Taller 1:
Análisis de la Incorporación de JavaScript en el Proyecto del Hospital

Integrantes:

- María Fernanda Avello
- Cristian Soto
- Pablo Soto

Fecha: 6/11/2024

Sección 1

I. Generalidades del Lenguaje JavaScript.

1.Historia de JavaScript: Describe cómo y por qué fue creado JavaScript, y su importancia en el desarrollo web

JavaScript fue creado en 1995 por Brendan Eich en Netscape con el objetivo de hacer las páginas web interactivas. Hasta ese momento, las páginas web eran estáticas y sólo mostraban información sin responder al usuario. Eich desarrolló JavaScript como un lenguaje simple que pudiera ejecutarse directamente en el navegador, lo que permitía agregar funcionalidades básicas sin instalar plugins.

Dentro de su importancia en el desarrollo web se destaca:
Interactividad: JavaScript permitió que los sitios web fueran interactivos, respondiendo en tiempo real a las acciones del usuario.
Funcionalidad en el Cliente: Ejecutándose en el navegador, JavaScript libera al servidor de tareas sencillas, optimizando la experiencia del usuario.
Fundamento del Desarrollo Web Moderno: Hoy en día, JavaScript es esencial para el desarrollo web, formando la base de frameworks modernos como React, Angular y Vue.

2.Uso de JavaScript en Navegadores Web: Explica el rol de JavaScript en los navegadores y cómo se ejecuta en el lado del cliente.

JavaScript es fundamental en el desarrollo web, ya que permite crear aplicaciones interactivas y dinámicas en el navegador. Su rol principal en los navegadores es mejorar la experiencia del usuario mediante la manipulación del Document Object Model (DOM), respondiendo a eventos del usuario y facilitando la comunicación asincrónica con servidores sin recargar la página.

Cuando un navegador carga una página web, también descarga y ejecuta el código JavaScript en el dispositivo del usuario. El motor JavaScript del navegador (como V8 en Chrome o SpiderMonkey en Firefox) interpreta y ejecuta el código en una "sandbox" segura, protegiendo el dispositivo y limitando el acceso a recursos sensibles. JavaScript maneja eventos del usuario (como clics o escritura) mediante un "bucle de eventos" que permite la ejecución eficiente y evita bloqueos. Esto contribuye a que las aplicaciones web sean fluidas y den una respuesta rápidamente, mejorando la experiencia del usuario.

3.Entornos Virtuales de JavaScript: Investiga los diferentes entornos donde se puede ejecutar JavaScript (navegador, Node.js, etc.).

Javascript puede ejecutarse en navegadores, servidores usando Node.js, por lo que permite usarse para aplicaciones backend, servidores REST o en sistemas IoT.
En el lado del servidor, puede referenciarse al uso de lenguaje de codificación de lógica, en tal caso puede acceder a la base de datos, hacer operaciones lógicas y responder a eventos que son desencadenados por el sistema operativo. La mayor ventaja es que admite un alto nivel de personalización de respuesta del website según sus requisitos, derechos de acceso y solicitudes de información desde la web.

4.Diferencias entre JavaScript y otros lenguajes: Compara JavaScript con otros lenguajes de programación en cuanto a propósito, uso y paradigmas soportados.

Propósito: JavaScript fue creado para el desarrollo web, permitiendo interactividad en el navegador. Otros lenguajes como Python, Java y C++ tienen usos más amplios fuera del navegador, como inteligencia artificial (Python) y aplicaciones de servidor (Java).

Uso: JavaScript se usa principalmente en frontend, y también en backend con Node.js. Python y Ruby son más comunes en backend, mientras que Java y C++ se utilizan en aplicaciones de gran escala y sistemas de alto rendimiento.

Paradigmas: JavaScript soporta programación orientada a objetos, funcional e imperativa, aunque su orientación a objetos se basa en prototipos. Otros lenguajes como Java se basan en clases, mientras que Python permite una estructura clara y versátil.

En resumen, JavaScript es clave en la web, mientras que otros lenguajes suelen preferirse en aplicaciones de servidor, ciencia de datos y desarrollo de software a gran escala.

5.Fortalezas y debilidades de JavaScript: Analiza las principales ventajas y limitaciones del lenguaje en el desarrollo web.
Ventajas de JavaScript:

- Rápido y sencillo: Se ejecuta en el navegador, evitando retrasos del servidor, y es relativamente fácil de aprender.
- Popular y compatible: Es el lenguaje más usado en la web y funciona en cualquier navegador y página.
- Reducción de carga del servidor: Al ser client-side, disminuye la demanda del servidor.

- Interactividad y versatilidad: Permite crear interfaces interactivas y desarrollar aplicaciones completas tanto en frontend como en backend con Node.js.
- Actualizaciones constantes: Se actualiza regularmente, manteniéndose moderno y eficiente.

Desventajas de JavaScript:

- Seguridad client-side: Los errores pueden ser explotados por terceros.
- Compatibilidad entre navegadores: Aunque hoy en día es menor, es necesario verificar que el código funcione en todos los navegadores.

JavaScript es potente y flexible, ideal para el desarrollo web, aunque con algunas limitaciones en seguridad y compatibilidad.

6. JavaScript como lenguaje asíncrono: Explica por qué JavaScript es asíncrono y cómo maneja la asincronía (callbacks, promises, async/await).

JavaScript es un lenguaje de un solo hilo que ejecuta instrucciones una por una. En modo síncrono, esto puede bloquear el flujo cuando hay tareas lentas, como solicitudes de datos. Para evitar bloqueos, JavaScript usa asincronía, permitiendo que el código siga ejecutándose mientras espera resultados.

Formas de manejar la asincronía:

- Callbacks: Funciones que se ejecutan después de completar una tarea, pero pueden complicar el código.
- Promesas: Objetos que simplifican el manejo de tareas asíncronas, con métodos .then() y .catch().
- Async/Await: Sintaxis moderna que hace que el código asíncrono se vea más simple y ordenado.

Estas herramientas mejoran el rendimiento y facilitan el manejo de tareas asíncronas en JavaScript.

Algunas personas deciden que tratar con código asíncrono es demasiado complicado para trabajar, por lo que intentan hacer que todo sea sincrónico. Por ejemplo, en lugar de usar setTimeout, podría crear una función síncrona para no hacer nada durante un período de tiempo determinado:

```
const pause = duration => {
  const start = new Date().getTime()
  while (new Date().getTime() - start < duration) {}
}
```

II. Evolución del Lenguaje JavaScript y el Estándar ECMAScript

1. Lenguaje Interpretado vs. Compilado: Describe las diferencias clave entre estos dos tipos de lenguajes y cómo se relaciona con JavaScript.

La diferencia entre lenguajes interpretados y compilados está en cómo el código se convierte en ejecutable:

El Interpretado se ejecuta línea por línea en tiempo real (ej., JavaScript, Python), permitiendo ajustes rápidos pero a menor velocidad.

En cambio el compilado se convierte a código máquina antes de ejecutarse (ej., C++, Java), lo que aumenta la velocidad, aunque requiere un paso previo de compilación.

JavaScript es del tipo interpretado y se ejecuta directamente en el navegador. Los motores modernos como V8 también aplican técnicas de *just-in-time (JIT) compilation* para mejorar su rendimiento, combinando velocidad y flexibilidad.

2. Evolución del Estándar ECMAScript: Detalla la evolución de ECMAScript, desde ES3 hasta ES9, y menciona las principales mejoras introducidas en cada versión.

ECMAScript, el estándar de JavaScript, ha evolucionado constantemente para mejorar el lenguaje y satisfacer las demandas del desarrollo web moderno. Dentro de sus principales mejoras de cada versión desde ES3 hasta ES9:

- **ES3 (1999)**: Introdujo características esenciales como expresiones regulares, control de excepciones (*try/catch*), y el modo estricto de igualdad (=== y !==). Sentó las bases para el desarrollo de JavaScript.
- **ES5 (2009)**: Tras un largo periodo sin cambios, ES5 trajo *strict mode* para mejorar la seguridad y el rendimiento. Agregó métodos como .forEach, .map, .filter, .reduce, entre otros, para trabajar mejor con arrays, además de Object.defineProperty para un control preciso sobre los objetos.
- **ES6 (2015)**: Esta actualización fue una revolución para JavaScript. Introdujo let y const para declarar variables, arrow functions para una sintaxis de funciones más concisa, *template literals*, clases, módulos, y *promises* para mejorar el manejo de asincronía.

- **ES7 (2016)**: Fue una actualización menor, pero importante. Agregó el operador de exponenciación (**) y el método Array.prototype.includes, que facilita la búsqueda de elementos en un array.
- **ES8 (2017)**: Incorporó características útiles como async/await para un manejo más sencillo de promesas, Object.entries y Object.values para trabajar con objetos, y el *padding* de strings (padStart y padEnd).
- **ES9 (2018)**: Mejoró el rendimiento y la simplicidad de código con el operador de *rest/spread* en objetos, Promise.finally para limpieza de recursos al manejar promesas, y mejoras en el *asynchronous iteration*, como los generadores asíncronos.

3. JavaScript vs. ECMAScript: Explica la relación entre ambos y cómo el estándar ECMAScript influye en las implementaciones modernas de JavaScript.

ECMAScript es el estándar técnico que define cómo debería funcionar un lenguaje de scripting de propósito general. Fue establecido por la organización ECMA International. Su propósito es asegurar que las implementaciones de JavaScript y otros lenguajes basados en ECMAScript sigan un comportamiento consistente.

JavaScript es el lenguaje de programación que sigue las especificaciones de ECMAScript. Al ser un "dialecto" de ECMAScript, se adhiere a las reglas del estándar, pero también incluye sus propias características adicionales, como la manipulación del DOM (Document Object Model) en el navegador, que no es parte del estándar ECMAScript.

4. TypeScript y sus Características: Investiga qué es TypeScript, sus principales características y por qué es una alternativa a JavaScript.

TypeScript es un superset hecho por Microsoft el cual tiene una sintaxis muy intuitiva y que nos recuerda a otros lenguajes orientados a objetos. Agrega funcionalidades que extiende lo que haría JavaScript por sí sólo, tal cómo Types y Decorators.

TypeScript proporciona varias características potentes para el desarrollo web moderno que abordan algunas de las limitaciones de JavaScript. Estas características ofrecen una mejor experiencia para el desarrollador y una mejor organización del código. Entre ellas se incluyen:

- Tipado Estático
- Tipado Opcional
- Funciones ES6+
- Organización del Código
- Características de la Programación Orientada a Objetos (OOP)
- Sistema de Tipos Avanzado

- Compatibilidad con JavaScript

Brinda ventajas a Javascript como la detección de errores de tipos antes de la ejecución, un mejor autocompletado y refactorización en editores de código, y una documentación más clara para objetos y funciones.

5. Ventajas y Desventajas de TypeScript

Analiza las ventajas y desventajas de utilizar TypeScript en lugar de JavaScript en proyectos como el del hospital.

Algunas ventajas de usar TypeScript en lugar de JavaScript en el proyecto del hospital:

- Tipado estático: permitiendo detectar errores de manera temprana en lugar de en tiempo de ejecución. Esto permite al desarrollador especificar tipos de datos para variables, parámetros de funciones y propiedades de objetos, identificando errores de manera rápida y temprana y previniendo comportamientos inesperados
- Legibilidad mejorada del código: La declaración explícita de tipos mejora la legibilidad del código, facilitando la colaboración con otros estudiantes al momento de ir construyendo entre varios el código para el sitio web del hospital.
- Compatibilidad con estándares JavaScript: es compatible con características ECMAScript 6 y versiones posteriores de JavaScript, permitiendo aprovechar las últimas funcionalidades del lenguaje como clases, módulos y funciones de flecha, sin problemas de compatibilidad.

Algunas desventajas serían:

- Curva de aprendizaje: es necesario “aclimatarse” para usar el tipado estático, de manera que la mayoría de los desarrolladores ya están acostumbrados a usar JavaScript puro.
- Necesidad de compilación: requiere de una etapa de compilación antes de que el código se pueda ejecutar en un navegador o entorno Node.js.
- Documentación limitada: la comunidad está en crecimiento, por lo que podría haber menos documentación que para JavaScript, por lo que los alumnos podrían encontrar menos información de algunos problemas al momento que van construyendo el código.

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

Al analizar la conveniencia de integrar JavaScript avanzado o TypeScript en la página web para el proyecto del hospital, se deben considerar las ventajas y desafíos que traen estas tecnologías, especialmente para garantizar seguridad, rendimiento y facilidad de mantenimiento en un contexto que requiere alta fiabilidad.

Ventajas de Usar JavaScript Avanzado o TypeScript

- **Organización y Mantenimiento**: TypeScript añade tipado estático, lo cual ayuda a reducir errores y facilita el mantenimiento del código a medida que el proyecto crece. Esto es crucial en aplicaciones para hospitales, donde la precisión en la lógica es vital.
- **Mejora de Rendimiento**: Usar JavaScript avanzado permite optimizar el código con técnicas de asincronía avanzada (`async/await`) y modularización, mejorando el rendimiento. Esto es útil para manejar interacciones complejas, como agendar o reservar citas online.
- **Escalabilidad**: TypeScript facilita la expansión de funcionalidades al crear un sistema robusto, que en un futuro podría manejar tareas más complejas, como agregar una inteligencia artificial de atención al cliente, o también una funcionalidad de apoyo médico al paciente permitiría preguntarle al paciente los síntomas que tiene actualmente para aconsejarle si asistir a un especialista por COVID según lo que indica, o si podría ser otra enfermedad.
- **Facilidad de Desarrollo en Equipo**: TypeScript ayuda a que los desarrolladores trabajen en equipo de manera más ordenada, pues su tipado y estructura clara previenen conflictos y errores en el código colaborativo.

Desventajas o Posibles Dificultades

1. **Curva de Aprendizaje**: TypeScript puede requerir una mayor curva de aprendizaje, especialmente para desarrolladores sin experiencia en lenguajes tipados. Para un equipo de frontend sin mucha experiencia, puede ser complejo adaptarse y aprovechar sus ventajas.
2. **Aumento en el Tiempo de Desarrollo**: La configuración inicial y el uso de TypeScript pueden alargar el desarrollo, pues implica más planificación y pruebas. Si el proyecto tiene un plazo ajustado, esto puede ser un desafío.
3. **Configuración y Herramientas Adicionales**: Usar TypeScript implica una configuración adicional y, en ocasiones, integrar herramientas como Babel o Webpack. Esto puede añadir complejidad y requerir un entorno de desarrollo más avanzado.
   La comunidad de Typescript está creciendo, por lo que no hay suficiente información para algunos problemas que podrían tener los desarrolladores frontend novatos.

Usar Typescript actualmente es un gran desafío sin tanta documentación y en algunas ocasiones se necesitaría usar otras herramientas adicionales.

Conclusión

Si el proyecto de la página web del hospital requiere alta seguridad, confiabilidad y potencial de crecimiento, integrar TypeScript es recomendable, ya que proporciona un código más estructurado y manejable. Sin embargo, si el equipo tiene un plazo ajustado o falta experiencia, puede ser preferible empezar con JavaScript avanzado para ganar velocidad en el desarrollo y simplicidad.