Hochschule
Albstadt-Sigmaringen

Albstadt-Sigmaringen University

Node.JS Crashcourse

Dipl. Ing. Sven Eppler (FH)
sodge IT GmbH

JavaScript escaped from the browser

Was ist Node.js?

- „Standalone“ Version von JavaScript
- Erschienen im Mai 2009
- Basiert auf Googles V8 JS-Engine (Juli 2008)
- Projektseite: https://nodejs.org/en/

Warum Node.js?

- „Write once, run anywhere“
- Cross-Plattform: Linux, Mac, Windows
- Non-Blocking I/O (Async I/O)
- Selbe Sprache im Frontend/Backend

Node.js Installieren

- Projektseite: https://nodejs.org/en/
- Wahlweise die LTS (LongTermSupport) Version oder die latest
- Nach der Installation steht das Kommando „node“ auf der Konsole zur Verüfung
- Node.js liefert keine GUI, nur ein CLI!
- Ohne Startparameter geht Node.js in den REPL-Modus (Read–eval–print loop) wie Python
- Linux/Mac: NVM (Node Version Manager)
  - https://github.com/nvm-sh/nvm

Node.js Anwendung starten

- Node.js Anwendungen haben typischerweise eine „index.js“ oder „app.js“ Datei als Entrypoint
- Diese kann direkt mit node ausgeführt werden
- Siehe „$Repo/Beispiele/Node.js/HelloWorld/app.js“

```
// app.js
console.log("Hello World!");


$ node app.js
Hello World!
```

Node.js Module

- **Node.js ist stark modulasiert und hat ein großes Ökosystem an Modulen**
- **Zur Verwaltung von Modulen gibt es den „Node Package Manager“ npm**
- **Module werden auf npmjs.com bereit gestellt**
  - OpenSource: Jeder kann seine Module dort veröffentlichen

Verwendung von npm

- **Alle notwendigen Module eines Projektes installieren:**
  - npm install
- **Ein Modul zum Projekt hinzufügen**
  - npm install $ModulName
- **Ein Modul deinstallieren**
  - npm uninstall $ModulName

package.json

- **Die Datei „package.json“ definiert:**
  - Wie ein Projekt/Modul heißt
  - Welche Version es hat
  - Metadaten zum Projekt (Autor, Beschreibung)
  - Welche Abhängigkeiten zu andere Modulen es gibt
  - Scripts zum Starten/Testen/etc.
- **Für eigene Projekte kann man mit „npm init“ eine package.json generieren**
- **Mit „npm install“ installiert man alle notwendigen Abhängigkeiten aus der „package.json“**
  - https://docs.npmjs.com/files/package.json

package-lock.json

- **Speichert nach einem lauf von „npm install“ alle installierten Module in ihrer konkreten Version**
- **Problem: Module können Abhängigkeiten „lax“ definieren:**
  - ModulA benötigt v2.0 oder größer von ModulB
  - Solange nur eine v2.0 von ModulB veröffentlich ist, wird auch immer diese installiert
  - Sobald z.B. v2.1 veröffentlicht wird, würde ein Aufruf von „npm install“ die v2.1 installieren
- **Ziel: Konsistente Laufzeitumgebung**

node_modules Ordner

- „npm install“ speichert alle heruntergeladenen Module im Ordner „node_modules“
- Dadurch kann auch „offline“ entwickelt werden
- Der Inhalt des Ordner wird komplett von NPM verwaltet, manuelle Änderungen sollten hier nicht statt finden
- Der Ordner sollte NICHT in die Versionsverwaltung eingecheckt werden

package.json Scripts

- In der Datei „package.json“ kann man im Scripts-Abschnitt kleine Scripte ablegen.
- Diese Script kann man mit „npm run $ScriptName“ ausführen

```
{
  "name": "dependenciesexample",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "my-script": "echo Output of my-script"
  },
}
```

Minimales Express Beispiel

- **Express ist das defacto Standard WebFramework für node.js**
- **Beispiel: „Beispiele/Node.js/MinimalExpress“**
  - Projekt mit „npm install“ startklar machen
  - Anwendung mit „npm start“ starten
  - Browser auf „localhost:3000“