Guía Académica de Buenas Prácticas en Desarrollo Frontend

Enfocada en TypeScript y HTML Semántico

Autor: Lian Venegas
Instituto IACC - Agosto 2025

Introducción

Esta guía aborda las buenas prácticas fundamentales para el desarrollo frontend moderno, con especial enfoque en TypeScript y HTML semántico. Se incluyen un diagnóstico de errores comunes, una propuesta de prácticas recomendadas y una orientación metodológica para construir proyectos sostenibles, accesibles y mantenibles desde un enfoque académico.

Diagnóstico de errores comunes en frontend

- Uso excesivo de variables globales y ausencia de tipado estático.
- Duplicación de código y funciones con múltiples responsabilidades.
- Falta de semántica en etiquetas HTML (uso de para todo).
- Ausencia de validaciones y manejo de errores en formularios.
- Organización desordenada de archivos sin modularización.
- Carencia de estándares de estilo y ausencia de linters (ESLint, Prettier).
- Poca atención a la accesibilidad: sin etiquetas ARIA ni cumplimiento de WCAG.
- Mezcla de lógica de negocio en componentes de presentación.

Guía de Buenas Prácticas

- Aplicar TypeScript para tipado estático y detección temprana de errores.
- Adoptar arquitectura modular basada en componentes (React, Angular o Vue).
- Usar etiquetas HTML semánticas (, , ,).
- Separar lógica de negocio y lógica de presentación.
- Aplicar principios SOLID y patrones de diseño (Observer, Factory).
- Implementar linters y formateadores (ESLint, Prettier) de forma automatizada.
- Cumplir las pautas de accesibilidad WCAG 2.1, usando roles ARIA cuando sea necesario.
- Mantener documentación clara, comentarios útiles y convenciones de nombres consistentes.
- Incluir pruebas unitarias y de integración (Jest, React Testing Library).
- Versionar el proyecto con Git y aplicar control de ramas (Git Flow).

Metodología para desarrollar un buen proyecto frontend

1. 1. Recolección de requisitos: documentar objetivos, usuarios y accesibilidad requerida.
2. 2. Diseño de arquitectura: definir estructura de carpetas, patrones de componentes y flujos de datos.
3. 3. Configuración del entorno: instalar dependencias (Node.js, TypeScript, ESLint, Prettier).
4. 4. Desarrollo iterativo con SCRUM: organizar sprints cortos con entregables funcionales.
5. 5. Implementación de componentes: empezar con versión mínima viable, luego refactorizar.
6. 6. Pruebas continuas: validar accesibilidad con WAVE/Lighthouse y calidad con ESLint.
7. 7. Despliegue controlado: usar CI/CD, mantener entornos de prueba y producción separados.
8. 8. Documentación y capacitación: entregar manual técnico y guías de estilo a futuros equipos.

Conclusión

La aplicación sistemática de buenas prácticas en TypeScript y HTML permite crear proyectos más accesibles, legibles y sostenibles. Con una correcta planificación metodológica, se logra un impacto positivo tanto en la experiencia del usuario como en la eficiencia de los equipos de desarrollo.