What is Node.js?

Node.js is an open-source platform built on Chrome's JavaScript runtime used to develop fast and scalable applications quickly. It uses an event-driven, non-blocking input/output architecture, making it lightweight, efficient, and ideal for data-driven, real-time distributed applications.

While Node.js can perform synchronously, it usually operates asynchronously, meaning that an application can call an event E_1 with a callback registered to handle the return. While awaiting the return for E_1, another event E_2 can join the execution queue. Once E_1 completes, its callback event is executed and handled by the function call that invoked the callback.

Why monitor Node.js?

Node.js is famous for its asynchronous, event-driven, non-blocking I/O model. However, this attribute causes an inconvenience because verifying the correctness of an application with asynchronous nested callbacks is complex, thus mandating a Node.js monitor.

Developers must monitor their applications in real-time using a Node.js monitoring dashboard to ensure optimal performance.

What should you look for in a Node.js dashboard?

Any reliable Node.js network monitor must provide enough information to help efficiently identify the problem sources. Some crucial information includes process ID, log management, request rate, application availability, resource usage, uptime, downtime, system health, error rates and handling, number of connections, load average, and latency.

What unique value does [Company Name] offer?

[Company Name]'s Node.js monitoring tool provides essential productivity and efficiency-enhancing innovations.

Our Service Maps acquaints developers with their system's architecture providing vital insights to identify issues quickly.

Our Error Analytics granularly pinpoints offending lines of code, relieving developers of a potentially arduous task so that they can concentrate on resolving issues.

Lastly, our solution offers historical data that could prove essential in process improvement in addition to real-time information. For example, developers can feed these data into an anomaly detection system for failure prediction, thus enhancing application support.

So if your organization asks, 'How do I monitor a Node.js application?'. [Company Name] is the answer, and you can contact us here today.

References

Ancona, D. *et al.* (2018) 'Towards runtime monitoring of Node. js and its application to the Internet of Things', *arXiv preprint arXiv:1802.01790*.

Gackenheimer, C. (2013) ‘Understanding Node. js’, in *Node. js Recipes*. Springer, pp. 1–26.

Kahuha, E. (2021) *Top Node.js Monitoring Tools | Engineering Education (EngEd) Program | Section*. Available at: https://www.section.io/engineering-education/top-node.js-monitoring-tools/ (Accessed: 1 September 2021).

Lei, K., Ma, Y. and Tan, Z. (2014) 'Performance comparison and evaluation of web development technologies in php, python, and node. js', in *2014 IEEE 17th international conference on computational science and engineering*, pp. 661–668.

Mardan, A. (2018) 'Getting Node. js Apps Production Ready', in *Practical Node. js*. Springer, pp. 331–364.

New Relic Inc. (2021a) *Errors page: Find, fix, and verify problems | New Relic Documentation*. Available at: https://docs.newrelic.com/docs/apm/apm-ui-pages/error-analytics/errors-page-find-fix-verify-problems/ (Accessed: 1 September 2021).

New Relic Inc. (2021b) *Introduction to service maps | New Relic Documentation*. Available at: https://docs.newrelic.com/docs/understand-dependencies/understand-system-dependencies/service-maps/introduction-service-maps/ (Accessed: 1 September 2021).