Taller 1 - Módulo 3

Análisis de la Incorporación de JavaScript en el Proyecto del Hospital

Fecha: 06 de noviembre de 2024

Integrantes del grupo
Edwin Maureira
Felipe Rodríguez
Carla García.

Curso: Especialización Front-end 2024 Grupo 1

1. Informe de Investigación: Generalidades del Lenguaje JavaScript (2 puntos)

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

JavaScript fue creado en sólo 10 días por Brendarn Eich en 1995. Fue creado como un lenguaje de scripting para Netscape Navigator para agregar interactividad básica a los sitios web. Actualmente, JavaScript ha evolucionado como un lenguaje versátil que se usa tanto para el desarrollo frontend como backend.

1.2 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 un lenguaje de programación que se utiliza para hacer que las páginas web sean más interactivas (controlar multimedia, animar imágenes, validar formularios).

Javascript se utiliza del lado del cliente y para eso descarga el código y luego se carga en la memoria del cliente y luego se interpreta el código línea a línea para ejecutar las instrucciones.

- Permite almacenar valores dentro de variables.
- Realizar operaciones sobre fragmentos de textos, por ejemplo colocar todo en mayúsculas, unir dos texto
- Ejecutar código en respuestas para algunos eventos, ejemplo al hacer click en un botón.
- Otra opción son el uso de API, como el DOM, APi de geolocalización, apis canvas y webGL para construcción de gráficos y API de audio y video para agregar multimedia

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

El nacimiento de Javascript está íntimamente ligado al desarrollo de la web: el propósito original de este lenguaje fue enriquecer los documentos HTML con más dinamismo e interactividad.

Con el tiempo tanto el lenguaje como los entornos de ejecución de javascript han mejorado enormemente, relegando a un segundo lugar a otras alternativas.

La situación actual es que existen dos tipos de entornos en los que el código javascript se puede ejecutar, el «clásico» que funciona sobre el navegador web y el «revolucionario» que lo hace directamente sobre el sistema operativo. Lo bueno es que ambos son extremadamente parecidos en su diseño, lo que permite aprender con fundamento a programar en javascript sobre el navegador web, sirve también para programar en javascript sobre el sistema operativo.

JavaScript se puede ejecutar en varios entornos virtuales, como:

- **Navegadores web**: Todos los navegadores web tienen un entorno de ejecución de JavaScript por defecto.
- **Chrome V8**: Este motor de JavaScript puede ejecutar código JavaScript tanto dentro como fuera de un navegador.
- **Node.js**: Se puede utilizar para desarrollar aplicaciones del lado del servidor.
- **Azure Portal**: Se puede utilizar para configurar un entorno de desarrollo de JavaScript.
- **Google App Engine**: Se puede utilizar para ejecutar JavaScript.
- **Nodeenv**: Se puede utilizar para crear entornos node.js aislados.

JavaScript es un lenguaje de programación que se puede insertar en cualquier página web y utilizar con otros lenguajes de desarrollo web. El código JavaScript se limita a lo que se puede hacer dentro de la plataforma sobre la que se está ejecutando el script.

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

A diferencia de otros lenguajes como Java o C ++, JavaScript tiene un tipado dinámico. Por otro lado, JavaScript se ejecuta línea por línea en lugar de ser compilado. También, es multiparadigma pues soporta programación orientada a objetos (Corfo, 2024).

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

Fortalezas:

- **Compatibilidad Multiplataforma**: lo que permite su uso en distintos navegadores y, dispositivos y sistemas operativos
- **Ejecutable en Muchos Navegadores**, Al no tener la compilación previa, es mas rapida su programación, facilitando la creación más accesible

- ❖ **Facilidad de Aprendizaje**, al ser de sintaxis sencilla facilita su aprendizaje y así mismo tiene mucha documentación
- ❖ **Interactividad en Tiempo Real**: sin necesidad de recargar las páginas, se logra interactividad de inmediato por ejemplo al hacer clic, se puede pintar una zona, reproducir una canción, sin necesidad de recargar la página

Una de las debilidades de javascript:

- v**ulnerabilidades de seguridad**, si las variables no son sanitizadas, se puede ingresar código no deseado que puede afectar a las base de datos.
- Otra forma es que los atacantes inyecten código malicioso y para luego ser ejecutado en los usuarios finales
- **Puede existir problemas de compatibilidad** entre navegadores, se puede dar el caso que el mismo código se comporte de una forma y en otro navegador de otra forma, lo que es importante de revisar bien el uso en distintos navegadores
- **Al depender del cliente**, depende del hardware del cliente y también entre mas compleja la aplicación con javascript mas recursos sobre todo en aplicaciones móviles

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

Javascript es un lenguaje síncrono. Todo lo que hagas se ejecutará en el mismo hilo. Javascript tiene un único hilo, por lo tanto, no ofrece la posibilidad de simultaneidad en su ejecución.

En programación, nos referimos a procesos síncronos a aquellos que se realizan de manera secuencial, uno detrás de otro, sin simultaneidad.

Los procesos asíncronos, sin embargo si se pueden realizar en javascript, se realizan en paralelo, todos al mismo tiempo.

Entonces, ¿cómo se realizan los procesos asíncronos en Javascript?

Los patrones asíncronos más comunes en Javascript son:

1. *Callback*. Función que se ejecuta cuando una operación asíncrona termina, como resultado de la misma.
2. Promesa. Representa el resultado de una operación asíncrona. Se configura con dos *callbacks* para resolver la promesa con éxito o con fallo.
3. Async/Await. Azúcar sintáctico para manejar promesas de una forma más simple. *Async* declara una función como asíncrona mientras que *await* gestiona la resolución de una promesa de forma automática. *Await* debe emplearse siempre dentro de declaraciones *async*. Atento a múltiples *await*, piensa bien el comportamiento que necesitas.

Ejemplo de async/await para gestionar una promesa

```
const getPokemon = async () => {
   const result = await  fetch('https://pokeapi.co/api/v2/pokemon/ditto/');
   const res = await result.json();
   console.log(res);
};
getPokemon();
```

2. Evolución del Lenguaje JavaScript y el Estándar ECMAScript (2 puntos)

Incluye en el informe un análisis sobre:

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

Lenguaje Interpretativo	Lenguaje Compilado
Es ejecutado línea por línea y a la vez ejecuta cada comando. Ejemplos PHP, Ruby, Python y JavaScript. son más flexibles	C++ o Java son compilados y se compilan a código máquina antes de ejecutarse. Después de programar pasan por un proceso de transformación del código a un lenguaje de más bajo nivel. Son más rápidos y eficientes al ejecutarse en comparación con los lenguajes interpretados.

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

ECMAScript específicamente es el estándar que a partir del año 2015 a la actualidad se encarga de regir como debe ser interpretado y funcionar el lenguaje JavaScript, siendo este (JS – JavaScript) interpretado y procesado por multitud de plataformas, entre las que se encuentran los navegadores web, NodeJS u otros ambientes como el desarrollo de aplicaciones para los distintos sistemas operativos que actualmente existen en el mercado.

Los responsables de dichos navegadores y JavaScript deben encargarse de interpretar el lenguaje tal como lo fija ECMAScript.
Evolución de ECMAScript

A continuación te daremos un resumen de la evolución que ha tenido ECMAScript:

- ➔ Año 1995: El programador y trabajador Brendan Eich de la empresa Netscape como se mencionó anteriormente creó un lenguaje de programación llamado “Mocha”, posteriormente la empresa Sun Microsystem adquirió Netscape y este pasó a llamarse JavaScript hasta la actualidad.

- ➔ 1997: Nace el estándar Document Object Model (DOM) para evitar las incompatibilidades entre los navegadores. Este mismo año, surge el estándar ECMA-262 y sale al mercado ECMAScript 1.

- ➔ 1998: Sale ECMAScript 2 con la actualización del formato en la especificación para alinearse con los estándares ISO.

- ➔ 1999: Es lanzado ECMAScript 3, cuya novedad era el soporte para expresiones regulares, gestión estructurada de excepciones y otras mejoras puntuales.

- ➔ 2009: Luego de una década nace ECMAScript 5 ya que la versión 4 no llegó a feliz término puesto que le querían agregar una cantidad considerable de cambios y perdería el propósito inicial de este. En ECMAScript 5 se añadió el soporte nativo para JSON o los getters y setters para propiedades, entre otras pequeñas mejoras.

- ➔ 2011: Es puesta en marcha la versión ECMAScript 5.1, está simplemente alineaba el estándar de ECMA con el formato correspondiente de ISO (ISO/IEC 16262:2011).
- ➔ 2015: Sale a producción ECMAScript 6 la cual llegó con una serie de mejoras de las cuales podemos mencionar la 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.

- ➔ 2016: ECMAScript 7 esta versión trajo consigo mejoras básicamente el operador de exponenciación y un método nuevo para las matrices que permite comprobar si existen ciertos elementos dentro de éstas.

- ➔ 2017: La octava edición ECMAScript 8 incluyó constructores async/await , los cuales funcionan usando generadores.

- ➔ 2018: ECMAScript 9 agregó los operadores rest/spread para variables.

- ➔ 2019: Este año se agregaron nuevas características entre las cuales podemos destacar las siguientes la función `Array.flat()` devuelve una nueva matriz con cualquier submatriz. Por su parte, `String.trimStart()` puede utilizarse para recortar el espacio en blanco desde el inicio de una cadena. Por otra parte, para el manejo de errores con try / catch se aplica el “error opcional en catch” ya que permite a los desarrolladores utilizar el `try / catch` sin la obligatoriedad de aplicar el parámetro de error dentro del bloque correspondiente.

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

ECMAScript específicamente es el estándar que desde el año 2015 (versión ES6) se ha encargado de regir la forma cómo debe funcionar y ser interpretado el lenguaje JavaScript; la especificación de ECMAScript definida en ECMA-262, nace para crear un lenguaje de scripting de propósito general.

Así mismo, JavaScript (JS) es un lenguaje scripting de programación interpretado de propósito general, *dialecto del estándar ECMAScript*, JavaScript es procesado por una variedad de plataformas entre las que destacan los navegadores web, es aquí donde interviene ECMAScript puesto que estos (los navegadores) deben interpretar el lenguaje tal como ECMA lo indica.

Como diferencias globales podemos destacar las siguientes:

- ECMAScript es un estándar para lenguajes de scripting.
- JavaScript es la implementación más popular del estándar ECMAScript.

Uno de los principales objetivos del comité ECMA al estandarizar JavaScript es garantizar la interoperabilidad entre diferentes implementaciones.
Al definir un estándar común, el comité de ECMA ayuda a garantizar que el código JavaScript se comporte de manera consistente en estas diferentes plataformas, lo que reduce los problemas de compatibilidad y facilita a los desarrolladores la creación de aplicaciones web que funcionan sin problemas en varios entornos.

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

Type Script es un superconjunto de JavaScript que añade tipado estático (Corfo, 2024). Entre sus características se destaca: es un verificador de tipo estático y detecta errores en el código sin ejecutarlo y lo hace en función de los tipos de valores; ofrecen la opción de autocompletado; introduce interfaces y alias de tipo, que permiten describir estructuras de datos de manera clara y reutilizable e incluye características modernas de JavaScript. Esto hace que el código sea más seguro y predecible. Se puede colocar cualquier código de JavaScript que funcione en TypeScript sin preocuparse por como está escrito.

TypeScript es una alternativa a JavaScript porque amplía el lenguaje base, añadiendo características que mejoran la experiencia de desarrollo, especialmente en proyectos grandes y complejos.

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

Ventajas de typeScript:

Capacidad de añadir tipos estáticos a Javascript: definir y asignar tipos a las variables, parámetro de funciones y valores de retorno, logrando una disminución en el errores durante la ejecución del código
Verificación de errores y autocompletado: Logrando reducir los tiempos de depuración

Desventajas de typeScript:

Al no ser tan masivo: se requiere un esfuerzo adicional en aprender y adaptarse a typescript
Proceso de transpilación: Puede aumentar el tamaño del archivo final en proyectos grandes logrando que el rendimiento sea menor
Al principio puede tener tiempo mayor de desarrollo, por el tipado estático ya que se debe definir y mantener para los tipos de datos.

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

A partir de la investigación, realiza un análisis crítico sobre si es pertinente o no integrar JavaScript avanzado o TypeScript en el desarrollo del sitio web del hospital.

3.1. Ventajas de utilizar JavaScript avanzado o TypeScript en el proyecto.

Ventajas de usar **TypeScript** en el hospital, al permitir el tipado estático lo que permite definir el tipo de datos que se puedan ingresar sobre todo en el formulario

3.2. Desventajas o posibles dificultades que podría traer la implementación de estas tecnologías.

Desventaja: Curva de Aprendizaje: al tener algún conocimiento de javascript, debemos primero aprender bien javascript y luego typescript lo que tomaría más tiempo. Al ser con más código el proyecto se puede volver más pesado en tema de tamaño de archivo lo que puede provocar una lentitud en el cliente final (Sobre todo si existe mala internet)

3.3. Conclusión: ¿Es recomendable incluir JavaScript avanzado o TypeScript en el proyecto? Justifica tu respuesta.

Según la información obtenida en el proyecto pequeño no es recomendable el uso de Typescript, pues se implementa en proyectos grandes y complejos, por lo que no es pertinente.

Referencias:

Corfo Becas Capital Humano . (2024). *Módulo 3 - Programación avanzada en JavaScript.*

Resources for Developers, by Developers. https://developer.mozilla.org/es/

Programamos. Unidad 2 entorno de ejecución.
https://programamos.es/unidad-2-los-entornos-de-ejecucion/

Typescriptlang. TypeScript for the New Programmer
https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html

OpenWebinar. https://openwebinars.net/blog/ecmascript-vs-javascript/

Ética. ¿Qué papel jugó el comité ECMA en la estandarización de JavaScript?.

https://es.eitca.org/web-development/eitc-wd-jsf-javascript-fundamentals/introduction-eitc-wd

¿Qué papel jugó el comité ECMA en la estandarización de JavaScript?

-jsf-javascript-fundamentals/java-vs-javascript/examination-review-java-vs-javascript/what-role-did-the-ecma-committee-play-in-the-standardization-of-javascript/

https://www.dongee.com/

https://openwebinars.net/blog/que-es-ecmascript/

https://lemoncode.net/lemoncode-blog/2018/1/29/javascript-asincrono#:~:text=Javascript%20est%C3%A1%20dise%C3%B1ado%20para%20aplicaciones,bien%20el%20comportamiento%20que%20necesitas.

https://pablomonteserin.com/curso/javascript/como-funcionan-la-asincronas-en-javascript/#:~:text=Asincron%C3%ADa%20vs%20Sincron%C3%ADa,los%20procesos%20as%C3%ADncronos%20en%20Javascript?

https://www.freecodecamp.org/espanol/news/lenguajes-compilados-vs-interpretados/

https://ventajasydesventajastop.com/