Modulo 1. Introducción

Intro.

Javascrip es un lenguaje interpretado. Lo que hace que pueda interpretarse es el motor js integrado en el navegador web, pero NO ES LA UNICA MANERA de ejecutar codigo JS.

NodeJs es un interprete que se instala de forma independiente al navegador. Es decir, se puede instalar en un sistema operativo como MacOS, Windows o linux. El uso de node.js te permite escribir programas en JavaScript que, por ejemplo, convertirán tu computadora en un servidor.

La mayoría de los motores de JavaScript modernos utilizan la técnica *Compilación Just In Time - Justo a Tiempo* (*Compilación JIT*). Esta técnica consiste en compilar fragmentos de código durante la ejecución del programa (más de una sola instrucción) y permite aumentar su rendimiento. Sin embargo, desde el punto de vista del usuario, dicho cambio es prácticamente imperceptible: sigue pareciendo que solo el intérprete está ejecutando el código fuente, instrucción por instrucción.

Antes de empezar a tirar codigo, es importante definir el problema a resolver. Antes de comenzar a explicarle algo a la computadora, en otras palabras, escribir un programa, debes comprender exactamente qué deseas lograr y cómo deseas lograrlo. En segundo lugar, la solución que proponemos y escribimos en forma de programa debe ser 100% precisa: la computadora no puede adivinar nada.

Origen.

Su origen se remonta a los años 90. La empresa NetScape le encomendo la tarea a Brendan Eich de crear un lenguaje que permitiera que las paginas web’s fueran más dinamicas, para su navegador web.

Cliente y Servidor.

El uso de JavaScript en los sitios web, que con el tiempo se ha vuelto cada vez más complejo y, a menudo, contiene una lógica muy sofisticada, se denomina programación del **lado del cliente**.

El código a ejecutar se carga junto con la página en el navegador, del lado del usuario, y el intérprete que forma parte del navegador web permite su ejecución.

Con el tiempo, JavaScript comenzó a aparecer en otras áreas, como en la programación de lado del servidor de aplicaciones web complejas, también llamadas back-end. Estos programas se ejecutan en servidores, procesando datos (por ejemplo, de bases de datos), que después del procesamiento estarán disponibles en el lado del cliente. La flexibilidad de este lenguaje y su relativa sencillez lo han hecho mucho más aplicable, por ejemplo, en aplicaciones móviles, o incluso en la programación de UAVs - Vehículo Volador No Tripulado (algunos drones ejecutan programas escritos en este lenguaje).

Desventajas.

Debido a su naturaleza, no es adecuado para ciertas aplicaciones. Por ejemplo, no tiene sentido usarlo para escribir programas que requieran cálculos matemáticos avanzados o un rendimiento muy alto.

Algunas limitaciones se deben al propio concepto del lenguaje, pero la gran mayoría están relacionadas con la plataforma en la que lo usamos. Esto es especialmente visible cuando se escribe código para ser ejecutado en un navegador. En tal situación, la funcionalidad de JavaScript está limitada por el hecho de que los navegadores, por razones de seguridad, ejecutan código de secuencia de comandos en un entorno *sandbox* (un entorno separado del mundo exterior), que no permite acceso a archivos y recursos locales (es decir, aquellos archivos que están en la computadora donde se inicia el navegador).

Otro inconveniente es que como el código no está compilado, entra en el navegador de la misma forma, o muy similar, a la que lo escribimos nosotros. ¿Por qué es esto una desventaja? Esto se debe a que todos pueden ver nuestra solución en una forma fácil de leer y usarla (ya sea en fragmentos o incluso en su totalidad) sin nuestro permiso.

Ventajas.

Otra gran ventaja es una gran cantidad de frameworks y librerías listas para usarse que brindan la mayoría de las funcionalidades y características comúnmente requeridas. El lenguaje en sí es relativamente fácil de aprender y nos permite centrarnos en el trabajo en lugar de luchar con la sintaxis (es decir, la forma de construir las instrucciones que componen el código de nuestro programa).

Una desventaja/ventaja es que es un lenguaje que no cuenta con un manejo estatico de datos. Es decir, una variable puede contener un numero, un char, etc. sin

necesidad de especificar el tipo.
Al agregar *manejo estático de los tipos de datos*, donde una variable solo puede contener un tipo de dato (por ejemplo, números) durante la ejecución del programa, se creo un nuevo lenguaje llamado **TypeScript**.

Herramientas de desarrollo

Entorno de desarrollo local

HTML; CSS; JS