Titel: Learning Node.
Autor: Shelley Powers
Jahr: First Edition September 2012
Verlag: O'Reilly
Seitenzahl: 374
Preis: 29,00 EUR
ISBN: 978-1-449-32307-3

1 Allgemeine Bemerkungen

Das Buch *Learning Node* richtet sich an Leser mit JavaScript-Kenntnissen, die mit Node serverseitiges bzw. weborientiertes Programmieren erlernen möchten.

2 Gliederung des Buches

Das Buch ist so strukturiert, dass es mit der Installation von *Node* beginnt und im Weiteren die Struktur von *Node* beschreibt. Dazu gehören die Kern-Module von Node, die seine Hauptfunktionalität ausmachen (wie z.B. *process*, *require* u.a.). Im weiteren Verlauf des Buches werden verschiedene Frameworks vorgestellt, die Node bei der Web-orientierten Programmierung einsetzt. Dabei werden auch unterschiedliche Datenbanken vorgestellt, die eine wichtige Rolle spielen. Schließlich wird die Rolle der Sicherheit bei *Node*-Anwendungen beschrieben.

Kapitel 1: Node.js: Up and Running

Im ersten Kapitel wird demonstriert, wie man eine *Node*-Entwicklungsumgebung einrichtet. Dabei wird erläutert, wie man *Node* in Ubuntu und unter Windows7 installiert und anschließend einrichtet.

Schließlich schreiben wir zur Einführung ein kleines Programm, welches „Hello World!“ ausgeben soll, und die Autorin führt die ersten wichtigsten Begriffe ein, die für *Node* relevant sind: „Asynchrone Funktionen“ und „Node Event Loop“.

Kapitel 2: Interactive Node with REPL

Im zweiten Kapitel wird erläutert, wie kleine *Node*-Anwendungen auf der Konsole geschrieben werden können und welche Hilfsprogramme dabei nützlich sind (wie z.B. *rlwrap*).

Kapitel 3: The Node Core

Im dritten Kapitel gewährt uns die Autorin erstmals einen genaueren Blick in die Kern-Funktionen von *Node*:

- **Globale *Node*-Objekte:**
  - *global* ist das Objekt des globales Namensraums. Es entspricht in gewisser Weise dem globalen Objekt *window* eines Browsers und ermöglicht den Zugang zu globalen Eigenschaften und Methoden. Der wesentliche Unterschied zu *window* besteht darin, dass *global* für dasjenige Modul global ist, in welchem sich die Funktion befindet, in dem die globale Variable definiert wurde.
  - *process* ist ebenso ein globales Objekt. Jede *Node*-Anwendung bildet dessen Instanz.
  - Schließlich ist *Buffer* ein globales Objekt für die Behandlung von binären Daten.
- **Timer-Funktionen** wurden in die *Node*-Kern-API integriert, da sie auch in Browser-Anwendungen nicht wegzudenken sind.
- **Server bzw. Sockets**:
  - TCP Sockets und Server
  - HTTP-Webserver
  - UDP/Datagram Socket
- **Streams, Pipes and Readline ('read-by-line')** stellen Daten dar, die zwischen Sockets transportiert werden und können dabei jeglicher Art Modifikation bzw. Manipulation unterzogen werden.
- sog. **Kindprozesse**, die in einer *Node*-Anwendung mittels der Methode **child_process.spawn** aufgerufen werden können, um Befehle auf Systemebene durchzuführen, wie. z.B. ein **pwd**-Befehl (print-work-direktory).
- **DNS-Module**, die für die Auflösung der DNS sorgen.
- **Utilities-Module**, mit deren Hilfe man z.B. testen kann, ob ein Objekt ein Array ist oder nicht.
- **EventEmitter-Objekte** sind eine der wichtigsten Komponenten in *Node*-Anwendungen, denn ohne sie wäre die asynchrones Abarbeiten der Handlungen bei den Objekten nicht möglich.

Kapitel 4: The Node Module System

Im vierten Kapitel stellt die Autorin kurz das *Node*-Modulsystem vor, und wie die einzelnen Module in eine Anwendung eingebunden werden können. Außerdem erläutert sie, wie man die Module von Drittanbietern installieren kann. Schließlich lernen wir, wie man eigene *Node*-Module erstellen und veröffentlichen kann.

Kapitel 5: Control Flow, Asynchronous Patterns, and Exception Handling

Im fünften Kapitel geht es in erster Linie um asynchrone Strukturen bei der Programmierung mit *Node.js*. Dabei werden sog. *nested callbacks* und *exceptions* ausführlich behandelt.

Schließlich werden folgende Module gesondert vorgestellt:

Step ermöglicht eine einfache Ablaufsteuerung für serielle und parallele Ausführung der Funktionen

Async hält solche Funktionalitäten bereit, die sog. Collections bearbeiten können (z.B. mit solchen Methoden wie **forEach**, **map**, **filter** u.a.)

Kapitel 6: Routing Traffic, Serving Files, and Middleware

Im sechsten Kapitel bauen wir zunächst einen einfachen statischen HTTP-Server.

Anschließend wird eine der wichtigen Zwischenanwendungen (Middleware) von Node vorgestellt: *Connect*, welches als Vermittler zwischen der Anwendung und dem drunter liegenden System dient.

Im weiteren Verlauf des Kapitels lernen wir, wie man Router und Proxy in einer *Node*-Anwendung aufsetzen kann.

Kapitel 7: The Express Framework

Im siebten Kapitel wird anhand eines praktischen Beispiels das *Express*-Framework vorgestellt, mit Hilfe dessen man Websites erstellen kann. Dabei wird ein gesonderter Augenmerk auf Fehlerbehandlung und Routing gelegt.

Schließlich wird der Befehl *curl* vorgestellt, der auf Konsole-Ebene ausgeführt werden kann, um eine Webanwendung zu testen.

Kapitel 8: Express, Template Systems, and CSS

Im achten Kapitel werden drei Arten der Darstellung von Websites vorgestellt, die in *Node* eingesetzt werden können:

Embeded JavaScript (EJS) wird direkt in ein HTML eingebunden

Jade Tepmlate System ist standardmäßig mit Express-Framework mit installiert. Es erzeugt dynamisch CSS-Layouts. Die Vereinfachung gegenüber EJS besteht darin, dass man die eckigen Klammern der HTML-Tags nicht mehr schreiben muss.

Stylus für CSS mit Hilfe von *Stylus* wird immer wieder ein statisches CSS-Layout erzeugt, auch bei jeder Modifikation.

Kapitel 9: Structured Data with Node and Redis

Im neunten Kapitel wird die von *Node* unterstütze Datenbank *Redis* anhand einer Spiele-Anwendung vorgestellt.

Kapitel 10: Node and MongoDB: Document-Centric Data

Im zehnten Kapitel wird eine weitere Datenbank vorgestellt (*MongoDB*), die ebenfalls von *Node* unterstützt wird. Es wird erläutert, wie man MongoDB installiert und Daten anlegen, abfragen, updaten, finden und löschen kann. Schließlich wird ein letztes Konzept von Datenbankzugriff vorgestellt: *Mongoose*, mit dessen Hilfe man Such-Objekte definieren kann und die Suchbefehle nicht mehr direkt in *MongoDB* schreiben muss.

Kapitel 11: The Node Relational Database Bindings

Im elften Kapitel werden die relationalen Datenbanken vorgestellt, die *Node* unterstützt:

- db-MySQL
- MySQL
- Sequelize

Kapitel 12: Graphics and HTML5 Video

Im zwölften Kapitel werden zunächst Techniken vorgestellt, mit deren Hilfe man PDF-Tools durch Kindprozesse einbinden kann. So kann man das *pdftk*-Programm in eine *Node*-Anwendung integrieren, um PDF-Dateien auf unterschiedlicher Art zu manipulieren, sei es zu splitten oder zusammenzufügen.

Anschließend wird erläutert, wie ImageMagick durch einen Kindprozess erreicht werden kann und wie dadurch Bilder manipuliert werden können.

Schließlich wird erläutert, wie eine HTML5 Video Anwendung mit HTTP zum Funktionieren gebracht werden kann.

Kapitel 13: WebSockets and Socket.IO

Im dreizehnten Kapitel erfahren wir alles Wissenswerte über WebSockets, wie man Socket.IO konfiguriert und mit dem Framework *Express* verwenden kann.

Kapitel 14: Testing and Debugging Node Applications

Im vierzehnten Kapitel werden Techniken vorgestellt, die für das Debugging von *Node*-Anwendungen von großer Bedeutung sind. Dazu gehören solche Module wie *inspector*.

Im weiteren Verlauf des Kapitels werden Module vorgestellt, mit deren Hilfe man Tests durchführen kann:

- **Unit Tests**
  - *assert*
  - *Nodeunit*
  - *Mocha*
  - *Jasmine*
  - *Vows*
- **Akzeptanztests**
  - *Soda*
- **Benchmark Tests**
  - *ApacheBench*
- **Laod Tests**
  - *Nodeload*

Kapitel 15: Guards at the Gate

Im fünfzehnten Kapitel wird erläutert, wie man die Node-Anwendungen sicherer macht und zwar mittels Verschlüsselungen: TLS/SSL oder *passport*-Module.
Schließlich stellt die Autorin eine weitere Sicherungsstrategie vor: die Twitter Passport Strategy *OAuth*. Ausserdem wird das Modul *node-validator* vorgestellt, welches prüfen soll, ob Daten, die hereinkommen, nicht nur sicher sind sondern auch konsistent sind.

Kapitel 16: Scaling and Deploying Node Applications

Im sechzehnten Kapitel wird sehr detailliert die Struktur der Datei **package.json** erläutert, die jedes Modul der Node-Anwendungen haben muss und was die Entwickler bei den Angaben beachten müssen, wenn sie ihre Module veröffentlichen möchten.

Schließlich wird kurz auf die Diskussion über die *Node*-Anwendungen im Cloud-Computing eingegangen und Für- und Kontra diskutiert.

2.1 Tippfehler

Seite 22 Beispiel mit der Inkrementierung *_++;*
muss geändert werden in:
++_;

Seite 36 global *windows*
muss zweimal geändert werden in:
window

Seite 41 Listing, die letzte schließende Klammer *}*
muss geändert werden in:
});
}

Seite 92 Step, erster Satz *... enables simplified control flow for serial and parallel execution.*
muss geändert werden in:
flow control

Seite 139 Example 7-3 Widget Description: * ... *
muss geändert werden in:
* ... *

Seite 156 Example 8-1 // render or error *...{title}: 'testing',...*
muss geändert werden in:
'title'

Seite 316 Setting Up TSL/SSL *TSL*
muss geändert werden in:
TLS

3 Fazit:

Das Buch *Learning Node* von Shelley Powers geht von der einseitigen Anwendung von *Node* bei der Programmierung im Server-Bereich aus und versucht schwerpunktmäßig diejenigen Leser zu gewinnen, die mit *Node* weborientierte Anwendungen realisieren möchten, die genauso sowohl robust als auch skalierbar sein sollen, wie die serverseitige Anwendungen. Dabei werden wichtige Frameworks wie *Express* vorgestellt, die dies erlauben. Außerdem werden wichtige Techniken zur Einbindung von CSS-Layouts vorgestellt und erläutert, wie man Grafiken und PDFs erstellen bzw. einbinden kann. Besonders lobend hervorzuheben ist, dass die Autorin sehr viele Beispiele aus der im Web verfügbaren *node.js*-Doku verwendet und diese ausführlich vertiefend und verständlich erläutert, was einen pädagogisch wichtigen Wiedererkennungswert erzeugt. Rundum ein ausgewogenes und klar verständliches Buch, das sich von vorne bis hinten sehr flüssig liest.