Microinteracciones Zero-JS y Accesibilidad WCAG

Estantería: DevSecOps e Infraestructura

Subtema: DevSecOps

mercedev.es — 2026-06-01 | Epic 7 - Fase 2

1. El Desafío (Síntoma)

Enriquecer la estética visual (UI/UX) del ecosistema, implementando microinteracciones modernas (como elevación de tarjetas y transiciones en botones) y ampliando la paleta de colores corporativa. El reto principal consistía en lograr este enriquecimiento sin introducir librerías de JavaScript que penalizaran el Tiempo de Bloqueo Total (TBT) y garantizando simultáneamente el cumplimiento estricto de las directrices de accesibilidad y contraste WAI-ARIA (WCAG AA/AAA).

2. La Maniobra (Lógica)

Se adoptó una política estricta de *Zero-JS* para la capa de presentación. Todas las animaciones y microinteracciones se delegaron íntegramente a transiciones nativas de CSS y variables SASS (`_variables.scss`).

En paralelo, se realizó un ajuste quirúrgico de la paleta de colores. Por ejemplo, al implementar el tono malva (Homenaje) en los avisos de la tienda, se validó matemáticamente el ratio de contraste entre el texto (`#5b21b6`) y el fondo translúcido (`rgba(139, 92, 246, 0.1)`), obteniendo un ratio superior a 7:1, lo que satisface el nivel más estricto (AAA). En los botones de la portada, se rediseñó el estado `:hover` aplicando una inversión semántica (Fondo claro / Texto oscuro) para asegurar un contraste visual superior a 15:1.

3. El Aprendizaje/Deuda Técnica

El engaño del color primario: Es un error común de diseño utilizar el color primario de la marca (a menudo tonos vibrantes como naranjas o malvas claros) como fondo de botón con texto blanco. Estos colores vibrantes rara vez superan el ratio 4.5:1 exigido por Lighthouse. La solución arquitectónica es derivar siempre variables específicas (`$color-regular` o `$color-homage-dark`) para asegurar la legibilidad en bloques sólidos interactivos.

El hilo principal es sagrado: Delegar las animaciones de interfaz al CSS nativo no solo reduce el peso de transferencia (cero dependencias externas), sino que permite que el navegador optimice los cálculos gráficos mediante la Unidad de Procesamiento Gráfico (GPU), manteniendo el hilo principal libre de bloqueos y conservando el TBT en unos perfectos 0ms.

💡 En resumen (Merci Explica):

El "hilo principal" del navegador funciona como el único camarero de un restaurante abarrotado. Pedirle que haga malabares (animaciones complejas en JavaScript) mientras atiende mesas colapsará el servicio. Al delegar estas transiciones a CSS puro, contratamos a un malabarista independiente (la tarjeta gráfica del dispositivo). La interfaz luce dinámica y premium, pero el camarero sigue 100% libre para procesar interacciones críticas al instante.

🔗 Lecturas Recomendadas

- Resolución del desbordamiento en bloques de código CSS
- La guerra de la especificidad CSS