Einführung in NodeJS

Name: Jacques Dattler

Matrikelnummer: 4701676

Kurs: WWI 16 SCC

Dozent: Elias Henrich

Inhaltsverzeichnis

Was ist NodeJS?

NodeJS ist eine Laufzeitumgebung für die Programmiersprache JavaScript. Sie basiert auf der V8 Engine, eine Javascript Implementierung von Google, welche auch im Browser Chrome Verwendung findet. Außerdem ist NodeJS eine ereignisgesteuerte Architektur, kann also durch Asynchronität viele Anfragen parallel abarbeiten. Dies ist vor allem bei Eingabe- und Ausgabelastigen Anwendungen hilfreich. NodeJS wird deshalb vor Allem in Serveranwendungen verwendet.

Node steht als Open-Source Software auf der eigenen Webseite für jedem zum Download zur Verfügung und kann in 2 Versionen runtergeladen werden, der LTS-Version (wird lange Zeit unterstützt, erhält Sicherheitsupdates und ändert sich syntaxmäßig nicht) und der Current-Version (bietet die neusten Features).

Hat der Nutzer NodeJS installiert, kann man mit dem Node-Befehl die Laufzeitumgebung starten und Javascript Code ausführen. Hier dazu ein Beispiel:

```
C:\Users\Jac>node
> var x = 10;
undefined
> var y = function(z){ return z+z};
undefined
> console.log(y(x));
20
undefined
>
```

Node Package Manager

Eines der größten Features von NodeJS ist der integrierte Node Package Manager, kurz NPM. Dieser erlaubt die Funktionalität von Node durch aktuell über 650.000 Pakete (Stand Juli 2018) zu erweitern. Bei der Installation von NodeJS wird dieser mitinstalliert und steht über die Kommandozeile zur Verfügung. Installieren kann man die Module mit folgendem Befehl:

```
C:\Users\Jac>npm install express -g
+ express@4.16.3
added 50 packages from 47 contributors in 1.178s
```

Hier lädt NodeJS automatisch das Modul für ExpressJS, ein Webframework für Node, herunter und installiert die Abhängigkeiten für dieses Modul. Der G-Tag, oder global Tag, sagt, dass das Modul global installiert werden soll.

Beispiele für die Verwendung von Node

Paypal:

Paypal soll dank Node 33% weniger Zeilen Code und 40% weniger Dateien brauchen, so Paypal's Ingenieur-Team. Außerdem soll sich die Entwicklungsgeschwindigkeit verdoppelt haben.

Uber:

Uber benutzt schon seit Beginn Die Technik von NodeJS.

Trello:

Für Trello war wegen den vielen Anfragen vor allem die hohe Skalierbarkeit wichtig, um nicht unter der Last zusammenzubrechen. Dank Node wurde dies ermöglicht.

Aber nicht nur die Funktionalitäten von Node, sondern auch die des Node Package Managers haben viele Einsatzmöglichkeiten. NPM dient dazu, andere Dienste, wie z.B. Angular, React oder ExpressJS oder CLI-Schnittstellen für diverse Dienste zu installieren.

Beispiel für einen Webserver in Node:

In diesem Beispiel beschreibe ich, wie man mithilfe von den Modulen ExpressJS und Request eine einfache Middleware mit einem Node-Server erstellen kann.

```
const express = require('express');
const request = require('request');
const app = express();
const router = express.Router();
const port = 3000;

router.get("/api/get1", function(req, res) {
   res.send("Test1");
});

router.get("/api/get2", function(req, res) {
    res.send("Test2");
});

router.get("/route/find/:from/:to", function(req, res) {
    let from = req.params["from"];
    let to = req.params["to"];

    let url = 'http://flights.eliashenrich.de/api.php?action=/route/find&from='+from+'&to='+to;
    request(url, function (error, response, body) {
        res.send(body)
    });
});

app.use(router);

app.listen(port);
```

Mit der Funktion require() kann man die verschiedenen Module einbauen. In diesem Beispiel werden Express und Request eingebunden. Danach wird mithilfe von Express eine Applikation und ein dazugehöriger Router + Port erstellt. Jetzt hat man die Möglichkeit, eigene Pfade für verschiedene GET-Requests anzulegen (im Beispiel /api/get1 und /api/get2) und eine Response dafür festzulegen.

Auch ist es möglich, Parameter in der Url auszulesen (im Beispiel /route/find/:from/:to). Der Node-Server kann hier nun mit den Parametern, welche er von dem GET-Request bekommt, eine weitere Anfrage erstellen. Dies ist mithilfe des Moduls Request möglich und gibt die Antwort von der Anfrage an den Client weiter. Zum Schluss muss man noch der Applikation den Router zuweisen und auf dem vordefinierten Port lauschen.

Mit dem Kommando Node + Dateiname kann man nun den Server starten:

```
C:\Users\Jac\Desktop\Server-Node>node server.js
```

In der Webanwendung kann man nun über die URL des Node-Servers (http://localhost:3000) in Verbindung mit der Verzeichnisstruktur (/route/fing/25/23) seine Anfrage an den Node-Server schicken, welcher die Anfrage als Middleware weiterleitet.

```
function FlightSearchAPI() {
    this.endpoint = 'http://localhost:3000';

    this.requestAPI = function(action, callback) {
        var url = this.endpoint + action;

        var request = new XMLHttpRequest();

        request.onreadystatechange = function() {
            console.log("Status hat sich geändert", this.readyState);

            // Ist die Anfrage beendet?
            if (this.readyState === 4) {
                // Http-Statuscode prüfen
                if (this.status === 200) {
                    console.log("Anfrage erfolgreich");
                    console.log(this.responseText);

                    callback(this.responseText);
                }
            }
        };

        request.open('GET', url, true);
        request.send();
    };
}

var flightSearch = new FlightSearchAPI();
flightSearch.requestAPI('/route/find/25/23', successCallback);
```

Vorteil hierbei ist, dass man strukturelle Änderungen auf dem externen Server nun durch den eigenen Server abfangen kann und man die Software nicht auf dem Client, sondern nur auf dem Server updaten muss.

Quellen:

http://www.modulecounts.com/

https://www.yuhiro.de/vorteile-und-nachteile-von-node-js/

https://nodejs.org/en/

https://de.wikipedia.org/