Javascript

Introducción

JavaScript es un lenguaje interpretado típico. Si ejecutamos un código escrito en JavaScript en un navegador web, **intérprete será el Motor JavaScript integrado en el navegador.** Esta no es la única forma de ejecutar código JavaScript.

Quizás hayas oído hablar de **node.js. También es un intérprete, pero se instala independientemente de los navegadores como entorno en el sistema operativo de la computadora** (puede ser macOS, Windows o Linux). El uso de node.js le permite escribir programas en JavaScript que, por ejemplo, convertirán su computadora en un servidor.

JavaScript es un lenguaje interpretado, de hecho, ejecutar un programa escrito en JavaScript parece como si estuviéramos ejecutando nuestro código fuente (es decir, el código que escribimos) paso a paso. Sin embargo, es posible que encuentres información sobre este idioma, y más específicamente sobre intérpretes concretos, pero que sea un poco diferente.

La mayoría de los motores JavaScript modernos utilizan la técnica de compilación Just In Time (compilación JIT). Esta **técnica consiste en compilar fragmentos de código durante la ejecución del programa (más de una única instrucción) y permite aumentar su rendimiento.** Sin embargo, desde el punto de vista del usuario, tal cambio es prácticamente imperceptible: todavía parece como si sólo el intérprete estuviera ejecutando el código fuente, instrucción por instrucción.

Independientemente del idioma que elija, algunas cosas permanecen iguales al escribir el programa. **En primer lugar, una etapa importante y probablemente la más difícil de este proceso es definir correctamente el problema que queremos resolver.** Sólo entonces intentamos encontrar la solución óptima, que finalmente presentaremos en forma de programa escrito en el lenguaje elegido.

Entonces, antes de comenzar a explicarle algo a la computadora, en otras palabras, **a escribir un programa, debe comprender exactamente qué desea lograr y cómo desea lograrlo.** En segundo

lugar, l**a solución que proponemos y escribimos en forma de programa debe ser 100% inequívoca: el ordenador no puede adivinar nada.**

Un ejemplo sencillo de un campo ligeramente diferente: en algún momento de tu vida, probablemente compraste un mueble que requería montaje. Montarlo es un problema que usted, el comprador, ha soportado. Para que pueda realizar esta tarea, recibirá un conjunto de instrucciones que lo guiarán a lo largo de todo el proceso. En este punto, usted actúa como intérprete y utiliza un programa que le permitirá completar la tarea. El éxito de su misión depende de la calidad de estas instrucciones, de si son precisas e inequívocas y no provienen de un mueble más. Al final, puede resultar que no hayas construido los muebles de tus sueños, sino una construcción surrealista de otra dimensión.

Para que las instrucciones sean buenas, alguien que las desarrolle debe saber exactamente qué deben ilustrar, en qué orden deben realizarse ciertas acciones, en qué etapas es más fácil confundir algo, etc. Y, por supuesto, deben saber qué efecto se quiere conseguir al final.

Es así que JavaScript es un lenguaje de programación de interpretado, es un lenguaje de alto nivel, es decir que es relativamente fácil de entender para la mayoría de las personas.

Nota: A principios de los 90, todas las páginas web eran estáticas. Las cosas cambiaron en 1995 cuando la corporación Netscape contrató a Brendan Eich y le encargó que desarrollara un nuevo lenguaje para su producto, el navegador web Netscape Navigator. El nuevo lenguaje se llamó LiveScript, pero poco después se cambió su nombre a JavaScript. **Su tarea principal era agregar dinámica a los sitios web, lo que permitiría, por ejemplo, una interacción más compleja con el usuario.** Y así comenzó la carrera de JavaScript.

Programación del lado del cliente vs Programación del lado del servidor

El uso de JavaScript en sitios web, 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.

Hoy en día, JavaScript es el único lenguaje admitido por los principales navegadores web y alrededor del 95% de las páginas web de todo el mundo incorporan código JavaScript. Desde el principio, las páginas web utilizaron JavaScript en el lado del cliente para agregar interactividad y cambiar dinámicamente el contenido.

Ahora es mucho más que eso, ya que **JavaScript ofrece muchos marcos excelentes sobre los cuales construir aplicaciones web y redes sociales enormes y complejas (probablemente hayas escuchado los nombres de marcos como React o Angular).**

Todo esto puede funcionar en una variedad de equipos, desde estaciones de trabajo de alto rendimiento hasta simples teléfonos inteligentes. Gracias al poder de JavaScript, podemos pedir comida, jugar juegos basados en navegador, ver películas en plataformas de streaming y estar en contacto constante con las personas importantes para nosotros. **JavaScript es tan popular que cada vez se dedica más esfuerzo a utilizarlo, no sólo como una solución del lado del cliente.**

Con el tiempo, **JavaScript** empezó a aparecer en otras áreas, como en la **programación de las partes del 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.

¿Es este el lenguaje de programación perfecto? – desventajas

A pesar de su popularidad y éxito, JavaScript no es un lenguaje de programación perfecto. Por su naturaleza no es adecuado para determinadas aplicaciones. Por ejemplo, no tiene sentido utilizarlo 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 utilizamos. Esto es especialmente visible cuando se escribe código para ejecutar en un navegador, lo que como dijimos anteriormente se llama del lado del cliente. En tal situación, **JavaScript tiene una funcionalidad limitada por el hecho de que los navegadores, por razones de seguridad, ejecutan código de script en un entorno sandbox (un entorno separado del mundo exterior), que no permite el 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, ingresa al navegador en la misma forma, o muy similar, a la que escribimos nosotros mismos. **¿Por qué es esto una desventaja? Esto se debe a que todos pueden ver nuestra solución en un formato fácil de leer y usarla** (ya sea en fragmentos o incluso en su totalidad) **sin nuestro permiso para escribir su propio programa.**

Alguna ayuda aquí puede ser la ofuscación de código, que consiste en transformar nuestro script listo en una forma un poco menos legible (por ejemplo, generando nombres cortos aleatorios de variables y funciones, eliminando signos de fin de línea, etc.), pero el simple hecho es que si alguien quiere robar nuestro código JavaScript, hay muy poco que podamos hacer para detenerlo.

Es este el lenguaje de programación perfecto? – ventajas

Por otro lado, JavaScript tiene muchas ventajas sobre otros lenguajes de programación, y una de las más importantes es una comunidad muy activa y solidaria. Es fácil encontrar soluciones a problemas comunes y encontrar ayuda en general. Esto también significa que se desarrollan activamente herramientas que funcionan con JavaScript.

Otra gran ventaja es una gran cantidad de marcos y bibliotecas listos para usar 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 pelearnos con la sintaxis (es decir, la forma de construir las instrucciones que componen el código de nuestro programa).

Entre sus ventajas podemos destacar en que podemos almacenar datos de cualquier tipo en una variable (una variable es un contenedor en el que almacenamos los datos que usaremos).

Por ejemplo, durante la ejecución del programa, podemos almacenar el número 10 en una variable, y en el siguiente paso usar la misma variable para almacenar la cadena "abc" (eliminando el valor anterior automáticamente, por supuesto; no te preocupes si No lo entiendo ahora porque cubriremos todos estos términos más adelante).

Por lo general, esto es muy conveniente, pero varias personas han encontrado que esta característica del idioma es una desventaja. En su opinión, facilita que un programador cometa errores en determinadas situaciones. Al agregar escritura estática , donde una variable solo puede contener un tipo de variable (por ejemplo, números) durante la ejecución del programa, se introdujo un nuevo lenguaje llamado TypeScript .
Comencemos ahora con JavaScript, que por su sintaxis flexible y sencilla es perfecto para aprender como primer idioma.

Materia: Programación III
Docente: Verónica Godoy
Curso: 6to