

CSS

RESET

Cada navegador tiene por defecto sus propios estilos que aplica a las principales etiquetas de css. Por ejemplo: la mayoría de navegadores da márgenes a la etiqueta `<body>`, o le da un determinado tamaño y margen a la etiqueta `<h1>`.

Para eliminar estos estilos y que no interfieran con nuestras clases se suelen usar unas hojas de estilos que anulan o “resetean” estos estilos predefinidos.

```
11 .wrapper {
12   border: 1px solid #999;
13   border-radius: 5px;
14   color: #222;
15   padding: 10px;
16   background: #ddd;
17 }
18
19 .element-1 {
20   color: #fff;
21   font-size: 24px;
22   line-height: 32px;
23   background: #569dc5;
24   padding: 25px 30px 20px 30px;
25   font-weight: 400;
26 }
```



Diversos desarrolladores han creado archivos de reset para CSS.

Resets de CSS Más Populares:

Sistema de reseteo	Descripción
Reset CSS	El reset de Eric Meyer , históricamente, quizás el primero.
Normalize.css	Uno de los reset más tradicionales y populares.
HTML5 Reset Stylesheet	Esta hoja de estilo de Richard Clark, es una modificación de la hoja de estilo de reset de CSS de Eric Meyer, orientada a los elementos HTML 5 modernos.
A CSS Reset for 2022	CSS Reset 2022 de Mayank99
Reset CSS Pro	Reset CSS de Eduardo Fierro.

CSS Remedy	CSS Remedy, de Jen Simmons
Preflight	Reset CSS de Tailwind CSS
Josh W. Comeau	Un reset moderno, explicado paso a paso.
Andy Bell	Un reset moderno de Andy Bell.
2024 CSS Reset	Simple Reset CSS

Utilizar un sistema de **reset CSS** es decisión del desarrollador y no es algo obligatorio.

Podemos incorporar estos archivos de reset a nuestro proyecto mediante un link al archivo externo.

El link al archivo CSS de reset debe estar dentro del head del html y debe ser la primera hoja de estilos que se ha de leer. Los estilos se leen en forma secuencial, por tanto, si estuviera en otra posición, podría eliminar los estilos definidos por las hojas de estilos anteriores.

Ejemplo de link de importación del archivo reset css de Meyer:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Selectores CSS</title>
  <!-- FAVICON -->
  <link rel="shortcut icon" href="assets/img/css-icone.svg" type="image/x-icon" />

  <!-- HOJAS DE ESTILO -->
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/meyer-reset/2.0/reset.min.css"/>
  <link rel="stylesheet" href="assets/css/style.css" />
</style>
</head>
<body>
```

`<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/meyer-reset/2.0/reset.min.css" />`

El archivo con extensión **“.min”**, es una versión comprimida que ocupa menos espacio, pues se han eliminado espacios innecesarios, sin perder la funcionalidad.

También podemos crear nuestra propia hoja de estilos, reseteando solo las propiedades que nos interese y guardarla en nuestro proyecto. El siguiente podría ser un ejemplo sencillo, que podría estar al inicio de nuestro archivo style.css.

```
* {  
  box-sizing: border-box;  
  margin: 0;  
  padding: 0;  
  font-family: "Gill Sans", sans-serif;  
}
```

Recuerda que en el caso de que el código de reset se escriba en el archivo style.css siempre debe ir al comienzo del archivo.

Es posible descargar el archivo de reset css e incorporarlo a nuestro proyecto o bien poner el link al archivo externo.

Para encontrar el link al archivo reset, podemos visitar la web del creador y buscar en ella el link al archivo o bien usar un buscador CDN (Content Distribution Net o red de distribución de contenido) para encontrar el link.

Encontrar link con actualizaciones:

<https://cdnjs.com/>

<https://cdnjs.com/libraries/>

Links a archivos css reset más utilizados:

<https://meyerweb.com/eric/tools/css/reset/>

<https://necolas.github.io/normalize.css/>

<http://html5doctor.com/html-5-reset-stylesheet/>

*en la tabla anterior, los nombres a la izquierda tienen link a cada web correspondiente.

Para HTML5/CSS3, se aconseja el uso de Normalize.

Referencias:

<https://3con14.biz/css/conceptos/3-los-reset-de-css.html>

<https://lenguajecss.com/cascada-css/herencia/reset-css/>

especificidad inline, interno, externo- NO PROCEDE

<https://lenguajecss.com/cascada-css/herencia/reset-css/>

PSEUDO-ELEMENTOS

Se utiliza un pseudo-elemento CSS para dar estilos a partes específicas de un elemento.

Al igual que las [Pseudo-classes](#), los pseudo-elementos se añaden a los selectores, pero en cambio, no describen un estado especial, sino que, permiten añadir estilos a una parte concreta del documento. Por ejemplo, el pseudo elemento `::first-line` selecciona solo la primera línea del elemento especificado por el selector.

```
selector #id .clase [atributo] :pseudoclase ::pseudoelemento {  
    propiedad : valor ;  
    propiedad : valor  
}
```

La sintaxis con `::` no surgió desde los inicios de CSS, por lo que aún hoy en día es posible encontrar fragmentos de código desactualizados que utilizan pseudoelementos como `::before` o `::after` con la sintaxis de pseudoclase: `:before` y `:after`.

Sintaxis simplificada

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

Muchos navegadores también admiten la sintaxis `selector:pseudo-elemento { propiedad: valor; }`, pero no se recomienda este uso

Tipos de pseudo elementos

Existen varios tipos de pseudo elementos, que se encuentran organizados en categorías. Puedes encontrarlos en la siguiente tabla:

Pseudoelementos	Significado	Más información
Contenido generado	Información generada desde CSS, sin existir en el HTML.	Ver Contenido generado
<code>::before</code> , <code>::after</code>		
Contenido tipográfico	Pseudo elementos relacionados con temas de tipografías.	Ver Tipográfico
<code>::first-line</code> , <code>::first-letter</code>		
Contenido destacado	Pseudo elementos para remarcar o destacar información.	Ver Resaltado
<code>::selection</code> , <code>::target-text</code> , <code>::spelling-error</code> , <code>::grammar-error</code>		
WebComponents	Pseudo elementos relacionados	Ver Slots y Parts

	con WebComponents	
::part, ::slotted		
View Transition API	Pseudo elementos de transición de cambio de página.	Ver View Transitions
::view-transition, ::view-transition-group, ::view-transition-image-pair ::view-transition-new, ::view-transition-old		
Otros pseudoelementos	Pseudo elementos de otras categorías variadas	(Ver más abajo)
::marker, ::placeholder, ::file-selector-button		

Otros pseudo elementos

Al margen de los pseudo elementos anteriores, explicados en su respectiva sección, nos quedan algunos pseudo elementos sin catalogar.

Vamos a repasarlos:

Pseudoelemento	Descripción
::marker	Aplica estilos a las marcas o símbolos de cada ítem de una lista.
::backdrop	Aplica estilos al fondo exterior de un elemento en primer plano (sin que afecte a este).
::placeholder	Aplica estilos a los textos de sugerencia de los campos <input>.
::file-selector-button	Aplica estilos a los botones de campo <input> de subir archivos.

ALGUNOS EJEMPLOS:

---1

::first-line aplica estilos a la primera línea de un párrafo.

Ejemplo de código:

CSS:

```
p::first-line {
  color: #ff0000;
  font-variant: small-caps;
}
```

Html:

```
<p>You can use the ::first-line pseudo-element to add a
special effect to the first line of a text. Some more text.
And even more, and more, and more, and more, and more, and
more, and more, and more, and more, and more, and more, and
more.</p>
```

Resultado:

YOU CAN USE THE ::FIRST-LINE PSEUDO-ELEMENT TO ADD A SPECIAL EFFECT to the first line of a text. Some more text. And even more, and more, and more, and more, and more, and more, and more, and more, and more, and more, and more, and more.

Hay toda una serie de propiedades que podremos aplicar:

- font properties
- color properties
- background properties
- word-spacing
- letter-spacing
- text-decoration
- vertical-align
- text-transform
- line-height
- clear

https://www.w3schools.com/css/tryit.asp?filename=trycss_firstline

---2

::first-letter aplica estilos a la primera letra de un párrafo.

Ejemplo de código:

CSS:

```
p::first-letter {  
  color: #ff0000;  
  font-size: xx-large;  
}
```

Html:

```
<p>You can use the ::first-letter pseudo-element to add a  
special effect to the first character of a text!</p>
```

Resultado:

You can use the ::first-letter pseudo-element to add a special effect to the first character of a text!

Estas son las propiedades que podremos aplicar:

- font properties
- color properties
- background properties
- margin properties
- padding properties
- border properties
- text-decoration
- vertical-align (only if "float" is "none")
- text-transform
- line-height
- float
- clear

https://www.w3schools.com/css/tryit.asp?filename=trycss_firstletter

---3

Pseudo elementos y clases HTML

Los pseudo elementos se pueden combinar con clases de HTML.

Ejemplo de código:

CSS:

```
p.intro::first-letter {  
  color: #ff0000;  
  font-size: 200%;  
}
```

Html:

```
<p class="intro">This is an introduction.</p>  
<p>This is a paragraph with some text. A bit more text  
even.</p>
```

Resultado:

This is an introduction.

This is a paragraph with some text. A bit more text even.

https://www.w3schools.com/css/tryit.asp?filename=trycss_pseudo-element

---4

Multiples pseudo elements

Se pueden combinar varios pseudo elementos.

Ejemplo de código:

CSS:

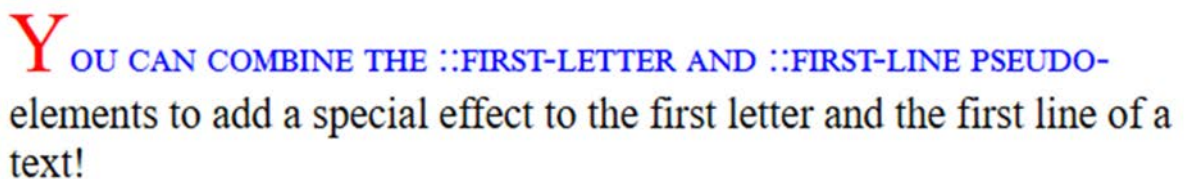
```
p::first-letter {  
  color: #ff0000;  
  font-size: xx-large;  
}
```

```
p::first-line {  
  color: #0000ff;  
  font-variant: small-caps;  
}
```

Html:

```
<p>You can combine the ::first-letter and ::first-line pseudo-  
elements to add a special effect to the first letter and the  
first line of a text!</p>
```

Resultado:



YOU CAN COMBINE THE ::FIRST-LETTER AND ::FIRST-LINE PSEUDO-
elements to add a special effect to the first letter and the first line of a
text!

https://www.w3schools.com/css/tryit.asp?filename=trycss_firstline_letter

---5

`::before` insertar algún contenido antes de un elemento.

Ejemplo de código:

CSS:

```
h1::before {  
  content: url(smiley.gif);  
}
```

Html:

```
<h1>This is a heading</h1>
```

```
<p>The ::before pseudo-element inserts content before the  
content of an element.</p>
```

```
<h1>This is a heading</h1>
```


Resultado:



https://www.w3schools.com/css/tryit.asp?filename=trycss_before

---6

::after insertar algún contenido despues de un elemento.

Ejemplo de código:

CSS:

```
h1::after {  
  content: url(smiley.gif);  
}
```

Html:

```
<h1>This is a heading</h1>
```

```
<p>The ::after pseudo-element inserts content after the  
content of an element.</p>
```

```
<h1>This is a heading</h1>
```

Resultado:



https://www.w3schools.com/css/tryit.asp?filename=trycss_after

Otros posibles contenidos en el atributo "content" con los pseudo elementos ::after y ::before

El atributo `content` en los pseudo-elementos `::before` y `::after` admite varios tipos de contenido. Los principales tipos de valores que se pueden usar y cómo se ven en la práctica.

1. Texto

Puedes agregar texto directamente entre comillas.

```
css
.elemento::before {
  content: "Inicio: ";
  color: blue;
}
html
<p class="elemento">Página web</p>
```

- **Resultado:** Aparece la palabra "Inicio: " antes del contenido de `<p>`.

2. Cadenas vacías

Se pueden usar cadenas vacías (`" "`), que a menudo sirven para crear efectos visuales, como separar o simular un espacio vacío.

```
css
.elemento::before {
  content: " ";
  display: block;
  width: 100%;
  height: 2px;
  background-color: gray;
  margin-bottom: 5px;
}
```

3. URLs (imágenes)

Puedes usar URLs para mostrar imágenes, aunque esto se usa con menos frecuencia.

```
css
.elemento::before {
  content: url('icono.png');
  display: inline-block;
  margin-right: 5px;
}
```

4. Atributos del elemento (`attr()`)

Permite insertar el valor de un atributo del elemento, como `title`, `data-*`, etc.

css

```
.elemento::after {  
  content: " - " attr(data-info);  
  color: gray;  
}
```

html

```
<p class="elemento" data-info="más detalles">Descripción</p>
```

- **Resultado:** "Descripción - más detalles".

5. Contadores (`counter()`)

Si usas `counter-reset` y `counter-increment`, puedes añadir contadores que cambian automáticamente.

css

```
.counter {  
  counter-reset: section;  
}  
  
.counter p::before {  
  counter-increment: section;  
  content: "Sección " counter(section) ": ";  
}
```

html

```
<div class="counter">  
  <p>Primera sección</p>  
  <p>Segunda sección</p>  
</div>
```

- **Resultado:** Cada `<p>` tendrá un número antes, como "Sección 1: Primera sección".

Resumen de los valores admitidos en `content`:

- **Texto** — cualquier cadena de texto.
- **Cadenas vacías** — `" "`, para efectos de estilo.
- **URLs** — para imágenes.
- **Atributos (`attr()`)** — insertar el valor de un atributo.
- **Contadores (`counter()`)** — para numeración automática.

Este atributo `content` permite crear efectos visuales y añadir información útil sin alterar el HTML.

Dimensionar imagen introducida con ::before y ::after

Para darle dimensiones a una imagen introducida con `::before` en CSS, lo primero es recordar que `::before` no inserta imágenes directamente, sino contenido textual o generado (como iconos en formato de texto o imágenes en formato base64 o como fondo de un elemento). Para mostrar una imagen, puedes usar `content` con una URL y luego ajustar las dimensiones con propiedades de `width` y `height`.

CSS

```
.elemento::before {
  content: url('ruta-de-tu-imagen.jpg'); /* Inserta la imagen */
  display: inline-block; /* Permite aplicar ancho y alto */
  width: 100px; /* Define el ancho */
  height: 100px; /* Define la altura */
  /* Ajusta si es necesario */
}
```

Otras Opciones

1. **Usar `background-image` en lugar de `content`:** Si necesitas más control (como posicionar la imagen de fondo), puedes usar `background-image`:

CSS

```
.elemento::before {
  content: ''; /* Necesario para activar ::before */
  display: inline-block; /* Necesario para ajustar dimensiones */
  background-image: url('ruta-de-tu-imagen.jpg');
  background-size: cover; /* Asegura que la imagen cubra todo el
  área */
  width: 100px;
  height: 100px;
}
```

2. **Ajustar el tamaño de una imagen SVG embebida en `content`:** Puedes insertar un SVG directamente en `content`, lo cual es una forma avanzada y funciona bien para iconos:

CSS

```
.elemento::before {
  content: url('data:image/svg+xml;utf8,<svg...></svg>');
  display: inline-block;
  width: 50px;
  height: 50px;
}
```

Aquí un ejemplo más extenso del uso de `::after` y `::before`

<https://lenguajecss.com/css/pseudoelementos/after-before/>

---7

`::marker` permite dar estilos a marcadores de elementos de listas. Funciona en cualquier elemento o pseudo elemento configurado para `display: list-item`, como `` o `<summary>`.

`::marker` puede aplicarse a un elemento, pero si no se especifica elemento aplica a todo el documento (`li::marker` o `::marker`).

*El elemento `<summary>` especifica un resumen, un título o una leyenda para un contenido. Al hacer clic en el elemento, se alterna entre el estado abierto y cerrado del contenido. También `<details>` actúa como un "toggle" de su contenido. `::marker` no afecta al estilo del icono de details.

Ejemplo de código:

CSS:

```
::marker {  
  color: red;  
  font-size: 23px;  
}
```

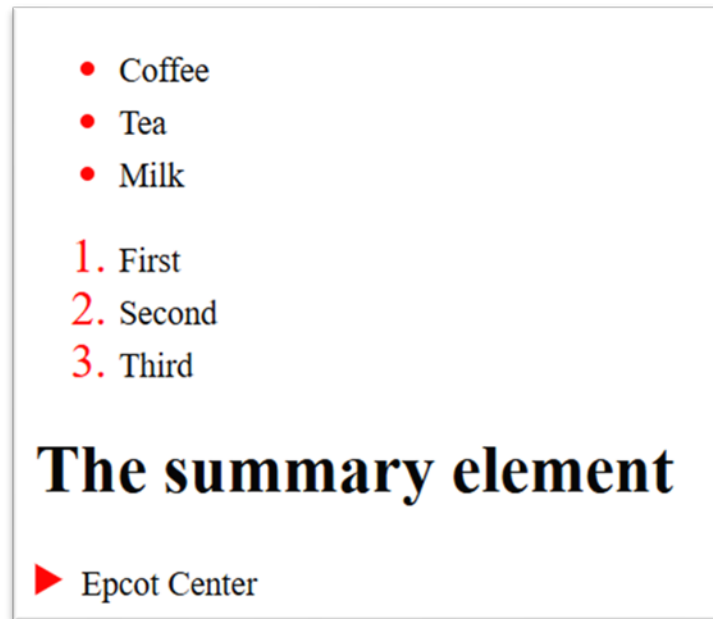
Html:

```
<ul>  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ul>  
  
<ol>  
  <li>First</li>  
  <li>Second</li>  
  <li>Third</li>  
</ol>  
  
<h1>The summary element</h1>  
  
<details>  
  
<summary>Epcot Center</summary>
```

`<p>Epcot is a theme park at Walt Disney World Resort featuring exciting attractions, international pavilions, award-winning fireworks and seasonal special events.</p>`

`</details>`

Resultado:



https://www.w3schools.com/css/tryit.asp?filename=trycss_marker
https://www.w3schools.com/tags/tryit.asp?filename=tryhtml5_summary

---8

`::selection` aplica estilo al elemento seleccionado por el usuario.

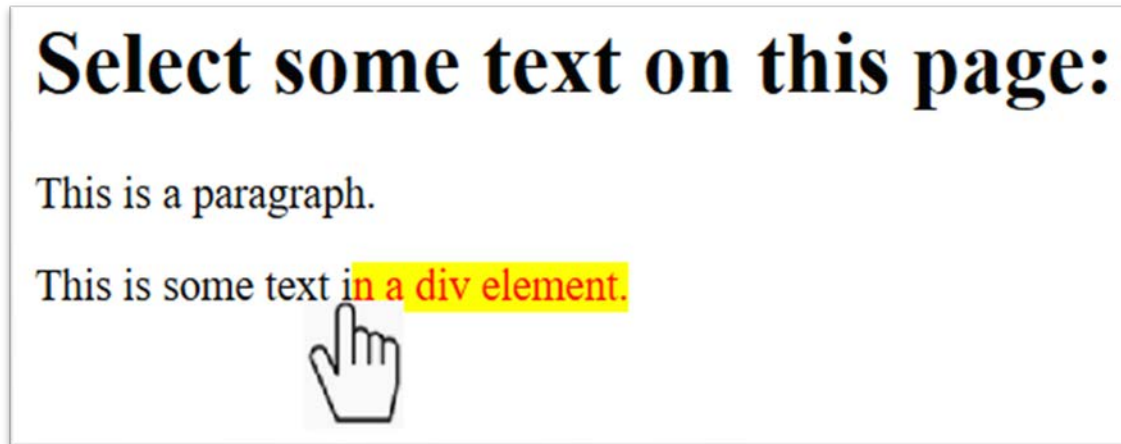
Puede aplicarse a un elemento, pero si no se especifica elemento aplica todo el documento (`h1::selection` o `::selection`).

Ejemplo de código:

CSS:

```
::selection {  
  color: red;
```

```
background: yellow;
}
Html:
<h1>Select some text on this page:</h1>
<p>This is a paragraph.</p>
<div>This is some text in a div element.</div>
Resultado:
```



***Se aplica el estilo cuando se hace la selección del texto**

A `::selection` pueden aplicarse estas propiedades:

- Color
- Background
- Cursor
- Outline

https://www.w3schools.com/css/tryit.asp?filename=trycss3_selection

---9

`::placeholder` permite dar formato al texto de referencia que se muestra en un campo de introducción de texto, como `<input>` o `<textarea>`.

Ejemplo de código:

```
input::placeholder {
  font-weight: bold;
  opacity: 0.5;
  color: red;
}
```

```
input {  
    margin-top: 0.5rem;  
}
```

Html

```
<label for="first-name">Your phone number:</label><br />
```

```
<input id="first-name" type="tel" name="phone" minlength="9"  
maxlength="9" placeholder="It must be 9 digits" />
```

Referencias:

<https://developer.mozilla.org/es/docs/Web/CSS/Pseudo-elements>

<https://lenguajecss.com/css/pseudoelementos/que-son/>

https://www.w3schools.com/css/css_pseudo_elements.asp

https://www.w3schools.com/cssref/css_ref_pseudo_elements.php

PSEUDO-CLASS



Las pseudo clases, junto con los pseudo elementos, permiten aplicar un estilo a un elemento no sólo en relación con el contenido del árbol de documento, sino también en relación a factores externos como el historial del navegador (**:visited**, por ejemplo), el estado de su contenido (como **:checked** en algunos elementos de formulario), o la posición del ratón (como **:hover** que permite saber si el ratón está encima de un elemento o no).

Sintaxis

```
selector:pseudoclase { propiedad: valor; }
```

Tiene **dos puntos** antes del nombre de la propiedad.

Hay un gran número de ellas, actualmente unas 46, veamos algunas de las más comunes:

---1

:hover se utiliza para seleccionar y dar estilo al pasar con el ratón por encima de un elemento, y se utiliza normalmente para enlaces `<a>` pero puede ser utilizada también para otros elementos.

```
img:hover {  
    transform: scale(1.5);  
}
```

Html

```

```

Esta regla aumentar el tamaño de la imagen un 50% al pasar el ratón sobre ella.

Las siguientes pseudo clases suelen aplicarse a los links (etiquetas a):

---2

:link La pseudo clase **:link** se utiliza para seleccionar y dar estilo a los enlaces que no fueron visitados.

Importante: la pseudo clase **:link** selecciona solo a los elementos `<a>` que tienen un atributo **href**.

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

Html

```
<a href="http://w3.misistio.info">misitio.info</a>
```

---3

:visited se utiliza para seleccionar y dar estilo a los enlaces que ya fueron visitados.

```
a:visited {  
    color:purple;  
}
```

---4

:active especifica y selecciona un elemento activo, y se utiliza normalmente para enlaces `<a>` pero puede ser utilizada para otros elementos. Decimos que el elemento es activo cuando el usuario hace clic en este elemento, pero antes de levantar el dedo del botón del ratón: o sea en **"mousedown"**.

```
a:active {  
    color:red;  
}
```

*También la regla **:hover** se aplica a los links.

Estilos para los estados del link <a>

A la hora de dar estilo a los enlaces <a>, es importante comprender cómo utilizar las pseudo clases en sus diferentes estados.

Estos son los estados en que puede estar el link:

- **a:link**- un enlace normal, no visitado.
- **a:visited**- un enlace que el usuario ha visitado.
- **a:focus**- Un enlace cuando tiene el foco*.
- **a:hover**- un enlace cuando el usuario pasa el ratón sobre él.
- **a:active**- un enlace en el momento en que se hace clic en él.

*Se dice que un enlace tiene el foco cuando se salta a este con la tecla Tab del teclado o se le da el foco mediante programación usando `HTMLElement.focus()`.

En muchos casos **a:focus** no se incluye en las reglas aplicadas al link pues en un estado poco común.

Para evitar que los estilos se anulan entre si, hay que seguir un orden a la hora de aplicar las pseudo clases.

```
/* link no visitado */
```

This is a link

Note: **a:hover** **MUST** come in order to be effective.

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

```
/* link visitado, ya hemos hecho clic sobre él */  
a:visited {
```

This is a link

Note: **a:hover** **MUST** come in order to be effective

```
  color: purple;  
}
```

```
/* mouse sobre el link */  
a:hover {
```

This is a link 

Note: `a:hover` MUST con
in order to be effective.

```
color: gray;  
}
```

```
/* link seleccionado, estamos haciendo clic sobre él */  
a:active {
```

~~This is a link~~ 

Note: `a:hover` MUST
in order to be effective

```
color: red;  
}
```

Las pseudo clases `:link`, `:visited`, `:hover` y `:active` son de las primeras pseudo clases implementadas en CSS.

Reglas mnemotécnica para aprender el orden de las pseudo clases en el link :

love and hate

l-:link
v-:visited
h-:hover
a-:active

Cuando usamos focus:

LoVe Fears HAte.

Pseudo clase de negación.

`:not` representa elementos que no coinciden con una lista de selectores. Como evita que se seleccionen elementos específicos, se lo conoce como la *pseudoclase de negación*.

```
/* Selecciona cualquier elemento que NO sea un párrafo */  
:not(p) {  
  color: blue;  
}  
/* elementos <p> que no están en la clase `fancy` */  
p:not(.fancy) {  
  color: green;  
}
```

```
/* Elementos que no son elementos <p> */  
body :not(p) {  
  text-decoration: underline;  
}
```

```
/* Elementos que no son elementos <div> o <span> */
```

```
body :not(div):not(span) {  
  font-weight:bold;  
}
```

Elementos child

Estos selectores son útiles para aplicar estilos específicos en posiciones particulares dentro de listas o elementos anidados.

:first-child — selecciona solo el primer hijo.

:last-child — selecciona solo el último hijo.

:nth-child(n) — selecciona el n-ésimo hijo (contando desde el inicio).

:nth-last-child(n) — selecciona el n-ésimo hijo contando desde el final.

:only-child — Selecciona elementos que son el único hijo de su contenedor.

:only-of-type — Selecciona elementos que son el único de su tipo en el contenedor.

:nth-of-type(n) — Selecciona el n-ésimo hijo de un tipo específico en el contenedor.

:nth-last-of-type(n) — Selecciona el n-ésimo hijo de un tipo específico desde el final.

:first-of-type — Selecciona el primer hijo de su tipo en el contenedor.

:last-of-type — Selecciona el último hijo de su tipo en el contenedor.

Selectores de ejemplo:

`tr:nth-child(odd)` o `tr:nth-child(2n+1)`

Representa las filas impares de una tabla HTML: 1, 3, 5, etc.

`tr:nth-child(even)` o `tr:nth-child(2n)`

Representa las filas pares de una tabla HTML: 2, 4, 6, etc.

`:nth-child(7)`

Representa el séptimo elemento.

`:nth-child(5n)`

Representa los elementos 5, 10, 15, etc.

`:nth-child(3n+4)`

Representa los elementos 4, 7, 10, 13, etc.

`:nth-child(-n+3)`

Representa los primeros tres elementos entre un grupo de hermanos.

`p:nth-child(n)`

Representa cada elemento `<p>` entre un grupo de hermanos. Esto es lo mismo que un simple selector `p`.

`p:nth-child(1)` O `p:nth-child(0n+1)`

Representa cada `<p>` que es el primer elemento entre un grupo de hermanos. Esto es lo mismo que el selector [`:first-child`](#).

Html

```
<ul>
```

```
<li>Primer elemento</li>
```

```
<li>Segundo elemento</li>
```

```
<li>Tercer elemento</li>
```

```
</ul>
```

CSS

```
ul li:first-child { color: blue; /* Solo el primer <li> será azul */ }
```

```
ul li:last-child { color: red; /* Solo el último <li> será rojo */ }
```

```
ul li:nth-child(2) { color: green; /* El segundo <li> será verde */ }
```

```
ul li:nth-child(odd) { background-color: lightgray; /* Los elementos impares  
tendrán fondo gris claro */ }
```

```
ul li:nth-last-child(2) { color: orange; /* El segundo <li> desde el final será  
naranja */ }
```

Html

```
<div class="container">
```

```
<p>Solo hijo</p>
```

```
</div>
```

```
<div class="container">
```

```
<p>Primer hijo</p>
```

```
<p>Segundo hijo</p>
```

```
</div>
```

CSS

```
.container p:only-child { color: purple; /* Solo el <p> que es hijo único será morado */ }
```

Html

```
<div class="container">
```

```
<p>Único párrafo</p>
```

```
<span>Texto adicional</span>
```

```
</div>
```

CSS

```
.container p:only-of-type { color: blue; /* El <p> es el único de su tipo y será azul */ }
```

Html

```
<div class="container">
```

```
<p>Primer párrafo</p>
```

```
<p>Segundo párrafo</p>
```

```
<p>Tercer párrafo</p>
```

```
</div>
```

CSS

```
.container p:nth-of-type(2) { color: green; /* Solo el segundo <p> será verde */ }
```

```
.container p:nth-last-of-type(1) { color: orange; /* El último <p> será naranja */ }
```

Html

```
<div class="container">
```

```
<p>Primer párrafo</p>
```

```
<p>Segundo párrafo</p>
```

```
<span>Primer span</span>
```

```
<p>Último párrafo</p>
```

```
</div>
```

CSS

```
.container p:first-of-type { color: blue; /* El primer <p> será azul */ }
```

```
.container p:last-of-type { color: red; /* El último <p> será rojo */ }
```

<https://developer.mozilla.org/es/docs/Web/CSS/:nth-child>

Para consultar acerca de las funcionalidades y usos de otras pseudo clases visitar los siguientes links.

Referencias:

<https://developer.mozilla.org/es/docs/Web/CSS/Pseudo-classes>

<https://lenguajecss.com/css/pseudoclasses/que-son/>

https://www.w3schools.com/css/css_pseudo_classes.asp

http://w3.unpocodetodo.info/css3/link_visited_hover_active.php

@IMPORT

La regla **@import** permite importar una hoja de estilo en otra hoja de estilo.

La regla **@import** debe estar en la parte superior del documento. Estas reglas deben preceder a todos los otros tipos de reglas, excepto a las reglas [@charset](#).

```
@charset "utf-8";  
@import url(ruta/estilos.css);
```

Sintaxis:

El url a utilizar puede aparecer dentro de una anotación funcional url(), o sencillamente entre comillas:

```
@import url(ruta/estilos.css);  
@import 'ruta/estilos.css';
```

```
@import url("menu.css");           /* Fichero en la misma ruta */  
@import url("menu/sidebar.css");    /* Ruta relativa, dentro de menú/ */  
@import "https://manz.dev/index.css"; /* Ruta absoluta, URL completa */
```

@import y @media

Tampoco puede ser utilizada dentro de un bloque @media, aunque sí es posible importar una hoja de estilos externa para ser utilizada con un cierto tipo de media.

```
@import url(estilos/print.css) print;  
.  
.  
.  
@media print {  
  /* las reglas CSS van aquí */  
}
```

Algunos ejemplos:

Carga la hoja de estilos custom.css

```
@import "custom.css";  
@import url("custom.css");
```

Importe la hoja de estilo "mobstyle.css" sólo si el medio es pantalla y la ventana gráfica tiene un máximo de 768 píxeles

```
@import "mobstyle.css" screen and (max-width: 768px);
```

Esta regla asegura que los estilos en common.css solo se apliquen en dispositivos de pantalla, como monitores, teléfonos, y tablets.

```
@import "common.css" screen;
```

Esta sólo se cargará cuando la pantalla está en modo horizontal (landscape)

```
@import url('landscape.css') screen and (orientation:landscape);
```

Importar estilos de fuentes de texto


```
@import url(https://fonts.googleapis.com/css?family=Raleway:400);
@import url("https://fonts.googleapis.com/css?family=Raleway:400");
@import 'https://fonts.googleapis.com/css?family=Raleway:400';
```

Importa la hoja de estilos cuando el dispositivo de salida es un proyector o una tv

```
@import url("bluish.css") projection, tv;
```

Importa la hoja de estilos cuando el dispositivo de salida es una pantalla o un proyector

```
@import "common.css" screen, projection;
```

Carga una hoja de estilo desde una ruta interna de navegador (poco común)

```
@import url("chrome://communicator/skin/");
```



Para tener en cuenta

La declaración `@import` puede tener un efecto negativo en el rendimiento de la página. Por el tiempo que conlleva la carga de estos archivos.

<https://caniuse.com/>

Browser Support

The numbers in the table specify the first browser version that fully supports the property.

Property					
@import	1.0	5.5	1.0	1.0	3.5

Esta propiedad CSS está soportada por los principales navegadores.

Referencias:

<https://developer.mozilla.org/es/docs/Web/CSS/@import>

<https://lenguajecss.com/cascada-css/estructura/regla-import/>

https://www.w3schools.com/cssref/pr_import_rule.php

<http://w3.unpocodetodo.info/css3/at-import.php>