¿Qué es exactamente Node.js?

Node.js es un entorno de ejecución de JavaScript. Suena genial, pero ¿qué significa eso? ¿Cómo funciona?

El entorno de tiempo de ejecución de Node incluye todo lo que necesita para ejecutar un programa escrito en JavaScript.

Node.js nació cuando los desarrolladores originales de JavaScript lo ampliaron de algo que solo podía ejecutar en el navegador a algo que podía ejecutar en su máquina como una aplicación independiente.

Ahora puede hacer mucho más con JavaScript que simplemente hacer que los sitios web sean interactivos.

JavaScript ahora tiene la capacidad de hacer cosas que otros lenguajes de programación como Python pueden hacer.

Tanto el JavaScript de su navegador como Node.js se ejecutan en el motor de tiempo de ejecución de JavaScript V8. Este motor toma su código JavaScript y lo convierte en un código de máquina más rápido. El código de máquina es un código de bajo nivel que la computadora puede ejecutar sin necesidad de interpretarlo primero.

¿Por qué Node.js?

Aquí hay una definición formal como se da en el sitio web oficial de Node.js :

> *Node.js® es un tiempo de ejecución de JavaScript construido en el motor de JavaScript V8 de Chrome .*
>
> *Node.js usa un modelo de E / S sin bloqueo controlado por eventos que lo hace liviano y eficiente.*
>
> *El ecosistema de paquetes de Node.js, npm , es el ecosistema de bibliotecas de código abierto más grande del mundo.*

Ya analizamos la primera línea de esta definición: “Node.js® es un tiempo de ejecución de JavaScript construido en el motor de JavaScript V8 de Chrome”. Ahora entendamos las otras dos líneas para que podamos descubrir por qué Node.js es tan popular.

E / S se refiere a entrada / salida. Puede ser cualquier cosa, desde leer / escribir archivos locales hasta realizar una solicitud HTTP a una API.

La E / S lleva tiempo y, por lo tanto, bloquea otras

funciones.

Considere un escenario en el que solicitamos una base de datos de back-end para los detalles de user1 y user2 y luego los imprimimos en la pantalla / consola. La respuesta a esta solicitud lleva tiempo, pero ambas solicitudes de datos del usuario se pueden realizar de forma independiente y al mismo tiempo.

Bloqueo de E / S

En el método de bloqueo, la solicitud de datos del usuario 2 no se inicia hasta que los datos del usuario 1 se imprimen en la pantalla.

Si se tratara de un servidor web, tendríamos que iniciar un nuevo hilo para cada nuevo usuario. Pero JavaScript es de un solo subproceso (en realidad no, pero tiene un ciclo de eventos de un solo subproceso, que discutiremos un poco más adelante). Por lo tanto, esto haría que JavaScript no sea muy adecuado para tareas de subprocesos múltiples.

Ahí es donde entra la parte de no bloqueo.

E / S sin bloqueo

Por otro lado, usando una solicitud sin bloqueo, puede iniciar una solicitud de datos para el usuario2 sin esperar la respuesta a la solicitud para el usuario1. Puede iniciar ambas solicitudes en paralelo.

Esta E / S sin bloqueo elimina la necesidad de subprocesos múltiples, ya que el servidor puede manejar múltiples solicitudes al mismo tiempo.