Taller: Análisis de la Incorporación de JavaScript en el Proyecto del Hospital

1. Informe de Investigación: Generalidades del Lenguaje JavaScript

JavaScript fue creado por Brendan Eich en 1995, se desarrolló para que las páginas web de ese entonces sean interactivas y dinámicas, Eich desarrolló JavaScript en solo 10 días, con el objetivo de ser un lenguaje ligero y accesible para los diseñadores y desarrolladores de páginas web. La importancia que tiene hoy en día JavaScript en el desarrollo web, es fundamental gracias a los frameworks y bibliotecas, como los entornos de ejecución, este tipo de lenguaje se usa tanto en el lado del cliente como en el servidor, permitiendo crear aplicaciones completas, su facilidad de uso y al estar integrado en todos los navegadores modernos ha hecho de JavaScript sea el lenguaje para el desarrollo web.

El rol principal de JavaScript en los navegadores permitir que las páginas web respondan dinámicamente las acciones cuando el usuario interactúa con la pagina sin tener que estar recargando la página cada vez que el usuario haga algo. El rol de JavaScript en los navegadores es la interactividad, la manipulación del DOM, la gestión de eventos, la comunicación asíncrona. JavaScript se ejecuta desde el navegador del lado del cliente, los navegadores de hoy en día incluyen motores que interpretan y ejecutan el código.

JavaScript inicialmente fue hecho para ejecutarse en los navegadores web con el tiempo esto fue cambiando y hoy en día se ocupa para aplicaciones de servidor, desarrollo móvil, aplicaciones de escritorio entre otros

Diferencias entre JavaScript y otros lenguajes

Lenguaje	Propósito Principal	Paradigmas	Tipado	Gestión de Memoria
JavaScript	Desarrollo web, interactividad, backend	Funcional, prototípico	Dinámico	Automática
Python	Ciencia de datos, backend, scripting	Orientación a Objetos (clases), funcional	Dinámico	Automática
Java	Aplicaciones empresariales, Android	Orientación a Objetos (clases)	Estático	Automática
C++	Sistemas de alto rendimiento	Orientación a Objetos, procedural, de bajo nivel	Estático	Manual
Ruby	Backend (Rails), scripting	Orientación a Objetos (clases), funcional	Dinámico	Automática
TypeScript	Desarrollo web escalable	Orientación a Objetos, funcional (con tipado)	Estático	Automática

Fortalezas y debilidades de JavaScript

Ventajas	Limitaciones
Alta interactividad y dinamismo	Vulnerabilidades de seguridad
Ejecución en el cliente, rápida respuesta	Tipado débil y dinámico
Uso en frontend y backend (Node.js)	Inconsistencias entre navegadores
Amplio ecosistema y comunidad	Limitado rendimiento en operaciones intensivas
Soporte de asincronía (async/await)	Manejo de errores y excepciones limitado
Compatible con todos los navegadores	Complejidad en la gestión de asincronía

JavaScript es asíncrono porque permite realizar tareas que requieren tiempo sin bloquear la aplicación, existen los callbacks que son funciones que se ejecutan cuando finaliza la operación asíncrona, las promesas representan noperaciones asíncronas con estadosde pedniete, resuelta o rechazada, Async/await es una sintaxis para escribir código asíncrono de forma sincrona

2. Evolución del Lenguaje JavaScript y el Estándar ECMAScript

Lenguaje Interpretado vs. Compilado:

Un lenguaje interpretado el código se ejecuta en el momento de la interpretación, el lenguaje compilado tiene dos pasos, primero convierte el código en un archivo ejecutable y luego se usa ese archivo

Nombre	Lenguaje Compilado	Lenguaje Interpretado	JavaScript
Ejecución	Compila a máquina antes de ejecutar.	Ejecuta línea por línea.	Interpretado con compilación JIT para velocidad.
Ventajas	Rápido y eficiente en ejecución.	Flexible, ideal para cambios rápidos.	Interactividad en tiempo real y alto rendimiento.
Desventajas	Menos flexible, requiere compilación previa.	Más lento que compilado.	Menos eficaz en tareas intensivas, pero JIT ayuda.
Ejemplos	C, C++, Java	Python, Ruby, PHP	JavaScript

Evolución del Estándar ECMAScript

Versión	Año	Principales Mejoras
ES3	1999	- Funciones try/catch para manejo de excepciones. - Mejoras en la manipulación de cadenas, expresiones regulares y arrays.
ES4	No lanzada	- Especificación que nunca fue completada, pero contenía propuestas como clases, módulos, y soporte para tipos estáticos.
ES5	2009	- **strict mode** para mayor seguridad y mejores prácticas. - Métodos de arreglos como for Each, map, filter, reduce. - Object.create(), Object.defineProperty(). - Mejora en la definición de propiedades.
ES6	2015	- Introducen **let** y **const** para declarar variables. - **Clases** y **extends** para herencia. - **Arrow functions** (() => {}). - **Promesas** para manejo de asincronía. - **Template literals** (comillas invertidas `). - **Desestructuración** de objetos y arreglos.
ES7	2016	- **Array.prototype.includes()**: Método para comprobar si un arreglo contiene un valor. - **Exponentiation operator** (**) para exponenciación.
ES8	2017	- **async/await** para manejo asíncrono más sencillo. - **Object.entries()** y **Object.values()**. - Mejoras en cadenas con **String.prototype.padStart()** y **padEnd()**. - **SharedArrayBuffer** para memoria compartida en operaciones concurrentes.
ES9	2018	- **Rest/Spread** en objetos y arreglos. - **Promise.prototype.finally()**: Método para realizar acciones al finalizar una promesa.

JavaScript vs. ECMAScript

ECMAScript es la especificación, JavaScript es la implementación

Nombre	JavaScript	ECMAScript
Definición	Lenguaje de programación para web y servidores.	Estándar que define las especificaciones del lenguaje.
Relación	Es una implementación de ECMAScript.	Define las reglas que JavaScript sigue.
Propósito	Crear aplicaciones web dinámicas e interactivas.	Establece cómo debe comportarse el lenguaje.
Implementaciones	Motores como V8 y SpiderMonkey lo ejecutan.	Especifica el comportamiento, no la implementación.

Influencia en JS	Adopta nuevas características de ECMAScript.	Define las nuevas características de JavaScript.

TypeScript y sus Características

TypeScript es una alternativa a JavaScript por su tipado estático y mejoras en la organización y seguridad del código.

Nombre	TypeScript
Definición	TypeScript es un superset de JavaScript que añade tipado estático y otras características avanzadas.
Tipado Estático	Permite declarar tipos para variables, funciones y objetos, lo que ayuda a detectar errores en tiempo de compilación.
Compilación	El código TypeScript se compila a JavaScript, lo que permite ejecutarlo en cualquier entorno que soporte JavaScript.
Compatibilidad	Totalmente compatible con JavaScript. Cualquier código JavaScript válido es también código TypeScript válido.
Soporte para Clases	Soporta clases, interfaces y módulos, con una sintaxis más moderna que JavaScript, basada en ES6+.
Herramientas de Desarrollo	Mejora el desarrollo con autocompletado, verificación de tipos, y navegación más fácil en editores como VSCode.
Escalabilidad	Más adecuado para proyectos grandes, ya que permite una mayor organización y mantenimiento gracias a su tipado y características de programación orientada a objetos.
Ventajas	Ayuda a evitar errores comunes de JavaScript, mejora la mantenibilidad y facilita el desarrollo en equipos grandes.
Alternativa a JavaScript	TypeScript es una alternativa a JavaScript porque proporciona una forma más segura y estructurada de escribir código, especialmente en aplicaciones grandes y complejas.

Ventajas y Desventajas de TypeScript

TypeScript es útil en proyectos grandes como con un hospital, su estructura y ayuda a evitar errores, mejora la estructura, facilita el trabajo es compatible con JavaScript y es ideal para proyectos grandes.

Nombre	Ventajas de TypeScript	Desventajas de TypeScript
Seguridad de Tipos	Ayuda a evitar errores comunes al detectar problemas en tiempo de compilación gracias al tipado estático.	Requiere declarar y gestionar tipos.

Mantenibilidad	Mejora la estructura del código, ideal para proyectos grandes y equipos de desarrollo, facilitando la escalabilidad.	Puede ser más difícil de aprender para desarrolladores nuevos o sin experiencia
Desarrollo Colaborativo	Facilita el trabajo en equipo al proporcionar autocompletado y documentación más precisa a través de los tipos.	Requiere un paso adicional de compilación antes de ejecutar el código, lo que aumenta el tiempo de desarrollo.
Herramientas	Integración excelente con editores como VSCode, con funciones como autocompletado, depuración y refactorización.	Necesita configuraciones adicionales en el entorno de desarrollo.
Compatibilidad	Compatible con JavaScript y bibliotecas existentes	La transición de un proyecto JavaScript a TypeScript puede ser complejo.
Escalabilidad	Ideal para proyectos grandes, como un sistema hospitalario, donde la claridad y la organización del código son cruciales.	Puede ser innecesario para proyectos pequeños donde JavaScript simple es suficiente.

3. Análisis de la Pertinencia de Integrar JavaScript Avanzado o TypeScript en el Proyecto

Nombre	Ventajas	Desventajas
JavaScript Avanzado	- Mejora la interactividad y dinamismo.	- Complejidad en la gestión de asincronía
	- Ejecución rápida desde el cliente, lo que mejora la respuesta del sitio web.	- Inconsistencias entre navegadores pueden generar problemas.
TypeScript	- Tipado estático que ayuda a detectar errores en tiempo de compilación.	- Requiere aprendizaje adicional si la persona no lo conoce.
	- Mejora la organización y escalabilidad del código	- Aumento del tiempo de desarrollo debido a la compilación y configuración adicional.
	- Facilita el trabajo en equipo	- La transición a TypeScript en proyectos grandes puede ser costosa y compleja.

Conclusión: yo pienso que si ayudara mucho Integrar uno de los dos al proyecto es beneficioso para mejorar la interactividad de la página, aunque su implementación puede ser más complicada, Será difícil será complejo presiento que el proyecto quedara bien.