

Change page

About Node.js®



About Node.js®

As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications. In the following "hello world" example, many connections can be handled concurrently. Upon each connection, the callback is fired, but if there is no work to be done, Node.js will sleep.

CJS ESM

```
1 const { createServer } = require('node:http');
2
3 const hostname = '127.0.0.1';
4 const port = 3000;
5
6 const server = createServer((req, res) => {
7   res.statusCode = 200;
8   res.setHeader('Content-Type', 'text/plain');
9   res.end('Hello World');
10 });
11
12 server.listen(port, hostname, () => {
13   console.log(`Server running at http://${hostname}:${port}/`);
14 });
```

JavaScript

</> Copy to clipboard

This is in contrast to today's more common concurrency model, in which OS threads are employed. Thread-based networking is relatively inefficient and very difficult to use. Furthermore, users of Node.js are free from worries of dead-locking the process, since there are no locks. Almost no function in Node.js directly performs I/O, so the process never blocks except when the I/O is performed using synchronous methods of Node.js standard library. Because nothing blocks, scalable systems are very reasonable to develop in Node.js.

If some of this language is unfamiliar, there is a full article on [Blocking vs. Non-Blocking](#).



About Node.js

