CSS

Pseudoclases + pseudo-elementos

CSS

Pseudoclases + pseudo-elementos

Pseudo-clases (:)

Filtran elementos reales según su estado actual.

(Ej: :hover, :focus, :checked)

Pseudo-elementos (::)

Generan partes virtuales que no existen en el DOM.

(Ej: ::before, ::after,)

CSS

Pseudo-clases

En el motor de CSS:

- El navegador parsea el DOM.
- Genera el árbol de elementos existentes.
- Al aplicar las reglas CSS:

La pseudoclase filtra elementos del árbol real según su estado actual.

- No crea nodos nuevos, solo modifica el selector.
- Es un filtro dinámico evaluado en tiempo de render.

button:hover

Funcionamiento con CSS

"Busco todos los button..."

"¿Está el ratón encima? Si sí → aplico la regla."

👉 Internamente:

```
for each element in DOM:
    if element.tag == 'button' and element.isHovered():
        apply(styles)
```

CSS

::before → mete contenido antes del contenido real.

::after → mete contenido después.

::first-letter → estiliza solo la primera letra.

::selection → estiliza el texto cuando lo seleccionas.

Pseudo-elementos

En el motor CSS:

El DOM tiene el nodo real (por ejemplo <p>).

Cuando el motor CSS evalúa la regla:
Crea nodos virtuales internos (pseudo-boxes) para representar partes del elemento.

No aparecen en el DOM real.
Existen sólo en el árbol de renderizado (layout tree / box tree).

Pseudoclase
El estado no está en el DOM, pero actúa como clase

Pseudo-elemento
El nodo no está en el DOM, pero actúa como elemento

Tipo	¿Qué hace?	Símbolo
Pseudoclase	Estado o condición	`:`
Pseudo-elemento	Parte virtual o decorativa	`::`

	Antes	Ahora
Pseudoclase	`:hover`	`:hover`
Pseudo-elemento	`:before`	`::before`

Buenas practicas

Selectores demasiado específicos

Mala idea escribir selectores largos y pesados como:

.container > ul > li:first-child > a:hover

¿Por qué?

Difícil de mantener

Difícil de sobrescribir

Código menos legible

Mejor:

Usa clases específicas y simples

Usa pseudoclases con moderación y claridad

Ejemplo: .btn:hover en vez de .header .menu ul li a:hover

Buenas practicas

Uso excesivo del !important

!important rompe la logica de la prioridad,

Rompe la cascada, dificulta mantenimiento, y genera caos.

Solo úsalo en emergencia absoluta, y no para forzar pseudoclases o pseudo-elementos.

Solución: refactoriza selectores,

Compatibilidad entre navegadores

Algunos pseudoelementos y pseudoclases NO tienen soporte en todos Navegadores

:has() → Chrome 105+, Safari 15.4+, Firefox 121+ (sí, Firefox viene muy tarde)

:indeterminate → funciona solo en checkboxes y no es estilo universal

::selection → bien soportado, pero cuidado con propiedades que no aplican