What is the Node.js Event Loop

الـ Event Loop في Node.js عشان ينظم ترتيب تنفيذ الكود، زي الـ async. يعني هو بيقرر إمتى كل مهمة تتنفذ ويمشي على الـ call stack والـ event queue عشان يضمن إن الكود يشتغل من غير ما أي حاجة توقف التانية.

What is Libuv and What Role Does It Play in Node.js

الـ Libuv ده مكتبة مكتوبة بلغة C++، وده اللي بيخلي Node.js يقدر يتعامل مع الـ async operations ويشتغل على أكتر من حاجة في نفس الوقت. Node.js نفسه نصه C++ ونصه JS، فالنص الـ C++ بيشتغل مع الـ Libuv عشان ينفذ العمليات، سواء كانت متعلقة بالـ file system أو الشبكة أو أي حاجه تانية، وده بيخلي Node.js سريع جدًا في التعامل مع المهام المتعددة.

How Does Node.js Handle Asynchronous Operations Under the Hood

لما Node.js يلاقي عملية async زي قراءة ملف أو طلب API، بيبعتها للـ Libuv. الـ Libuv بتدي العملية دي thread . بعد ما العملية تخلص، النتيجة بتتسلم للـ Event Loop، اللي بدوره بيحط الـ callback على الـ call stack عشان ينفذ. كده Node.js يقدر يشغل حاجات كتير من غير ما يقف على أي عملية طويلة.

What is the Difference Between the Call Stack, Event Queue, and Event Loop in Node.js

- **Call Stack**: ده المكان اللي الكود الـ synchronous بيتنفذ فيه، أي task هنا تتخلص تتشال.
- **Event Queue**: ده الطابور اللي بيتجمع فيه الـ callbacks بتاعت الـ async tasks لحد ما يبقى وقت تنفيذهم.
- **Event Loop**: ده الحلقة اللي بتلف على الـ stack كل شوية، لو فاضي بيجيب من الـ event queue callbacks وينفذهم، وده اللي بيخلي Node.js يشتغل بطريقة non-blocking.

What is the Node.js Thread Pool and How to Set the Thread Pool Size

الـ Thread Pool ده مجموعة من الـ threads اللي الـ Libuv بتستخدمها لتنفيذ العمليات التقيله من غير ما توقف الـ main thread. عددهم افتراضيًا 4، لكن ممكن تزوده عن طريق size=6 لو عايز مثلاً 6 threads. العدد الأمثل بيعتمد على عدد الـ logical processors في جهازك، وده بتقدر تشوفه من Task Manager.

How Does Node.js Handle Blocking and Non-Blocking Code Execution

JavaScript في الأساس synchronous وsingle-threaded، يعني أي عملية تقيلة ممكن توقف كل الكود اللي بعدها. Node.js حل المشكلة دي عن طريق الـ Libuv والـ Event Loop والـ thread pool، اللي بيخلوا العمليات الثقيلة تشتغل بره الـ main thread، والنتيجة ترجع لما تخلص. كده الكود البسيط والـ async tasks التانية تقدر تشتغل من غير أي توقف، وده اللي بنسميه non-blocking execution.