

Übung 2: Node.js

Jülide Cesit |

09. 01. 2023

##### **Was ist node.js?**

Node.js ist eine JavaScript-Laufzeitumgebung, die auf der V8 JavaScript-Engine von Chrome aufbaut. Sie ermöglicht die serverseitige Ausführung von JavaScript und die Erstellung serverseitiger Anwendungen mit JavaScript. Node.js wurde entwickelt, um skalierbare Netzwerkanwendungen zu erstellen und ist besonders nützlich für Echtzeitanwendungen wie Chats, Newsfeeds und Push-Benachrichtigungen.

**Nenne Beispiele für Plattformen, auf denen node.js ausgeführt werden kann.**

Node.js kann auf einer Vielzahl von Plattformen ausgeführt werden, darunter:

Windows, macOS, Linux, Unix,OpenBSD, NetBSD

Node.js kann auch in einem Webbrowser mit Tools wie Browserify oder Webpack

ausgeführt werden.

**Was ist NPM?**

npm (kurz für Node Package Manager) ist ein Paketmanager für die Programmiersprache JavaScript. Es ist der Standard-Paketmanager für die JavaScript-Laufzeitumgebung Node.js. npm hilft bei der Verwaltung und gemeinsamen Nutzung der Pakete (Bibliotheken, Frameworks, Tools usw.), die für ein Projekt benötigt werden, und erleichtert so die Entwicklung und Wartung von Projekten. Mit npm können Sie Pakete aus der npm-Registry, einem öffentlichen Repository für Open-Source-Pakete, installieren und verwalten. npm macht es auch einfach, Ihre eigenen Pakete mit anderen zu teilen.

**Nennen Beispiele für nützliche “Packages”**

**Lodash**: Eine JavaScript-Hilfsbibliothek, die Funktionen für gängige Aufgaben wie die Iteration über Arrays und Objekte, die Bearbeitung von Strings und die Erstellung zusammengesetzter Funktionen bietet.

**Moment:** Eine JavaScript-Bibliothek zum Parsen, Validieren, Manipulieren und Formatieren von Daten.

**Axios:** Eine JavaScript-Bibliothek zur Erstellung von HTTP-Anfragen.

**React:** Eine JavaScript-Bibliothek für die Erstellung von Benutzeroberflächen.

**Webpack:** Ein JavaScript-Bündler, der Sie bei der Verwaltung und Optimierung der Assets in Ihrer Anwendung unterstützt.

**Jest:** Eine JavaScript-Testbibliothek.

**Express:** Ein JavaScript-Framework für die Erstellung von Webanwendungen.

Pug: Eine Template-Engine für HTML.

Babel: Ein JavaScript-Compiler, mit dem Sie JavaScript-Funktionen der nächsten Generation schon heute nutzen können.

**TypeScript:** Eine typisierte Obermenge von JavaScript, die sich zu einfachem JavaScript kompilieren lässt.

**Was ist Express?**

Express ist ein beliebtes JavaScript-Framework für die Erstellung von Webanwendungen auf der Grundlage von Node.js. Es ist leichtgewichtig, flexibel und benutzerfreundlich, was es zu einer guten Wahl für die Erstellung von Webanwendungen und APIs (Application Programming Interfaces) macht.

Eines der Hauptmerkmale von Express ist, dass es Ihnen ermöglicht, eine Reihe von Routen für Ihre Anwendung zu definieren. Eine Route ist eine Kombination aus einem bestimmten URL-Pfad und einer entsprechenden HTTP-Methode (z. B. GET oder POST), die das Verhalten Ihrer Anwendung definiert, wenn diese URL angefordert wird.

Express bietet darüber hinaus eine Reihe weiterer Funktionen, wie z. B. Middleware-Unterstützung, Template-Rendering und Fehlerbehandlung. Aufgrund seiner Einfachheit und Flexibilität ist es eine beliebte Wahl für die Erstellung von Webanwendungen mit Node.js.

# **Erstelle Skripte, um folgende Aufgaben zu lösen. Nutze dafür Node.js. Kopiere den Quellcode direkt in deinen Antwortbogen.**

Lies die Datei “beispieldaten.txt” vom Dateisystem ein.

Schreibe folgendes JSON Objekt in die Datei “*person.json”*:

[

{

    “name”:”Andy”

    “Nachname”:”Warhol”

    “Beruf”:”Künstler”

},

{

    “name”:”unbekannt”

    “Nachname”:”unbekannt”

    “Pseudonym”:”Banksy”

    “Beruf”:”Künstler”

}

]

const fs = require('fs');

// liest "beispieldaten.txt" und ihren Inhalt der Datei

fs.readFile('beispieldaten.txt', (err, data) => {

if (err) {

console.error(err);

return;

}

// writeFile schreibt die angegebenen Daten in "person.json"

fs.writeFile('person.json', JSON.stringify([

{

"name": "Andy",

"Nachname": "Warhol",

"Beruf": "Künstler"

},

{

"name": "unbekannt",

"Nachname": "unbekannt",

"Pseudonym": "Banksy",

"Beruf": "Künstler"

}

]), (err) => {

if (err) {

console.error(err);

return;

}

console.log("JSON object written to person.json");

});

});

Erstelle einen *HTTP Server*, der auf Anfrage über *Port 3456* reagiert

const http = require('http');

const server = http.createServer((req, res) => {

// Set the response status code and content type

res.statusCode = 200;

res.setHeader('Content-Type', 'text/plain');

// Send the response body

res.end('Hello World!\n');

});

// Start the server

server.listen(3456, 'localhost', () => {

console.log('Server running at http://localhost:3456/');

});

# **Lege ein leeres Verzeichnis *“my-simple-server” an sowie eine leere Datei “index.js”* in diesem Verzeichnis. *Welcher Befehl ist erforderlich, um ein NPM Projekt zu initialisieren?***

Öffne das Terminalfenster und navigiere zum Verzeichnis "my-simple-server".

Führe den Befehl npm init aus. Du wirst aufgefordert, Informationen über das Projekt einzugeben, z. B. den Namen und die Version.

Nach Beendigung des Befehls wird eine Datei namens "package.json" im Verzeichnis "my-simple-server" erstellt. Diese Datei enthält Metadaten über das Projekt, wie den Namen und die Versionsnummer.

Führe den Befehl touch index.js aus, um eine leere Datei "index.js" im Verzeichnis "my-simple-server" zu erstellen.

Der Befehl npm init wird verwendet, um ein NPM-Projekt zu initialisieren, indem eine "package.json"-Datei erstellt wird. Der Befehl touch ist ein Unix-Befehl, der eine leere Datei erstellt.

# **Gegeben sei das Programm *“my-simple-server”* mit dem Einstiegspunkt “index.js.”**

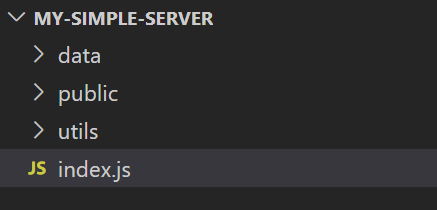
# **Was ist erforderlich, damit durch das Kommando “*npm start”* das Programm ausgeführt wird?**

scripts" der Datei "package.json" aktualisieren.

"scripts": {

"start": "node index.js"

}



# **Füge dem zuvor angelegten Programm *“my-simple-server”* das node.js Framework “Express” hinzu. Nutze dafür npm. Wie lautet das Kommando?**

npm install express --save

# **Wozu dienen die Dateien *“package.json”* und ”*package-lock.json”*?**

Die Datei "package.json" ist eine JSON-Datei, die Metadaten über ein Node.js-Projekt enthält. Sie gibt Dinge wie den Namen des Projekts, die Version, Abhängigkeiten und Skripte an.

Die Datei "package-lock.json" ist eine JSON-Datei, die automatisch von npm erzeugt wird, wenn Sie Pakete in Ihrem Projekt installieren. Sie beschreibt den genauen Baum der Pakete, die installiert wurden, einschließlich der genauen Versionen der einzelnen Pakete und aller ihrer Abhängigkeiten.

# **Sofern du dein Projekt nicht weiter konfiguriert hast, welche Wirkung hat der Befehl “*npm test”*?**

# **Welche HTTP Methoden gibt es?**

# **Welche HTTP Methoden können über ein HTML Formular ausgeführt werden?**