Über Node.js®

Als asynchrone, Event-basierte Laufzeitumgebung wurde Node speziell für die Entwicklung von skalierbaren Netzwerkanwendungen entworfen. Im nachfolgenden "Hallo Welt"-Beispiel können viele Verbindungen gleichzeitig bearbeitet werden. Bei jeder neuen Anfrage wird die Callback-Funktion ausgeführt. Gibt es jedoch nichts zu tun, befindet sich Node im Ruhezustand.

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hallo Welt\n');
});

server.listen(port, hostname, () => {
  console.log(`Server läuft unter http://${hostname}:${port
});
```

Dies steht im Gegensatz zu den heutzutage üblichen Modellen für Nebenläufigkeit, bei denen Threads des Betriebssystems genutzt werden. Thread-basiertes Networking is vergleichsweise ineffizent and sehr

schwer umzusetzen. Zudem müssen sich Node-Nutzer nicht um Deadlocks im Prozess sorgen, da es keine Blockierung gibt. Fast keine Funktion in Node.js führt direkt I/O-Operationen aus, daher wird der Prozess nie blockiert. Da nichts blockiert, können mit Node sinnvoll skalierbare Systeme entwickelt werden.

Wenn einige dieser Konzepte unbekannt sind, gibt es hier einen Artikel zum Thema blockierend vs. blockierungsfrei (auf Englisch).

Node ähnelt im Design und ist beeinflusst von Systemen wie Rubys "Event Machine" oder Pythons "Twisted". Node führt das Event-Modell noch etwas weiter. Die Ereignisschleife ist ein Konstrukt direkt in der Laufzeitumgebung und wird nicht über eine Bibliothek eingebunden. In anderen Systemen ist immer ein blockierender Aufruf notwendig, um die Ereignisschleife zu starten. Üblicherweise wird das Verhalten in Callback-Funktionen am Anfang des Skripts definiert und am Ende wird mit einem blockierenden Aufruf wie `EventMachine::run()` ein Server gestartet. In Node gibt es keinen solchen Aufruf, um die Ereignisschleife zu starten. Node beginnt einfach mit der Ereignisschleife, nachdem das Eingabe-Skript ausgeführt wurde. Node verlässt die Ereignisschleife, wenn keine Callback-Funktionen mehr auszuführen sind. Dieses Verhalten ist wie bei Browser-JavaScript - die Ereignisschleife ist vor dem Nutzer versteckt.

HTTP ist ein Basiselement in Node, entworfen mit Fokus auf Streaming und geringe Latenz. Dadurch ist Node sehr gut als Grundlage für Web-Bibliotheken oder Frameworks geeignet.

Dass Node ohne Threads entworfen ist, bedeutet nicht, dass man die Vorteile von mehreren Kernen auf einer Maschine nicht ausnutzen kann. Untergeordnete Prozesse können mit der `child_process.fork()` API gestartet werden und sie wurden so entworfen, dass man leicht mit ihnen kommunizieren kann. Auf der gleichen Schnittstelle setzt das `Cluster` Modul auf, dass es Prozessen erlaubt Sockets gemeinsam zu nutzen, um Lastverteilung über Kerne hinweg zu ermöglichen.