Análisis de la Incorporación de JavaScript en el Proyecto del Hospital

Catherine Rebolledo - Javiera Allende - Felipe Pineda

1.- Generalidades del Lenguaje JavaScript

1.1- Historia de JavaScript

El primer motor de JavaScript fue creado por Brendan Eich en Netscape Communications Corporation, para el navegador web Netscape Navigator. El motor, denominado SpiderMonkey, está implementado en C. Desde entonces, ha sido actualizado (en JavaScript 1.5) para cumplir con el ECMA-262 edición 3.

Este es un lenguaje de programación que fue desarrollado inicialmente en 10 días y concebido como un lenguaje complementario a Java.[1]

1.2.- ¿Cómo y por qué fue creado?

Brendan Eich Trabajaba en Netscape Communications, uno de los primeros navegadores web populares, ellos querían agregar un lenguaje de scripting a su navegador para permitir a los desarrolladores web agregar interactividad a las páginas web [2]

1.3.- Importancia en el desarrollo

Es una herramienta fundamental para el desarrollo web, aún cuando actualmente siguen apareciendo muchos lenguajes, aprender javascript puede ser muy útil para quienes desean especializarse en esta área, la comunidad de javascript es muy activa, por ende, constantemente actualizan el lenguaje y las bibliotecas, esto abre a muchas oportunidades de empleo.[3]

---

[1] Fuente https://tinyurl.com/49beuuc4
[2] Fuente https://dev.to/maricarmendev/como-empezo-javascript-4dnf
[3] Fuente http://surl.li/iblinq

1.4- Uso de JavaScript en Navegadores Web:

La función de JavaScript es añadir características interactivas a las páginas web, aplicaciones de servidor, desarrollo de aplicaciones móviles y de escritorio, etc y se logra a través de la ejecución de código que crea animaciones, menús emergentes, botones, efectos de estilo dinámicos. Se ejecuta en el navegador web del usuario final.[4]

1.5 - Entornos Virtuales de JavaScript:

¿Qué es un entorno virtual? Un espacio aislado donde se puede ejecutar un script o código. Los navegadores de internet modernos se pueden considerar entornos virtuales (cada uno siendo un entorno diferente). Además existen entornos para ejecutar programas JS creados en la máquina local. Entre entornos virtuales pueden haber diferencias en el motor de ejecución, capacidades y limitaciones, rendimiento entre otras cosas.

¿Qué es un Motor de Ejecución? Es la herramienta que interpreta el código JavaScript y lo ejecuta. Al ejecutar un código con distintos motores, aunque el resultado pueda parecer el mismo, pueden haber diferencias en rendimiento o seguridad de los datos, ya que un motor puede hacer una interpretación diferente en direcciones de memoria, o algoritmos de procesamiento.[5] [6]

Navegadores Web			
Chrome	**Edge**	**Firefox**	**Safari**
Motor V8	Chakra	Spider Monkey	JavaScriptCore

Motores Locales		
NodeJS	**Bun**	**Deno**
Motor V8	JavaScriptCore	Motor V8

[4] Fuente https://aws.amazon.com/es/what-is/javascript/
[5] Fuente https://n9.cl/dxq9w
[6] Fuente https://onx.la/945d2

1.6- Diferencias entre JavaScript y otros lenguajes:.

JavaScript: Lenguaje Interpretado. Aunque en un principio estuvo orientado principalmente al desarrollo Web, actualmente es más multipropósito. Posee cierta facilidad para manejar procesos asíncronos y es de tipado dinámico Paradigmas soportados: Programación imperativa, Programación orientada a objetos (OOP), Programación funcional.

Python: Lenguaje Interpretado Multipropósito. Actualmente se orienta más a tareas I+D y Machine Learning. Sus paradigmas son los mismos de JS.

C: Lenguaje Compilado Multipropósito. Es un lenguaje de nivel medio-bajo. Posee un gran manejo sobre las direcciones de memoria. Por esta cualidad se utiliza en el desarrollo para sistemas de menor capacidad de cómputo o donde es importante obtener el máximo rendimiento del programa. Los paradigmas de este lenguaje son Imperativo y Funcional. C++ añade el paradigma de programación Orientada a Objetos.[7] [8]

1.7 - Fortalezas y debilidades de JavaScript:

Fortalezas:

- **Velocidad:** JavaScript tiende a ser muy rápido porque a menudo se ejecuta inmediatamente en el navegador.

- **Simplicidad:** La sintaxis de JavaScript está inspirada por Java y es relativamente sencillo de aprender comparado a otros lenguajes de programación populares como C++.|

- **Compatibilidad:** A diferencia de PHP u otros lenguajes scripting, JavaScript puede ser usado en cualquier página web. JavaScript puede ser usado en diferentes tipos de aplicaciones gracias al soporte en otros lenguajes como Pearl y PHP.

- **Interfaces sencillas:** JavaScript puede ser usado para crear características como arrastrar y soltar, y componentes tales como las diapositivas, lo cual mejora enormemente la interfaz de usuario y la experiencia del sitio.

Debilidades:

- **Vulnerabilidades de Seguridad:** Uno de los mayores problemas de JavaScript es su vulnerabilidad a los ataques de seguridad. Dado que

[7] Fuente https://goo.su/I8z5Otg
[8] Fuente https://developer.mozilla.org/es/docs/Learn/JavaScript/Objects

JavaScript se ejecuta en el navegador web, cualquier script puede ser manipulado por el usuario o, en el peor de los casos, por un atacante externo.

- **Problemas de Compatibilidad entre Navegadores:**Aunque JavaScript es compatible con la mayoría de los navegadores, la implementación de algunas características puede variar. Esto significa que el mismo código JavaScript puede comportarse de manera diferente en distintos navegadores o versiones de páginas web.

- **Complejidad de la Programación Asíncrona:** JavaScript está diseñado para manejar múltiples tareas de manera asíncrona, lo que permite que una aplicación continúe ejecutándose mientras espera que se completen tareas como las solicitudes a servidores externos. Sin embargo, la programación asíncrona en JavaScript puede ser difícil de manejar, especialmente para los desarrolladores principiantes.
[9] [10]

1.8 - JavaScript como lenguaje asíncrono:

JavaScript es asíncrono porque permite que un programa inicie una tarea de larga duración (Ej: Esperar la carga de algún recurso) y siga respondiendo a otros eventos mientras la tarea se ejecuta.

Callbacks

Una función de callback es una función que se pasa a otra función como un argumento, que luego se invoca dentro de la función externa para completar algún tipo de rutina o acción*. En JS se usa principalmente para ejecutar un proceso cuando una operación asíncrona termina. Aunque son simples, pueden generar problemas, siendo el más conocido el Callback Hell**.

Promesas

Es un objeto que representa el eventual resultado de una operación asíncrona ya sea terminada exitosamente o con algún error. Se configuran con dos callbacks para resolver la promesa con éxito o con fallo.

Async/await

---

[9] Fuente https://www.joystick.com.mx/historia-de-javascript-ventajas-y-desventajas-de-su-uso-y-para-que-es-usado-actualmente/

[10] Fuente https://www.dongee.com/tutoriales/javascript-ventajas-y-desventajas/

Es un azúcar sintáctico que permite manejar promesas de una forma más simple. La palabra clave async declara una función como asíncrona, mientras que await gestiona la resolución de una promesa de forma automática [11] [12]

2. Evolución del Lenguaje JavaScript y el Estándar ECMAScript

2.1- Lenguaje Interpretado vs. Compilado:

Lenguaje Interpretado: son aquellos que convierten su lenguaje al de la máquina a medida que ejecutan el código. Es decir, el lenguaje interpretado es el lenguaje que entendemos como programadores, basado en palabras clave, se traduce al lenguaje que entiende el procesador, caracterizado por tener valores '1' y '0'. En estos lenguajes interpretados, esto sucede a medida que el programa lee nuestro lenguaje de arriba abajo y de derecha a izquierda. Los lenguajes de programación como python, JavaScript y PHP son lenguajes interpretados.

Lenguaje Compilado: Los lenguajes de programación compilados que requieren que las instrucciones (código fuente del programa), sean traducidas, mediante un programa compilador, a un lenguaje que entienda la máquina (lenguaje máquina), con el fin de generar una versión ejecutable del programa. Ejemplo de lenguajes compilados son Pascal, C, C++, Cobol, Fortran, entre otros.[13] [14]

2.2 - Evolución del Estándar ECMAScript

ECMAScript 3 (1999)	soporte para expresiones regulares, gestión estructurada de excepciones y otras mejoras puntuales.
ECMAScript 5 (2009)	Se añadió el soporte nativo para JSON o los getters y setters para propiedades. Agrega el modo estricto (
ECMAScript 5.1 (2011)	alineaba el estándar de ECMA con el formato correspondiente de ISO (ISO/IEC 16262:2011).

[11] Fuente *https://developer.mozilla.org/es/docs/Glossary/Callback_function
[12] Fuente http://callbackhell.com/
[13] Fuente: **https://departamentos.colegiosansaturio.com/deptomatesweb/4ESO/informatica%20web/temas/Unidad_6/pagina1.html**

[14] Fuente **https://keepcoding.io/blog/lenguajes-programacion-interpretados-compilados/**

ECMAScript 6 (2015)	mejora de la sintaxis y actualización de la misma ya que trajo consigo los símbolos, las lambdas y tipos de datos que no existían en las versiones anteriores, así como también fueron mejoradas las estructuras iteración. Principales cambios - Funciones arrow - let, const
Actualidad	Han salido versiones más recientes de ECMAScript, pero se consideran actualización de ES6. Los cambios más destacables de estas actualizaciones son: - Gestión de promesas con async / await - Operadores rest/ spread

2.3 - JavaScript vs. ECMAScript:

ECMAScript es un estándar establecido por ECMA international que indica cómo debe estructurarse un lenguaje de Scripting. JavaScript es un lenguaje de programación que sigue este estándar.

El estándar utilizado actualmente es ECMA-262.[15]

2.4 - TypeScript y sus Características:

TypeScript es un lenguaje de programación que principalmente añade tipado estático, interfaces y decoradores a JavaScript.
Para ejecutar un script TS este debe ser transpirado (traducido) a JS. Dependiendo de la configuración del usuario este proceso puede ser más o menos estricto con el análisis del tipo de datos antes de traspasar el script a JS.

2.5 - Ventajas y Desventajas de TypeScript:

Ventajas:

- La principal ventaja de TypeScript es la prevención de algunos errores que podrían ocurrir en el tiempo de ejecución que en JS no serían detectados.
- Facilita la programación a gran escala, ya que otros programadores tienen conocimientos de las estructuras de los datos utilizados en funciones.
- Aunque JavaScript ES6 introdujo clases, TypeScript las extiende con características como modificadores de acceso (público, privado, protegido) y propiedades estáticas de clase.

[15] https://programacionymas.com/blog/diferencia-entre-javascript-y-ecmascript

Desventajas:

- ### Curva de Aprendizaje
  Para aquellos que vienen de JavaScript, aprender TypeScript puede representar un desafío adicional. La necesidad de comprender interfaces, tipos genéricos y otros conceptos puede ser abrumadora para principiantes.

- ### Tiempo de Compilación
  TypeScript debe ser compilado a JavaScript antes de que pueda ejecutarse en un navegador o entorno. Este proceso añade un paso adicional al flujo de trabajo de desarrollo y puede significar un tiempo de espera mayor en proyectos muy grandes.

- ### Integración y Adopción
  Incorporar TypeScript en un proyecto existente puede ser complejo, especialmente si el código base es extenso y ya está escrito en JavaScript puro. Además, hay bibliotecas o terceros que podrían no tener tipos definidos para TypeScript, implica un esfuerzo adicional para crearlos o encontrarlos. [16]

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

Al desarrollar un sitio web para un hospital, es importante equilibrar la funcionalidad, la accesibilidad y la simplicidad en la implementación de las tecnologías. Si bien el uso de JavaScript avanzado o TypeScript podría ofrecer ventajas en términos de organización y escalabilidad, también puede traer complejidades que podrían no ser necesarias para un proyecto con funcionalidades relativamente sencillas. A continuación, se analiza si es pertinente o no integrar estas tecnologías en el proyecto del Hospital Base San José Osorno.

Ventajas:

- **Mejor Gestión del Código:** JavaScript avanzado (ES6+) facilita la escritura de un código más limpio y fácil de mantener, mientras que TypeScript, al agregar tipado estático, ayuda a identificar errores antes de que ocurran, lo que aumenta la calidad del proyecto.

---

[16] Fuente: https://nelkodev.com/blog/cuando-usar-typescript-sobre-javascript-ventajas-y-desventajas/
https://typescriptutorial.com/es/ventajas-y-desventajas/

- **Escalabilidad:** Si el proyecto creciera con el tiempo, estas tecnologías podrían hacer que añadir nuevas funcionalidades, como interacciones más complejas, fuera más fácil y ordenado.
- **Interactividad Mejorada:** En el futuro, si se agregan más características dinámicas al sitio (como un sistema de reservas o un panel interactivo), JavaScript avanzado o TypeScript permitirían que esas interacciones sean más eficientes.
- **Mejoras de Accesibilidad:** Las cualidades de la programación dinámica, permitiría modificar elementos o el layout completo de la página según las necesidades de los usuarios. (Ej: Cargar la página en distintos idiomas, cambiar colores o disposición de botones para personas con discapacidad visual.
- **Facilita la modificación de datos**: A través de JS se permite el acceso a información presente en bases de datos o archivos. Esto permitiría cambiar la información desplegada en la página sin necesidad de actualizar el archivo html.

Desventajas:

- **Complejidad y Tiempo de Desarrollo:** Integrar estas tecnologías podría aumentar la carga de trabajo y retrasar el desarrollo, especialmente si el equipo no está familiarizado con ellas.

- **Curva de Aprendizaje:** TypeScript, en particular, requiere un tiempo de aprendizaje adicional, lo cual no es ideal en un proyecto donde la interactividad es limitada.

- **Sobrecarga Innecesaria:** Si un sitio no requiere funciones demasiado complejas, agregar estas tecnologías podría ser innecesario y complicar el proceso de desarrollo sin un beneficio claro.

Si bien JavaScript/TypeScript posee una curva de aprendizaje, la implementación de estas tecnologías permitiría mejorar la accesibilidad del sitio del hospital, además de facilitar un futuro desarrollo de esta página ya que permitiría un manejo más eficiente de la información.