

EL CÓDICE DEL DESARROLLADOR

Libro de consulta rápida sobre el lenguaje de hojas de
estilo **CSS3**.

VOLUMEN II

CÉSAR DELGADO

© 2026 César Delgado Ruiz.

El Códice del Desarrollador.

Manual de consulta rápida sobre el lenguaje de hojas de estilo CSS.

Reservados todos los derechos. Queda prohibida la reproducción total o parcial de esta obra por cualquier medio, incluido el electrónico, sin la autorización escrita del autor.

CSS es un estándar del W3C compuesto por múltiples módulos. Las marcas comerciales y nombres citados en esta obra pertenecen a sus respectivos propietarios. Este manual es una recopilación original de referencia y no está afiliado al W3C.

ISBN: 9798177283593

Índice general

Parte I — Consulta diaria.....	5
Prólogo.....	7
Cómo usar este manual.....	11
Capítulo 1 — Selectores.....	13
Capítulo 2 — Cascada, especificidad y herencia.....	37
Capítulo 3 — Valores, tipos y unidades.....	45
Capítulo 4 — Color y fondo.....	55
Capítulo 5 — Texto y tipografía.....	71
Capítulo 6 — Modelo de caja.....	97
Capítulo 7 — Visualización y posición.....	107
Capítulo 8 — Flexbox.....	119
Capítulo 9 — Grid.....	131
Capítulo 10 — Transformaciones, transiciones y animaciones.....	147
Capítulo 11 — At-rules.....	165
Capítulo 12 — Variables y funciones.....	175
Capítulo 13 — Propiedades lógicas.....	189
Capítulo 14 — Efectos visuales.....	201

Capítulo 15 — Containment y rendimiento.....	207
Capítulo 16 — Scroll Snap.....	213
Capítulo 17 — Multi-column Layout.....	219
Parte II — Ampliación y CSS reciente.....	225
Capítulo 18 — Contenido generado y fragmentación.....	227
Capítulo 19 — Estilos de Lista y Tablas.....	237
Capítulo 20 — Scroll y Scrollbar.....	245
Capítulo 21 — Apariencia, Color Scheme y Navegación.....	251
Capítulo 22 — Texto Avanzado.....	259
Capítulo 23 — Tipografía Avanzada.....	271
Capítulo 24 — Escritura y Dirección.....	285
Capítulo 25 — Animación Avanzada y Timeline.....	291
Capítulo 26 — Container Queries Avanzadas.....	297
Capítulo 27 — View Transitions y Position Try.....	303
Capítulo 28 — Descriptores de At-Rules.....	315
Capítulo 29 — Clip, Shape y Mask.....	323
Capítulo 30 — SVG en CSS.....	337
Capítulo 31 — Funciones Avanzadas.....	351
Capítulo 32 — Selectores avanzados y pseudo-elementos modernos..	369
Capítulo 33 — At-Rules Especiales.....	387
Capítulo 34 — Containment Avanzado y Columnas.....	399

Capítulo 35 — Miscelánea y Propiedades de Interactividad.....	403
Capítulo 36 — Scroll Timeline y Animaciones de Scroll.....	411
Capítulo 37 — Formularios e interfaz.....	417
Anexo A — Propiedades obsoletas y no recomendadas.....	429
Anexo B — Derechos sobre el contenido.....	433
Anexo C — Soporte de navegadores (Actualizado).....	435
Anexo D — Entidades y escapes de caracteres en CSS.....	453
Glosario.....	457
Índice analítico.....	459
Referencias.....	463

Parte I

Consulta diaria

Prólogo

Todos los desarrolladores, independientemente de su experiencia, comparten una misma realidad: mientras maquetan una página web, surgen continuamente pequeñas dudas que interrumpen el trabajo.

¿Cuál era la propiedad correcta? ¿Qué valor es el más adecuado para este caso? ¿Cómo era exactamente la sintaxis? ¿Esta función sigue siendo válida en CSS actual?

Son preguntas sencillas, pero cada una de ellas tiene un coste. Se abandona el editor de código, se abre el navegador, se consulta la documentación, se revisan varios ejemplos y, finalmente, se vuelve al proyecto. Cinco minutos aquí, diez minutos allá... Al final de la jornada, esas pequeñas interrupciones representan una pérdida considerable de tiempo y, sobre todo, rompen el ritmo de trabajo.

Este vademécum nace precisamente para evitar esas interrupciones.

No pretende sustituir a los manuales de aprendizaje ni a la documentación oficial. Tampoco pretende competir con los buscadores o con las herramientas de inteligencia artificial. Su objetivo es mucho más práctico: convertirse en una guía de consulta rápida que permita resolver las dudas más frecuentes sin abandonar el flujo de programación.

En estas páginas encontrarás reunidas propiedades de CSS, sus selectores, at-rules, funciones, valores y ejemplos de uso organizados de forma clara y accesible. La información ha sido seleccionada para que puedas localizar rápidamente lo que necesitas cuando estás desarrollando un proyecto real.

Este manual está pensado especialmente para estudiantes, desarrolladores junior, programadores que trabajan con varias tecnologías y profesionales que desean disponer de una referencia rápida sin tener que navegar constantemente entre diferentes páginas web o documentación dispersa.

Este libro reúne el vademécum completo, organizado en dos partes. La **Parte I — Consulta diaria** recoge lo que consultarás casi cada día al maquetar: selectores, cascada, modelo de caja, Flexbox, Grid, color, texto, animaciones habituales y formularios. La **Parte II — Ampliación y CSS reciente** amplía con tipografía avanzada, animaciones con timeline, container queries, view transitions, máscaras y otros APIs de frontera; no es obligatoria para maquetar bien, pero está ahí cuando un proyecto te lo pida o te pique la curiosidad.

Con el paso del tiempo descubrirás que muchas consultas dejarán de ser necesarias porque habrás interiorizado los conceptos. Hasta que ese momento llegue, disponer de una referencia fiable te permitirá trabajar con mayor seguridad, reducir errores y dedicar más tiempo a programar que a buscar información.

Ese es el propósito de este vademécum: ayudarte a mantener el foco en lo importante.

Porque programar ya es suficientemente complejo como para perder tiempo buscando aquello que podrías tener al alcance de la mano.

Al final de este libro encontrará un **glosario** de términos, un **índice analítico** para localizar cualquier propiedad, selector o concepto por su

nombre, y las **referencias oficiales** del estándar (W3C, CSSWG y MDN), con la fecha de verificación de los datos de compatibilidad.

Espero que esta obra se convierta en una herramienta útil durante tu aprendizaje y también en tu trabajo diario, acompañándote cada vez que una pequeña duda amenace con detener tu productividad.

César Delgado Ruiz

Cómo usar este manual

Este libro es una referencia rápida para el trabajo diario de maquetación. La Parte I cubre lo esencial del día a día; la Parte II amplía con CSS avanzado y APIs recientes para cuando un proyecto lo pida o quieras explorar temas especializados.

Formato de las fichas

- Cada **propiedad** se presenta con su **tipo de valor**, una **descripción**, los **valores** posibles, su **valor inicial**, si **hereda** o no, a qué **elementos aplica**, y un **ejemplo** de uso.
- Cada **selector**, **pseudoclase** y **pseudoelemento** se explica con su **descripción**, un **ejemplo** y cómo se vería.
- Las **at-rules** básicas (como `@media` o `@font-face`) se tratan en el Capítulo 11.
- Las **at-rules** especiales se tratan en el Capítulo 33.
- Las **propiedades obsoletas** tienen su propio anexo con el reemplazo moderno.
- Los **datos de especificidad y herencia** se explican en el Capítulo 2.

- El **glosario** define los términos técnicos empleados en el libro; el **índice analítico** permite localizar cualquier propiedad, selector o concepto por su nombre; las **referencias** citan la documentación oficial (W3C, CSSWG y MDN).

Símbolos usados

- **heredada** → la propiedad se transmite a los descendientes.
- **experimental / obsoleto** → marca únicamente propiedades, at-rules o valores no recomendados o aún no estándar.
-  Propiedad obsoleta o deprecated (ver Anexo A).
-  Consejo de buenas prácticas o accesibilidad.
-  Propiedad que no hereda.
-  Propiedad shorthand (abreviatura de varias propiedades).

Capítulo 1

Selectores

Este capítulo recopila todos los mecanismos de selección de CSS. Comienza con los **selectores básicos**: universal (*), por tipo o elemento (div), por clase (.clase), por identificador (#id) y por atributo ([atributo]) con sus siete operadores de coincidencia (^=, \$=, *=, ~=, |=, etc.). Luego cubre los **combinadores** que relacionan elementos entre sí: descendiente (espacio), hijo directo (>), hermano adyacente (+) y hermano general (~). Continúa con las **pseudoclases** para estilizar estados dinámicos y estructurales: estado del enlace (:link, :visited, :hover, :active, :focus), posición entre hermanos (:first-child, :last-child, :nth-child(), :not()), el nuevo selector padre :has(), los agrupadores :is() y :where(), y pseudoclases de formulario (:checked, :disabled, :empty, :root). Finaliza con los **pseudoelementos** que generan contenido virtual: ::before, ::after, ::first-line, ::first-letter, ::placeholder y ::selection. El **peso (especificidad)** que cada selector aporta se explica en el Capítulo 2.

Caso de uso típico: Personalizar estados de interacción en formularios (resaltar campos inválidos con :invalid), seleccionar filas alternas

de una tabla con `:nth-child(even)`, o estilizar el primer párrafo de una sección con `::first-letter`.

Selectores básicos

Los selectores elementales, de clase, de identificador y de atributo que apuntan a elementos individuales del DOM.

* (universal)

- **Descripción:** Selecciona cualquier elemento del documento.
- **Valores / sintaxis:** asterisco `*`.
- **Ejemplo:**

```
* { margin: 0; padding: 0; }
```
- **Cómo se vería:** Todas las cajas del documento quedan sin márgenes ni relleno.

div (elemento o tipo)

- **Descripción:** Selecciona todos los elementos de un tipo (etiqueta) concreto.
- **Ejemplo:**

```
div { color: red; }
```
- **Cómo se vería:** Todos los div se pintan de rojo.

.clase (clase)

- **Descripción:** Selecciona los elementos que llevan la clase indicada en su atributo class.
- **Ejemplo:**

```
.resaltado { background: yellow; }
```
- **Cómo se vería:** Texto marcado con la clase se resalta en amarillo.

#id (identificador)

- **Descripción:** Selecciona el elemento único cuyo id coincide. Solo puede existir uno por página.
- **Ejemplo:**

```
#cabecera { border-bottom: 2px solid black; }
```
- **Cómo se vería:** El elemento con ese identificador recibe la línea inferior.

[atributo] (atributo)

- **Descripción:** Selecciona elementos según la presencia y/o el valor de un atributo. Existe un selector de presencia y seis operadores de coincidencia. Los operadores admiten los modificadores i (insensible a mayúsculas) y s (sensible a mayúsculas).

- **Valores / sintaxis:**

Selector	Significado	Ejemplo
[attr]	Tiene el atributo (presencia)	[href]
[attr="valor"]	Valor exacto	[href="https://example.com"]
[attr^="valor"]	Empieza por	[href^="https"]
[attr\$="valor"]	Termina en	[href\$=".pdf"]
[attr*="valor"]	Contiene la subcadena	[href*="example"]
[attr~="valor"]	Palabra en lista (separada por espacios)	[title~="nuevo"]
[attr\ ="valor"]	Prefijo con guion o valor exacto	[lang\ ="es"]

- **Ejemplo:**

```
[href^="https"] { color: green; }  
[href$=".pdf"] { color: red; }  
[lang|="es"] { font-style: italic; }
```

- **Cómo se vería:** Los enlaces seguros se ven verdes, los documentos PDF rojos, y el texto en español se muestra en cursiva.

Combinadores

Símbolos que relacionan dos selectores para ampliar o restringir el alcance: descendiente, hijo directo, hermano adyacente y hermano general.

div p (descendiente)

- **Descripción:** Selecciona todo p dentro de un div, a cualquier nivel de anidación.

- **Ejemplo:**

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

- **Cómo se vería:** Todos los párrafos que cuelgan de un div se pintan de rojo.

div > p (hijo directo)

- **Descripción:** Selecciona el p que es hijo directo (primer nivel) de un div.
- **Ejemplo:**

```
div > p { color: red; }
```
- **Cómo se vería:** Solo los párrafos inmediatos del div cambian a rojo, no los anidados más profundo.

h2 + p (hermano adyacente)

- **Descripción:** Selecciona el primer p inmediatamente después de un h2, con el mismo padre.
- **Ejemplo:**

```
h2 + p { color: red; }
```
- **Cómo se vería:** El primer párrafo que sigue a cada h2 se pinta de rojo.

h2 ~ p (hermano general)

- **Descripción:** Selecciona cualquier p que siga a un h2 (mismo padre), sin necesidad de ser adyacente.
- **Ejemplo:**

```
h2 ~ p { color: red; }
```
- **Cómo se vería:** Todos los párrafos hermanos posteriores a cada h2 se pintan de rojo.

Pseudoclases

Palabras clave que seleccionan elementos en un estado especial (hover, focus, checked) o según su posición en el árbol DOM (first-child, nth-child).

a: hover

- **Descripción:** Aplica cuando el ratón pasa por encima del elemento.
- **Ejemplo:**

```
a: hover { color: red; }
```
- **Cómo se vería:** Enlace al pasar el ratón se muestra en rojo.

a: active

- **Descripción:** Aplica mientras se mantiene presionado el clic sobre el elemento.
- **Ejemplo:**

```
a: active { color: red; }
```
- **Cómo se vería:** El enlace se vuelve rojo en el instante del clic.

a: focus

- **Descripción:** Aplica cuando el elemento recibe el foco (teclado, tabulación o clic).
- **Ejemplo:**

```
a: focus { color: red; }
```
- **Cómo se vería:** El enlace se pinta de rojo al tabular hasta él.

:focus-visible

- **Descripción:** Aplica cuando el elemento recibe el foco **y el navegador determina que debe mostrarse un indicador de foco** (navegación por teclado, no clic de ratón). Similar a :focus pero con lógica de visibilidad del agente de usuario.

- **Ejemplo:**

```
a:focus-visible { outline: 2px solid blue; outline-  
offset: 2px; }  
button:focus-visible { box-shadow: 0 0 0 3px rgba(0,  
100, 255, 0.5); }
```

- **Cómo se vería:** El indicador de foco solo aparece al navegar con teclado, no al hacer clic con el ratón.

💡 **Accesibilidad:** :focus-visible es la opción recomendada para indicadores de foco accesibles. Permite mostrar el indicador solo cuando el usuario lo necesita (navegación por teclado) y evitar el “foco fantasma” en clics de ratón.

a:visited

- **Descripción:** Aplica a los enlaces que el usuario ya ha visitado.

- **Ejemplo:**

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

- **Cómo se vería:** Los enlaces ya visitados se muestran en rojo.

a:link

- **Descripción:** Aplica a los enlaces no visitados.

- **Ejemplo:**

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

- **Cómo se vería:** Los enlaces sin visitar se muestran en rojo.

li:first-child

- **Descripción:** Selecciona el primer hijo de su padre.

- **Ejemplo:**

```
li:first-child { color: red; }
```

- **Cómo se vería:** El primer elemento de cada lista se pinta de rojo.

li:last-child

- **Descripción:** Selecciona el último hijo de su padre.

- **Ejemplo:**

```
li:last-child { color: red; }
```

- **Cómo se vería:** El último elemento de cada lista se pinta de rojo.

li:nth-child()

- **Descripción:** Selecciona los hijos según su posición (número, patrón o palabra clave odd/even).

- **Valores / sintaxis:** 2 (posición 2), odd (impares), even (pares), 3n (cada 3), 2n+1, etc.

- **Ejemplo:**

```
li:nth-child(2) { color: red; }  
li:nth-child(odd) { color: red; }  
li:nth-child(3n) { color: red; }
```

- **Cómo se vería:** Se resaltan las filas en la posición indicada (p. ej. las impares en una tabla).

li:not(.active)

- **Descripción:** Selecciona los elementos que **no** cumplen el selector que recibe.

- **Ejemplo:**

```
li:not(.active) { color: red; }
```

- **Cómo se vería:** Todos los elementos de la lista salvo el que tiene la clase active se pintan de rojo.

p:has(> img)

- **Descripción:** Selecciona un elemento **si contiene** a otro que cumple el selector interno. El “selector padre”.

- **Ejemplo:**

```
p:has(> img) { color: red; }  
.form:has(:invalid) { border: 1px solid red; }
```

- **Cómo se vería:** El párrafo que contiene una imagen directamente se pinta de rojo.

:is(h1, h2, h3)

- **Descripción:** Agrupa varios selectores en uno solo. La especificidad resultante es la del selector más específico del grupo.

- **Ejemplo:**

```
:is(h1, h2, h3) { color: red; }
```

- **Cómo se vería:** Los encabezados h1, h2 y h3 se pintan de rojo.

:where(h1, h2, h3)

- **Descripción:** Igual que `:is()` pero con **especificidad siempre cero**, fácil de sobrescribir.

- **Ejemplo:**

```
:where(h1, h2, h3) { color: red; }
```

- **Cómo se vería:** Los encabezados h1, h2 y h3 se pintan de rojo, pero cualquier otra regla puede ganarles fácilmente.

:checked

- **Descripción:** Aplica a casillas (checkbox), botones de opción (radio) u opciones de lista ya marcadas.

- **Ejemplo:**

```
:checked { color: red; }
```

- **Cómo se vería:** El control marcado cambia su aspecto.

:disabled

- **Descripción:** Aplica a elementos deshabilitados (con el atributo `disabled`).

- **Ejemplo:**

```
:disabled { color: red; }
```

- **Cómo se vería:** Los controles desactivados se muestran en rojo.

:empty

- **Descripción:** Selecciona elementos que no tienen hijos ni contenido de texto.

- **Ejemplo:**

```
:empty { color: red; }
```

- **Cómo se vería:** Las cajas vacías reciben el estilo.

:root

- **Descripción:** Selecciona la raíz del documento (<html>). Lugar habitual para definir variables globales.

- **Ejemplo:**

```
:root { color: red; }
```

- **Cómo se vería:** Aplica a todo el documento.

:focus-within

- **Descripción:** Aplica cuando el elemento **o cualquiera de sus descendientes** recibe el foco. Útil para resaltar contenedores que contienen campos activos.

- **Ejemplo:**

```
form:focus-within { border-color: blue; background:  
#f0f8ff; }  
.card:focus-within { box-shadow: 0 0 10px rgba(0, 0, 0,  
0.1); }
```

- **Cómo se vería:** El formulario cambia de color cuando cualquier campo interno recibe el foco.

:default

- **Descripción:** Selecciona el elemento por defecto entre un grupo de elementos relacionados (botones de radio, checkboxes, opciones de <select>).

- **Ejemplo:**

```
input:default { border: 2px solid green; }  
option:default { font-weight: bold; }
```

- **Cómo se vería:** El elemento predeterminado (marcado por defecto) se resalta.

:indeterminate

- **Descripción:** Selecciona checkboxes o botones de radio en estado indeterminado (ni marcado ni desmarcado), típicamente establecido por JavaScript.
- **Ejemplo:**

```
input:indeterminate { outline: 2px dashed orange; }
```

- **Cómo se vería:** Un checkbox en estado indeterminado muestra un guion en lugar de una marca.

:valid

- **Descripción:** Selecciona elementos de formulario cuyo contenido cumple con las reglas de validación del navegador.
- **Ejemplo:**

```
input:valid { border-color: green; background: #f0fff0; }  
}
```

- **Cómo se vería:** Los campos válidos muestran un borde verde.

:invalid

- **Descripción:** Selecciona elementos de formulario cuyo contenido **no** cumple con las reglas de validación.
- **Ejemplo:**

```
input:invalid { border-color: red; background: #fff0f0; }  
}
```

- **Cómo se vería:** Los campos inválidos muestran un borde rojo.

:required

- **Descripción:** Selecciona elementos que tienen el atributo `required`.

- **Ejemplo:**

```
input:required { border-left: 3px solid red; }
```

- **Cómo se vería:** Los campos obligatorios tienen un borde rojo a la izquierda.

:optional

- **Descripción:** Selecciona elementos que **no** tienen el atributo `required`.

- **Ejemplo:**

```
input:optional { border-left: 3px solid green; }
```

- **Cómo se vería:** Los campos opcionales tienen un borde verde a la izquierda.

:read-only

- **Descripción:** Selecciona elementos que no pueden ser editados por el usuario (atributo `readonly`).

- **Ejemplo:**

```
input:read-only { background: #eee; color: #666; }
```

- **Cómo se vería:** Los campos de solo lectura tienen fondo gris.

:read-write

- **Descripción:** Selecciona elementos que **pueden** ser editados por el usuario.

- **Ejemplo:**

```
input:read-write { background: white; border: 1px solid #ccc; }
```

- **Cómo se vería:** Los campos editables tienen fondo blanco.

:first-of-type

- **Descripción:** Selecciona el primer hermano de su tipo entre sus hermanos.

- **Ejemplo:**

```
p:first-of-type { font-weight: bold; }
```

- **Cómo se vería:** El primer párrafo de su tipo aparece en negrita.

:last-of-type

- **Descripción:** Selecciona el último hermano de su tipo entre sus hermanos.

- **Ejemplo:**

```
p:last-of-type { font-style: italic; }
```

- **Cómo se vería:** El último párrafo de su tipo aparece en cursiva.

:only-of-type

- **Descripción:** Selecciona elementos que son el **único** hijo de su tipo entre sus hermanos.
- **Ejemplo:**

```
p:only-of-type { color: red; }
```
- **Cómo se vería:** El párrafo único de su tipo se pinta de rojo.

:nth-of-type()

- **Descripción:** Selecciona elementos que ocupan una posición específica entre los hermanos de su tipo.
- **Valores / sintaxis:** 2 (segundo), odd (impares), even (pares), 2n+1, etc.
- **Ejemplo:**

```
li:nth-of-type(2) { color: red; }  
p:nth-of-type(odd) { background: #f0f0f0; }
```
- **Cómo se vería:** Se estilizan los elementos según su posición entre los hermanos de su tipo.

:nth-last-of-type()

- **Descripción:** Similar a :nth-of-type() pero contando desde el final.
- **Ejemplo:**

```
li:nth-last-of-type(1) { font-weight: bold; }
```
- **Cómo se vería:** El último elemento de su tipo se resalta.

:lang()

- **Descripción:** Selecciona elementos según su idioma (atributo lang o xml:lang).
- **Ejemplo:**

```
:Lang(es) { font-style: italic; }  
:Lang(en) { font-style: normal; }  
blockquote:Lang(fr) { quotes: "«" "»"; }
```
- **Cómo se vería:** El texto en español se muestra en cursiva, el inglés en normal.

:dir()

- **Descripción:** Selecciona elementos según su dirección de texto: ltr (izquierda a derecha) o rtl (derecha a izquierda).
- **Ejemplo:**

```
:dir rtl { text-align: right; }  
:dir ltr { text-align: left; }
```
- **Cómo se vería:** El texto se alinea según la dirección del documento.

:user-valid

- **Descripción:** Selecciona elementos de formulario que el navegador considera **válidos después de la interacción del usuario** (no solo según la validación estática). A diferencia de :valid, solo se aplica cuando el usuario ha interactuado con el campo.
- **Ejemplo:**

```
input:user-valid { border-color: green; }  
input:user-invalid { border-color: red; }
```

- **Cómo se vería:** Los campos se pintan de verde o rojo solo después de que el usuario escribe en ellos, evitando errores al cargar la página.

:user-invalid

- **Descripción:** Selecciona elementos de formulario que el navegador considera **inválidos después de la interacción del usuario**. Equivalente a :invalid pero solo tras interacción.

- **Ejemplo:**

```
input:user-invalid { border-color: red; background: #fff0f0; }
```

- **Cómo se vería:** El campo muestra error solo después de que el usuario ha escrito algo inválido.

:popover-open

- **Descripción:** Selecciona elementos popovers que están actualmente **abiertos/visibles** (con el atributo popover o abierto con showPopover()).

- **Ejemplo:**

```
[popover]:popover-open { opacity: 1; transform: scale(1); }
```

- **Cómo se vería:** El popover abierto se muestra con opacidad total y escala normal.

:modal

- **Descripción:** Selecciona elementos que están en estado modal (exclusivo), como <dialog> abierto con showModal() o elementos con popover="auto".

- **Ejemplo:**

```
:modal { border: 2px solid blue; }  
:modal::backdrop { background: rgba(0, 0, 0, 0.5); }
```

- **Cómo se vería:** El diálogo modal muestra un borde azul y un fondo semitransparente.

:fullscreen

- **Descripción:** Selecciona el elemento que está actualmente en pantalla completa.

- **Ejemplo:**

```
:fullscreen { background: black; }
```

- **Cómo se vería:** El elemento en pantalla completa tiene fondo negro.

:defined

- **Descripción:** Selecciona cualquier elemento que haya sido definido (constructor registrado para custom elements, o elementos HTML estándar). Útil para evitar FOUC en web components.

- **Ejemplo:**

```
:not(:defined) { opacity: 0; }  
mi-componente::defined { display: none; }
```

- **Cómo se vería:** Los elementos no definidos se ocultan hasta que su constructor esté registrado.

:placeholder-shown

- **Descripción:** Selecciona campos de formulario que muestran su placeholder (cuando están vacíos y sin foco).

- **Ejemplo:**

```
input:placeholder-shown { border-style: dashed; }
```

- **Cómo se vería:** Los campos vacíos muestran borde discontinuo mientras muestran el placeholder. ### Pseudoelementos

Selectores que crean elementos virtuales dentro de un elemento real (before, after) o estilizan partes específicas del contenido (first-line, first-letter, placeholder, selection).

p::before

- **Descripción:** Genera contenido **antes** del elemento. Requiere la propiedad content.

- **Ejemplo:**

```
p::before { content: "→ "; }
```

- **Cómo se vería:** → Flecha insertada al inicio del párrafo

p::after

- **Descripción:** Genera contenido **después** del elemento. Requiere la propiedad content.

- **Ejemplo:**

```
p::after { content: " ←"; }
```

- **Cómo se vería:** Párrafo de texto ← Flecha insertada al final

p::first-line

- **Descripción:** Estiliza la primera línea de un bloque de texto.

- **Ejemplo:**

```
p::first-line { font-weight: bold; }
```

- **Cómo se vería:** Esta **primera línea aparece en negrita** y el resto del párrafo mantiene su estilo normal.

p::first-letter

- **Descripción:** Estiliza la primera letra de un bloque (efecto capitular).
- **Ejemplo:**

```
p::first-letter { font-size: 2em; }
```
- **Cómo se vería:** Esta primera letra se agranda como capitular.

input::placeholder

- **Descripción:** Estiliza el texto de marcador de posición de un campo de formulario.
- **Ejemplo:**

```
input::placeholder { color: gray; }
```
- **Cómo se vería:** El texto de ayuda dentro del campo se muestra en gris.

::selection

- **Descripción:** Estiliza el texto seleccionado por el usuario.
- **Ejemplo:**

```
::selection { background: yellow; }
```
- **Cómo se vería:** Al seleccionar texto, el fondo se pinta de amarillo.

::backdrop

- **Descripción:** Estiliza el fondo detrás de un elemento en pantalla completa o un diálogo modal. Se aplica al “fondo” que cubre el resto de la página.
- **Ejemplo:**

```
dialog::backdrop { background: rgba(0, 0, 0, 0.5); }  
video::backdrop { background: black; }
```
- **Cómo se vería:** El fondo detrás del diálogo se oscurece con un overlay semitransparente.

::marker

- **Descripción:** Estiliza la viñeta o el número de un elemento de lista (). Permite personalizar el marcador sin cambiar el contenido del elemento.
- **Valores:** color, content, font-properties, propiedades de texto.
- **Ejemplo:**

```
li::marker { color: red; font-size: 1.5em; }  
li::marker { content: "→ "; }
```
- **Cómo se vería:** Las viñetas de la lista aparecen en rojo y más grandes, o reemplazadas por una flecha.

::file-selector-button

- **Descripción:** Estiliza el botón de selección de archivos de un <input type="file">.
- **Ejemplo:**

```
::file-selector-button {  
  background: #007bff;  
  color: white;
```

```
border: none;
padding: 8px 16px;
border-radius: 4px;
cursor: pointer;
}
```

- **Cómo se vería:** El botón “Elegir archivo” se ve como un botón personalizado azul.

::cue

- **Descripción:** Estiliza las pistas de subtítulos WebVTT en elementos de video.
- **Ejemplo:**

```
::cue { background: rgba(0, 0, 0, 0.7); color: yellow;
font-size: 1.2em; }
::cue(b) { color: white; }
```

- **Cómo se vería:** Los subtítulos del video tienen fondo semitransparente y texto amarillo.

::slotted()

- **Descripción:** Selecciona elementos que han sido insertados en un slot de un Shadow DOM. Se usa dentro de estilos definidos en el documento principal que afectan a elementos proyectados en un componente web.
- **Sintaxis:** ::slotted(selector)
- **Ejemplo:**

```
::slotted(p) { color: blue; }
::slotted(.highlight) { background: yellow; }
```

- **Cómo se vería:** Los párrafos y elementos con clase .highlight que se insertan en un slot se estilan según las reglas del componente.

💡 **Buenas prácticas:** Usa selectores específicos pero no excesivamente anidados. Mantén la especificidad baja para facilitar la sobreescritura. Prefiere clases CSS a selectores de tipo para estilos reutilizables.