09 CSS: Pseudoclases de Estado

Las **pseudoclases** en CSS son palabras clave que se añaden a los selectores para especificar un *estado especial* o una característica del elemento o elementos seleccionados. No seleccionan basándose en el nombre, atributos o contenido del elemento en sí, sino en condiciones externas o estados que cambian con el tiempo, como las interacciones del usuario. Se distinguen porque comienzan con dos puntos (`:`).

En esta sección, nos centraremos en las **pseudoclases de estado**, que se aplican a los elementos cuando se encuentran en un estado particular, a menudo como resultado de la interacción del usuario. Son fundamentales para crear interfaces web dinámicas, interactivas y accesibles.

Veamos las más comunes e importantes:

1. `:link`
   - **Selecciona:** Enlaces (`<a>` con atributo `href`) que aún **no han sido visitados** por el usuario.
   - **Uso:** Permite diferenciar visualmente los enlaces que el usuario ya ha seguido de los que no.
   - **Ejemplo:**

     ```
     a:link {
       color: blue; /* Color para enlaces no visitados */
       text-decoration: underline;
     }
     ```

2. `:visited`
   - **Selecciona:** Enlaces (`<a>` con atributo `href`) que **ya han sido visitados** por el usuario.
   - **Uso:** Complementa a `:link` para dar feedback visual sobre el historial de navegación del usuario.
   - **Importante (Privacidad):** Por razones de privacidad, los navegadores restringen severamente los estilos que se pueden aplicar con `:visited`. Principalmente se pueden cambiar propiedades relacionadas con el color (`color`, `background-color`, `border-color`, `outline-color`) y colores de relleno/trazo SVG. No se pueden cambiar layouts, tamaños de fuente, añadir fondos de imagen, etc., ya que esto podría usarse para extraer el historial de navegación del usuario.
   - **Ejemplo:**

     ```
     a:visited {
       color: purple; /* Color para enlaces visitados */
     }
     ```

3. `:hover`
   - **Selecciona:** Cualquier elemento cuando el cursor del ratón del usuario **se encuentra sobre él**.

- **Uso:** Esencial para la interactividad. Se usa comúnmente para cambiar el fondo, color, subrayado de enlaces, botones, elementos de menú, o incluso para mostrar/ocultar otros elementos al pasar el ratón. Funciona en la mayoría de los elementos, no solo enlaces.
- **Ejemplo:**

```css
button:hover {
  background-color: darkgrey;
  cursor: pointer; /* Cambia el cursor a una mano */
}
.menu-item:hover {
  background-color: lightblue;
}
```

4. `:active`
   - **Selecciona:** Un elemento durante el breve momento en que está siendo **activado** por el usuario. Para enlaces y botones, esto ocurre entre el momento en que se presiona el botón del ratón y se suelta.
   - **Uso:** Proporciona una respuesta visual inmediata a la acción del usuario (feedback de "click").
   - **Ejemplo:**

```css
a:active {
  color: red; /* Cambia a rojo mientras se hace clic */
}
button:active {
  position: relative; /* Necesario para usar top/left */
  top: 1px; /* Simula un pequeño hundimiento */
}
```

5. `:focus`
   - **Selecciona:** Un elemento que actualmente tiene el **foco de entrada**. Un elemento recibe el foco cuando es seleccionado por el usuario para interactuar con él, ya sea mediante un clic del ratón o, muy importante, a través de la navegación con teclado (tecla Tab).
   - **Uso:** Crucial para la **accesibilidad**. Permite a los usuarios que navegan con teclado saber qué elemento está activo. Se aplica típicamente a enlaces, botones, campos de formulario (`<input>`, `<textarea>`, `<select>`) y cualquier elemento al que se le haya añadido el atributo `tabindex`. Los navegadores suelen aplicar un contorno (outline) por defecto a los elementos enfocados.
   - **Ejemplo:**

```css
input:focus {
  border-color: blue;
  box-shadow: 0 0 5px rgba(0, 0, 255, 0.5); /* Un resaltado más visible */
  outline: none; /* A menudo se quita el outline por defecto si se reemplaza */
}
```

```
a:focus {
  outline: 2px solid orange; /* Estilo de foco personalizado */
}
```

- **¡Cuidado!** Nunca elimines el `outline` del estado `:focus` sin proporcionar una alternativa visual clara de enfoque. Hacerlo dificulta o imposibilita la navegación por teclado.

6. `:focus-within`
   - **Selecciona:** Un elemento si él mismo tiene `:focus` o si **alguno de sus elementos descendientes** tiene `:focus`.
   - **Uso:** Muy potente para estilizar contenedores basados en el foco de sus hijos. Por ejemplo, resaltar toda una sección de formulario (`<fieldset>` o `<div>`) cuando uno de sus campos de entrada (`<input>`) recibe el foco.
   - **Ejemplo:**

```
.form-group:focus-within {
  background-color: #f0f8ff; /* Resalta el grupo entero */
  border-left: 3px solid blue;
}
```

7. `:focus-visible`
   - **Selecciona:** Un elemento que tiene foco, pero solo cuando el navegador determina (mediante heurísticas internas) que mostrar un indicador de foco es útil para el usuario. Generalmente, esto significa que el indicador de foco se muestra durante la navegación por teclado (Tab), pero puede ocultarse si el elemento recibió foco mediante un clic del ratón (depende del elemento y del navegador).
   - **Uso:** Ayuda a resolver la queja común de los "anillos de foco feos" que aparecen al hacer clic en botones, sin sacrificar la accesibilidad del teclado. Permite aplicar estilos de foco visibles principalmente para usuarios de teclado.
   - **Ejemplo:**

```
/* Estilo de foco estándar (puede aparecer con clic) */
button:focus {
   outline: none;
}
/* Estilo de foco que SÓLO debería aparecer con navegación por teclado */
button:focus-visible {
   outline: 3px solid dodgerblue;
}
```

8. `:target`
   - **Selecciona:** El elemento único (si existe) cuyo `id` coincide con el fragmento de la URL actual (la parte después del `#`).
   - **Uso:** Útil para resaltar la sección de una página a la que se ha navegado mediante un enlace interno (ej. `pagina.html#seccion2` seleccionará el elemento con `id="seccion2"`). También se

usa creativamente para crear elementos como modales, pestañas o acordeones solo con CSS.

- **Ejemplo:**

```
<a href="#info-adicional">Mostrar más info</a>
<section id="info-adicional">Contenido...</section>
```

```
#info-adicional {
  display: none; /* Oculto por defecto */
}
#info-adicional:target {
  display: block; /* Se muestra cuando es el target */
  background-color: lightyellow;
}
```

9. **Pseudoclases de Formularios:** Hay varias pseudoclases específicas para elementos de formulario:
   - `:enabled` / `:disabled`: Seleccionan elementos de interfaz (inputs, botones, etc.) que están habilitados o deshabilitados (tienen el atributo `disabled`).

     ```
     input:disabled { background-color: #eee; cursor: not-allowed; }
     ```

   - `:checked`: Selecciona checkboxes (`<input type="checkbox">`) o radio buttons (`<input type="radio">`) que están marcados. También aplica a `<option>` en un `<select multiple>`. Muy útil para estilizar controles personalizados o crear interacciones basadas en el estado marcado.

     ```
     input[type="checkbox"]:checked + label { font-weight: bold; }
     ```

   - `:indeterminate`: Selecciona checkboxes/radio buttons en estado indeterminado (generalmente establecido vía JavaScript) o elementos `<progress>` sin valor.
   - `:required` / `:optional`: Seleccionan campos de formulario marcados con el atributo `required` o los que no lo tienen.

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

   - `:valid` / `:invalid`: Seleccionan campos de formulario cuyo contenido actual es válido o inválido según las reglas de validación HTML5 (p.ej., `type="email"`, `pattern`, `min`, `max`).

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

   - `:in-range` / `:out-of-range`: Para inputs numéricos (`type="number"`, `type="range"`) con atributos `min` y `max`, selecciona si el valor está dentro o fuera del rango permitido.

- `:read-only` / `:read-write`: Selecciona elementos que son o no editables por el usuario (`readonly` atributo, `contenteditable`).

Orden de las Pseudoclases (LVHA / LVFHA):

Debido a cómo funciona la especificidad y la cascada (que veremos más adelante), el orden en que defines ciertas pseudoclases de enlace y acción en tu CSS puede importar. Una regla común es **L**ink-**V**isited-**H**over-**A**ctive (LoVe HAte) o incluyendo Focus: **L**ink-**V**isited-**F**ocus-**H**over-**A**ctive.

```
a:link { color: blue; }      /* 1. Sin visitar */
a:visited { color: purple; }  /* 2. Visitado */
a:focus { outline: thin solid orange; } /* 3. En foco */
a:hover { color: green; }    /* 4. Al pasar el ratón */
a:active { color: red; }     /* 5. Mientras se hace clic */
```

Si `:hover` viniera antes que `:visited`, el efecto hover no funcionaría en enlaces ya visitados porque la regla `:visited` (más específica o posterior en el código) la sobrescribiría. El orden asegura que los estados más específicos o activos tengan prioridad visual.
