESTADO

Existen una serie de pseudoclases para comprobar el estado visual de un elemento que se considera modal, es decir, que «centran» la interacción del usuario en un elemento principal (y sus hijos) y no permiten la interacción con otros elementos hasta que se cierre ese elemento principal.

Estas pseudoclases son las siguientes:

Pseudoclase	Descripción
:fullscreen	Selecciona si la página está en modo de pantalla completa.
:modal	Selecciona el elemento que es de tipo modal.

La pseudoclase :fullscreen

Mediante la pseudoclase :fullscreen podemos seleccionar elementos que se encuentren en modo pantalla completa, lo que habitualmente se realiza **mediante la API FullScreen de Javascript.**

```html
<div class="screen" open>
  <p>Has colocado esta ventana en pantalla completa.</p>
  <button class="close">Cerrar</button>
</div>

<button class="open">Abrir en FullScreen</button>
```

```js
const button = document.querySelector(".open");
const close = document.querySelector(".close");
const screen = document.querySelector(".screen");

button.addEventListener("click", () => screen.requestFullscreen());
close.addEventListener("click", () => document.exitFullscreen());
```

```css
.screen {
  display: none;
}

:fullscreen {
  display: block;
  background: linear-gradient(120deg, black, indigo);
  border: 3px solid red;
  color: white;
  padding: 1rem;
}
```

Abrir en FullScreen

La pseudoclase :modal

Mediante la pseudoclase :modal se puede seleccionar un elemento que está actuando como una ventana o elemento modal, o lo que es lo mismo, un elemento que centra la atención del usuario y no permite interacción con otros elementos que no son sus hijos.

En el ejemplo anterior, modificando :fullscreen por :modal continuaría siendo válido y funcionando porque **un elemento a pantalla completa con .requestFullscreen() también es un elemento modal, ya que no permite interacción con otros elementos fuera de él hasta que se cierre el modo pantalla completa.**

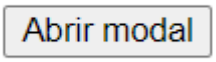
```html
<dialog class="screen">
  <p>Has abierto este diálogo.</p>
  <button class="close">Cerrar</button>
</dialog>

<button class="open">Abrir modal</button>
```

```js
const button = document.querySelector(".open");
const close = document.querySelector(".close");
const modal = document.querySelector(".screen");

button.addEventListener("click", () => modal.showModal());
close.addEventListener("click", () => modal.close());
```

```css
:modal {
  display: block;
  background: linear-gradient(120deg, black, indigo);
  border: 3px solid red;
  color: white;
  padding: 1rem;
}
```



También ten en cuenta que existe la posibilidad de que múltiples elementos sean modales, aunque sólo uno de ellos sea el que se encuentre activo y capaz de recibir interacciones.

En este ejemplo, hemos utilizado una etiqueta <dialog> para crear ventanas de diálogo, en este caso, una ventana modal.

BIBLIOWEB: Información extraída de CSS3 The journing of Centering