Extends y cómo evitar el exceso de especificidad

¿Qué Es el Extend de CSS y Cómo Puede Afectar Tu Código?

Una funcionalidad que siempre considero es el `@extend` de CSS, que proporcionado por SASS, nos permite compartir un conjunto de propiedades CSS de un selector a otro. Es como decirle al compilador “Oye, quiero que este conjunto de estilos se aplique también aquí”. Veamos cómo esto puede impactar en nuestro código.

Compartiendo estilos de manera inteligente

Imagina que tienes varios botones con un estilo básico común, pero cada uno tiene sus particularidades. En lugar de repetir el mismo código una y otra vez, con el `@extend` puedes tener un estilo base y extenderlo en los selectores específicos. Aquí un ejemplo práctico:

```
.error {
    color: red;
}

.error-in-form {
    @extend .error;
    background-color: #ffcccc;
}

/* ... much more code ... */
```

```
.alert {
    @extend .error;
    border: 1px solid red;
}
```

Si un día quisiera cambiar mi estilo `.error`, debo ser consciente de que también podría estar alterando `.error-in-form` y `.alert`, incluso si están en diferentes secciones de mi código. Esto se complica aún más si extiendes selectores complejos o ya extendidos previamente.

La idea detrás de `@extend` es promover un código DRY («Don’t Repeat Yourself»), pero como podéis ver, su uso debe ser con atención y estrategia, o de lo contrario podríamos acabar escribiendo CSS que es difícil de mantener y depurar a largo plazo.

Los Peligros de un Extend Mal Implementado

He visto de primera mano cómo un `@extend` mal usado puede plantear serios problemas. A veces, parece una solución rápida y limpia: reutiliza estilos con facilidad y evita duplicaciones innecesarias. Pero, amigos y amigas, hay una linea muy fina entre el ahorro de código y un auténtico caos en nuestras hojas de estilo.

Para ponernos en contexto, `@extend` en SASS es esa funcionalidad que, con una breve línea, permite que una selección herede los estilos de otra. Ideal, ¿no? El problema surge cuando se abusa de esta característica, o se utiliza sin una estrategia clara. Un extend mal implementado puede llevar a una cascada de herencia que haga que las clases se vuelvan extremadamente específicas y difíciles de sobrescribir.

Imagina que tienes un selector `.btn` con un conjunto de

estilos básicos para los botones. En algún momento decides que los botones de error `.btn-error` deberían extender los estilos de `.btn`. Pero luego, si decides agregar otra clase que modifique ligeramente estos estilos, digamos `.btn-small`, puedes terminar con una especificidad que no esperabas. Aquí un ejemplo:

```
%btn-base {
padding: 10px 15px;
border: none;
cursor: pointer;
}

.btn-primary {
@extend %btn-base;
background-color: blue;
color: white;
}
```

Con este método, solo la clase concreta `.btn-primary` va a aparecer en el CSS, manteniendo la especificidad en un nivel ideal. Además, evitamos que selectores innecesarios inflen el tamaño de nuestro código.

Otra buena práctica es evitar extender selectores complejos o compuestos. Cuando tengo que trabajar con estos, opto por descomponerlos en partes más pequeñas y aplicar `@extend` solo en aquellos lugares donde realmente es beneficioso. Un error común es extender una cadena de selectores como `.navbar .nav-link.active` que aumenta la especificidad a niveles altos. Preferible sería algo así:

```
.btn {
padding: 10px 20px;
```

```
border: none;
border-radius: 5px;
background-color: blue;
color: white;
}
```

Y quieres un botón de éxito que herede las mismas propiedades básicas pero con un color diferente: