TALLER 1
MÓDULO 3

Programación Avanzada en Java Script

INTEGRANTES:

Víctor Ramírez
Matías Espinoza
Karina Hidalgo

Informe de Investigación y Análisis Crítico

Análisis de la Incorporación de JavaScript en el Proyecto del Hospital Raccoon City

1. Generalidades del Lenguaje JavaScript

Historia y Evolución:

JavaScript fue creado en 1995 por Brendan Eich para el navegador Netscape con el fin de agregar interactividad a las páginas web. Desde entonces ha evolucionado hasta convertirse en uno de los lenguajes fundamentales del desarrollo web, abarcando tanto frontend como backend gracias a entornos como Node.js.

Uso en Navegadores Web:

JavaScript permite crear interacciones en tiempo real, manipular el DOM y gestionar eventos. En el proyecto del **Hospital Raccoon City**, esto es esencial para responder rápidamente a acciones del usuario como el llenado de formularios de emergencia, visualización de datos del equipo médico o simulación de alertas sanitarias.

Entornos Virtuales de Ejecución:

Además del navegador, JavaScript se ejecuta en el servidor mediante Node.js. Esto permitirá en futuras fases del hospital centralizar registros de pacientes, gestionar alertas o monitorear la red de clínicas.

Comparación con Otros Lenguajes:

- Es de **tipado dinámico**, a diferencia de lenguajes como Java.
- Es **interpretado**, lo que permite ver resultados inmediatos sin compilación previa.
- Soporta múltiples paradigmas: funcional, orientado a objetos e imperativo.

Fortalezas:

- Asincronía: permite mostrar notificaciones, cargar datos o validar formularios sin recargar la página.
- Amplio ecosistema: librerías como React, y herramientas como Vite o Webpack optimizan el desarrollo.

Debilidades:

- El tipado débil puede generar errores difíciles de rastrear en proyectos grandes.
- Sin una arquitectura clara, el código puede volverse difícil de mantener.

JavaScript como Lenguaje Asíncrono:

Utilizamos **async/await** para integrar futuras APIs (como reportes de incidentes o actualizaciones de estado clínico en tiempo real), lo cual mejora la experiencia de usuario.

2. Evolución del Lenguaje JavaScript y el Estándar ECMAScript

Lenguaje Interpretado vs. Compilado:

JavaScript es interpretado por el navegador en tiempo real, ideal para desarrollo rápido e interactivo. Los lenguajes compilados como Java requieren un paso previo, menos ágil para la web.

Evolución de ECMAScript:

- **ES5 (2009):** Introducción de **JSON**, modo estricto y mejoras en el manejo de arrays.
- **ES6 (2015):** Se incorporan **let**, **const**, arrow functions, clases, promesas y módulos.
- **ES9 (2018):** Operador spread/rest y **Promise.finally()** para mejorar la asincronía.

JavaScript vs. ECMAScript:

ECMAScript define los estándares que JavaScript implementa. Es como la "constitución" que los navegadores respetan al ejecutar JS.

TypeScript:

Un superconjunto de JavaScript con tipado estático. Ideal para proyectos como Raccoon City donde queremos minimizar errores lógicos y mejorar la escalabilidad.

Ventajas de TypeScript:

- Más seguridad al escribir código.
- Detecta errores en tiempo de desarrollo.
- Mejora la colaboración en equipos.

Desventajas:

- Necesita una etapa de compilación.
- Requiere configuración adicional.
- Puede sentirse innecesario en proyectos más pequeños.

3. Análisis Crítico: ¿Debemos integrar JavaScript Avanzado o TypeScript en el Hospital Raccoon City?

Ventajas:

- Mejora el mantenimiento y comprensión del código del hospital.
- Permite integrar funcionalidades complejas como sistemas de alertas, validación de usuarios y almacenamiento de formularios.
- Favorece futuras integraciones con backend en Node.js o APIs REST para reportes y monitoreo.

Desventajas:

- El equipo debe adquirir experiencia en TypeScript.
- Se requiere una infraestructura de desarrollo (compilador, bundler, etc.).

Conclusión:

La integración de JavaScript moderno o TypeScript es muy recomendable. Nos permite construir un sistema robusto y preparado para escalar: desde una página informativa hasta una plataforma completa de atención sanitaria online. La seguridad del código es crítica en un proyecto como este, donde los datos de pacientes, profesionales y emergencias están involucrados.

Propuesta Técnica (Extra)

Para introducir TypeScript de forma progresiva en el Hospital Raccoon City, se sugiere:

- Crear un sistema de validación de formularios en TS para registrar pacientes.
- Modularizar las funcionalidades JS en módulos separados: **contacto.js**, **equipo.js**, etc.
- Reescribir la lógica de navegación y mensajes interactivos con tipado fuerte.
- Integrar Webpack o Vite para compilar TS y organizar la estructura.
- Utilizar un archivo **tsconfig.json** para manejar la configuración y escalar según se necesite.

Esta migración progresiva será clave para asegurar que el Hospital pueda evolucionar tecnológicamente sin perder control del código.