WMC - Einführung in Node.js und NPM

Node.js ist eine Plattform, die es ermöglicht, JavaScript auf der Server-Seite auszuführen.

Wozu wird Node.js genutzt?

Node.js eignet sich besonders gut für Anwendungen, die schnelle und effiziente I/O-Operationen erfordern. Es ist beliebt für die Entwicklung von:

- **Zugriff auf Dateisysteme**: Node.js kann verwendet werden, um Dateien zu lesen, zu schreiben und zu verwalten.
- **Zugriff auf Datenbanken**: Node.js kann verwendet werden, um Datenbanken wie MongoDB, MySQL und PostgreSQL zu verwalten.
- **Webservern**: Node.js kann verwendet werden, um skalierbare und schnelle Webserver zu erstellen, die HTTP-Anfragen effizient verarbeiten.
- **APIs**: RESTful-APIs und GraphQL-APIs werden oft in Node.js entwickelt, da es leichtgewichtig und asynchron ist.
- **Mikroservices**: Node.js eignet sich hervorragend für die Erstellung von leichtgewichtigen Mikroservices, die auf Container-basierte Architekturen wie Docker und Kubernetes abgestimmt sind.

Merkmale von Node.js

Node.js bietet einige wichtige Funktionen, die es für bestimmte Anwendungsfälle besonders nützlich machen:

- **Asynchron**: Durch seine nicht-blockierende I/O-Natur kann Node.js mehrere Aufgaben gleichzeitig bearbeiten, was es effizient für Anwendungen mit hoher Leistung macht.
- **Single-Threaded**: Trotz der Verwendung eines einzigen Threads für die Verarbeitung verwendet Node.js asynchrone Ereignisse, um gleichzeitig mehrere Verbindungen zu verwalten.
- **V8 Engine**: Node.js verwendet die Google V8 Engine, um JavaScript sehr effizient auf der Server-Seite auszuführen.

Einführungsbeispiel: Ein einfacher HTTP-Server

Node.js wird oft verwendet, um Webserver zu erstellen. Hier ist ein einfaches Beispiel, das einen HTTP-Server erstellt, der auf Anfragen antwortet:

```
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('Hello, World!\n');
});

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

Erklärung

- `require('http')`: Importiert das eingebaute HTTP-Modul von Node.js, mit dem wir einen Server erstellen können.
- `http.createServer()`: Erstellt einen HTTP-Server, der auf eingehende Anfragen reagiert.
- `res.end()`: Sendet die Antwort "Hello, World!" zurück an den Client.
- `server.listen()`: Startet den Server und lässt ihn auf Port 3000 lauschen.

Um den Server auszuführen, speichere das Skript als `server.js` und führe es mit Node.js aus:

```
$ node server.js
```

Dann öffne deinen Browser und rufe `http://127.0.0.1:3000` auf, um die Ausgabe "Hello, World!" zu sehen.

Fazit

Node.js bietet eine schnelle und effiziente Möglichkeit, JavaScript auf der Server-Seite zu verwenden. Es kann als Basis für moderne Webanwendungen, APIs und Echtzeitanwendungen verwendet werden.

NPM: Node Package Manager

NPM (Node Package Manager) ist das Standard-Tool von Node.js zum Installieren, Verwalten und Teilen von Paketen und Bibliotheken. Es ermöglicht Entwicklern, externe Abhängigkeiten (Pakete) einfach in ihren Projekten zu verwenden.

NPM installieren

NPM wird automatisch zusammen mit Node.js installiert. Um zu überprüfen, ob NPM installiert ist und welche Version verwendet wird, kannst du den folgenden Befehl ausführen:

```
$ npm -v
```

NPM-Pakete installieren

Um ein Paket zu installieren, wird der Befehl `npm install` verwendet. Zum Beispiel kannst du das beliebte Web-Framework Express.js installieren:

```
$ npm install express
```

Dies installiert Express.js in deinem Projekt und legt ein Verzeichnis `node_modules/` an, in dem alle Abhängigkeiten gespeichert werden.

Um Pakete global auf deinem System zu installieren (z.B. für CLI-Tools), kannst du das Flag `-g` verwenden:

```
$ npm install -g nodemon
```

Nodemon ist ein Tool, das es ermöglicht, Node.js-Skripte automatisch bei Dateiänderungen neu zu starten.

Datei package.json

Wenn du NPM-Pakete in deinem Projekt installierst, wird die Datei `package.json` erstellt oder aktualisiert. Diese Datei enthält wichtige Informationen über dein Projekt und alle installierten Abhängigkeiten.

Hier ist ein Beispiel für eine einfache `package.json` Datei:

```
{
  "name": "mein-projekt",
  "version": "1.0.0",
  "description": "Ein Beispielprojekt",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "dependencies": {
    "express": "^4.17.1"
  }
}
```

Die `package.json` Datei listet die Abhängigkeiten deines Projekts auf und macht es einfach, das Projekt auf einem anderen Computer zu replizieren. Du kannst alle Abhängigkeiten durch den folgenden Befehl neu installieren:

```
$ npm install
```

Dieser Befehl installiert alle Pakete, die in der `package.json` Datei unter "dependencies" aufgeführt sind.

Einführungsbeispiel: Ein einfacher HTTP-Server mit Express.js

Express.js ist ein minimaler und flexibler Web-Framework für Node.js, das die Erstellung von Webanwendungen und APIs erleichtert. Hier ist ein einfaches Beispiel, das zeigt, wie du einen Webserver mit Express.js erstellen kannst:

Installiere zuerst Express.js in deinem Projekt:

```
$ npm install express
```

Dann erstelle eine Datei `server.js` mit folgendem Inhalt:

```
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => {
  res.send('Hello, World!');
});

app.listen(port, () => {
  console.log(`Server running at http://localhost:${port}/`);
});
```

Erklärung

- `require('express')`: Importiert das Express-Framework.
- `const app = express()`: Erstellt eine neue Express-Anwendung.
- `app.get('/')`: Definiert eine Route, die auf HTTP-GET-Anfragen an der Root-URL (`/`) mit der Nachricht "Hello, World!" antwortet.
- `app.listen(port)`: Startet den Server auf dem angegebenen Port (hier 3000) und gibt eine Bestätigung aus, sobald der Server läuft.

Um den Server auszuführen, speichere das Skript als `server.js` und führe es mit Node.js aus:

```
$ node server.js
```

Dann öffne deinen Browser und rufe `http://127.0.0.1:3000` auf, um die Ausgabe "Hello, World!" zu sehen.