NPM

Node Package Manager

- Paketmanager für die JavaScript-Laufzeitumgebung Node.js
- unter npm Registry bzw. npm Open Source wird ein Repository mit über 350.000 Paketen betrieben

Pakete finden

```
npm search <searchKey>
```

oder auf https://www.npmjs.com/

Module installieren

Lokal

```
npm install <package-name>
```

Ordner `node_modules` wird erstellt und beinhaltet das Package.

global

```
npm install <package-name> -g
```

Ist nicht im `node_modules` Ordner.

Module deinstallieren

Lokal

```
npm uninstall <package-name>
```

global

```
npm uninstall <package-name> -g
```

Module updaten

```
npm update <package-name>
```

Installierte Module auflisten

```
npm list
```

Bibliotheken einbinden

```
const modulname = require('modulname')
```

Alle Basismodule hier: https://nodejs.org/docs/latest-v9.x/api/

package.json

Um eigene Module entwickeln zu können, egal ob für den internen oder öffentlichen Gebrauch, muss sich eine Datei namens `package.json` pro Modul anlegen. Es enthält wichtige Metadaten zum Modul.

```
{
  "name": "modulname",
  "description": "Beschreibung des Modules.",
  "author": "Kevin Hertwig <hertwig@campus.tu-berlin.de>",
  "homepage": "github.com/kevinhertwig/modulname",
  "bugs": "github.com/kevinhertwig/modulname/issues",
  "version": "0.3.0",
  "license": "MIT",
  "keywords": [
    "blog",
    "markdown",
    "live-update",
    "documents",
    "ideas",
    "documentation",
    "specification"
  ],
  "repository": {
    "type": "git",
    "url": "git://github.com/kevinhertwig/modulname.git"
  },
  "dependencies": {
    "mime": ">=1.2.4",
    "connect": ">=2.0.0",
    "marked": ">=0.1.2",
    "optimist": ">=0.3.0",
    "qs": ">=0.4.0",
    "winston": ">=0.5.9"
```

```
  },
  "scripts": {
    "start": "node server.js -s"
  },
  "engines": {
    "node": ">= 0.6.0"
  },
  "devDependencies": {}
}
```

Neben Angaben zum Namen, Beschreibung, Autor, Version, Lizenz und Keywords des Modules befinden sich einige sehr hilfreiche Definitionen in dieser Datei:

Dependencies

Diese definieren die Abhängigkeiten des Modules, die via NPM möglich sind und zusammen mit dem Modul installiert werden sollen. Für die Angabe der abhängige Version stehen mehrere Varianten zur Verfügung:

- version: Muss exakt version entsprechen
- =version: Entspricht version
- >version: Größer als die definierte Version
- >=version: Größer bzw. gleich der definierten Version
- <version: Kleiner als die definierte Version
- <=version: Kleiner bzw. gleich der definierten Version

> Als Abhängigkeit kann auch ein Verweis zu einem Repository angegeben werden.

devDependencies

Hier werden die referenzierten Module definiert, die für die Entwicklung an diesem Modul notwendig sind. Beispielsweise betrifft dies Abhängigkeiten zu Testing-Modulen, die für die Entwicklung notwendig, für die Verwendung jedoch unerheblich sind. Standardmäßig werden alle Abhängigkeiten installiert. Um sicher zu gehen, dass die für die Entwicklung notwendigen Abhängigkeiten nicht installiert werden, sollte das Modul mit folgendem Aufruf installiert werden:

```
npm install –production
```

Scripts

Hier können für unterschiedlichste Fälle Scripts angegeben werden. Unterstützt werden folgende Einträge:

- preinstall: Werden vor der Installation ausgeführt
- install, postintall: Werden nach der Installation ausgeführt
- preuninstall, uninstall: Werden vor der Deinstallation ausgeführt
- postuninstall: Werden nach der Deinstallation ausgeführt
- preupdate: Laufen vor der Aktualisierung
- update, postupdate: Laufen nach der Aktualisierung
- prepublish: Laufen vor der Veröffentlichung des Paketes

- publish, postpublish: Laufen nach der Veröffentlichung des Paketes
- pretest, test, posttest: Werden durch den Aufruf von `npm test` ausgeführt
- prestop, stop, poststop: Werden durch den Aufruf von `npm stop` ausgeführt
- prestart, start, poststart: Werden durch den Aufruf von `npm start` ausgeführt
- prerestart, restart, postrestart: Werden durch den Aufruf von `npm restart` ausgeführt. Wird kein `restart`-Script angegeben, wird zuerst `stop` und dann `start` durchlaufen.

Veröffentlichen von Modulen

Registrierung auf https://www.npmjs.com/ mit einem Benutzer, welcher via

```
npm add-user
```

der lokalen Registrierung hinzugefügt werden muss. Danach kann das Modul (package.json muss vorhanden sein) via

```
npm publish <Verzeichnis>
```

veröffentlicht werden. Wahlweise kann auch eine Url oder ein Tar-Archiv verwendet werden. Sollte es Probleme beim Aktualisieren geben (Version existiert bereits) kann ein Überschreiben via –force durchgeführt werden.