# Node.js #01 - Was ist Node.js?

## Inhaltsverzeichnis



# 1 Geschichte

2009 wurde Node.js von Ryan Dahl aufgrund einer vergleichsweise trivialen Aufgabe erfunden: Dahl hatte es sich zur Aufgabe gemacht einen Fortschrittsbalken für einen Dateiupload umzusetzen. Dabei versuchte er es mit den verschiedensten Skript- und Programmiersprachen, wobei er bei allen getesteten Sprachen die Grenzen dieser erreichte und die Lösung somit nur spärlich erreicht werden konnte.

Somit fand Dahl die Lösung dann in einer Umsetzung in JavaScript, da dieses seinen gesamten Anforderungen gerecht werden konnte. Im Laufe der Entwicklung von Node.js entstanden unter Anderem der Node Package Manager und die Windows-Unterstützung.

Als nach etwa 5 Jahren Entwicklung noch immer keine stabile Lösung von Node.js veröffentlicht werden konnte, entstand im Dezember 2014 io.js als Fork von Node.js, da viele Unternehmen Node.js für wichtige Systeme einsetzen wollten, diese aber aufgrund der Instabilität von Node.js davon abgehalten wurden. Dies sollte sich dann mit io.js ändern.

Im Juni 2015 wurden dann die beiden Einzelprojekte Node.js und io.js wieder vereint. Fortan wurde die Entwicklung von der Node.js Foundation koordiniert und weitergeführt. Seither ist mit häufigen Releases und stabilen mit Langzeit-Support zu rechnen.

# 2 Funktionsweise

## 2.1 Allgemeines

Node.js nutzt ein sogenanntes nicht-blockierendes Input-/Output-System (Nonblocking IO) für sämtliche Operationen und ist dabei dennoch Single-Thread-basiert.

Das Prinzip dahinter ist ein eventgetriebenes System. So wird bei Interaktion eines Nutzers ein Event ausgelöst auf welches das System mit Funktionsaufrufen reagieren kann. So werden beispielsweise (zeitintensive) Leseoperationen von Dateien durch eine Ereignisschleife auf das Betriebssystem ausgelagert. Mit Hilfe von Callbacks kann dann das Ergebnis der Leseoepration dann vom Betriebssystem empfangen und verarbeitet werden.

## 2.2 V8-Engine

Neben dem eventgetriebenen System spielt aber auch Google's V8-Engine, die in Google Chrome genutzt wird, eine wichtige Rolle bei der Funktionsweise von Node.js. Mit Google's V8-Engine hat Node.js eine aktuelle und schnelle JavaScript-Engine. Durch die stetige Weiterentwicklung dieser Engine durch Google für Chrome ist auch eine stets aktuelle Basis geschaffen.

Da eine genaue Beschreibung der V8 den Rahmen dieses Eintrags sprengen würde nur kurz die wichtigsten Merkmale:

Die V8-Engine kompiliert beim Ausführen der Anwendung den Quellcode mittels einer Just-In-Time-Kompilierung in Maschinencode, wodurch sämtliche Operationen schneller und effizienter ausgeführt werden können. Neben der Just-In-Time-Kompilierung beinhaltet die V8 eine weitere, zweite Kompilierung: Die Ahead-Of-Time-Kompilierung. Diese verbessert während der Laufzeitausführung den Code und macht diesen somit erneut performanter. Dazu führt die V8-Engine interne Statistiken über die durchschnittliche Ausführungszeit und Anzahl an Funktionsaufrufen und verbessert dann je nachdem den Maschinencode.

Neben diesen Optimierung sind weitere Optimierungen beispielsweise ein internes Caching von Daten oder Hidden Classes, auf welche ich aufgrund ihrer Komplexität hier nicht eingehen kann. Desweiteren kommt ein Garbage-Collector zum Einsatz, da sonst Anwendungen, die mehrere Wochen oder Monate am Stück laufen, den RAM-Verbrauch in kürzester Zeit in die Höhe treiben würden.

Ein solcher Garbage-Collector kommt bei Clientseitigen Anwendungen außerdem nicht zum Einsatz, da durch das Neuladen einer Seite der Arbeitsspeicher geleert wird und daher ein Garbage-Collector nicht von Nöten ist.

### 2.3 Node Package Manager

Der Node Package Manager ist bereits seit November 2011 in Node.js integriert und bietet somit einen Manager für Erweiterungen. Diese Erweiterungen werden von jeglichen Nutzern aus der Node.js-Community entwickelt, veröffentlicht und gepflegt. Dank dieser Erweiterungen/Pakete stellt sich die Entwicklung vieler Anwendungen als eine Leichtigkeit heraus, da beispielsweise Datenbankanbindungen oder Mailclients nicht selber implementiert werden müssen und somit wird auch Node.js selber nicht mit ggf. unnötigem Müll zugepackt. Ich werde im Verlauf dieser Reihe noch weiter auf den Package Manager eingehen.

## 3 Anwendungsbereiche

Node.js kann weitreichende Anwendungsbereiche abdecken. So können mittels Node.js große, als auch dynamische Webseiten umgesetzt werden, als auch kleine Webseiten, einfache REST-Dienste oder Socket-Server. Aufgrund der Funktionsweise von Node.js sind einem bei den Anwendungsbereichen keine Grenzen gesetzt.

## 4 Vorteile und Nachteile

Wie so ziemlich jede Programmier- oder Skriptsprache hat auch Node.js seine Vor- und Nachteile, auf welche ich nun genauer eingehen werde.

### 4.1 Vorteile

Node.js basiert auf JavaScript, was den Vorteil hat, dass man bereits wichtige Standards wie z.B. ECMA-Script geliefert bekommt und man somit auch keine neue Programmier- bzw. Skriptsprache erlernen muss. Diese Standards sind außerdem gut und umfangreich dokumentiert und im Internet als auch in Büchern nachzulesen. Somit bietet JavaScript für Node.js bereits eine weitreichende Basis an Funktionen.
Außerdem hat JavaScript bereits eine große Community (lt. GitHub-Trends sind die meisten Projekte in

JavaScript).

In der Funktionsweise lässt sich ein weiterer Vorteil feststellen: Das non-blocking IO. So wird die gesamte Ausführung der Sofwtare auf verschiedene Komponenten ausgelagert und das Hauptprogramm wird nicht blockiert, bis eine Operation fertig ausgeführt wurde.

Der letzte entscheidende Vorteil ist die native Unterstützung von JSON. So können beispielsweise REST-Dienste einfach umgesetzt werden. Auch in Zusammenhang mit Websockets ist die native Unterstützung von JSON von Vorteil.

### 4.2 Nachteile

Es gibt in meinen Augen nur einen entscheidenden Nachteil von Node.js: Das Stoppen der gesamten Laufzeit bei Auftritt eines Fehlers.

Sobald man einen Fehler erhält, stoppt die Laufzeit ihre Ausführung. Dies kann allerdings mit einigen Möglichkeiten verhindert werden, welche ich später in dieser Reihe genauer erläutern werde.

## 5 Abschluss

Ich hoffe ich konnte eine erste Einführung in Node.js geben.

Im nächsten Eintrag werden wir Node.js installieren, den Node Package Manager genauer betrachten und unser erstes Projekt beginnen.

*Dir hat mein Eintrag gefallen? Dann würde ich mich über einen Like freuen.*

*Dir fehlt etwas? Dann lass es mich in den Kommentaren wissen.*

*Du möchtest, dass ich in Zukunft auf etwas bestimmtes eingehe? Dann schreib auch das mir.*