












Blockwoche: Web Programming Lab

Onboarding – Programm Blockwoche

Montag 	Dienstag  	Mittwoch  	Donnerstag    	Freitag  
Architekturansätze von Web Anwendungen JavaScript Sprachkonzepte I	Client-Side- Javascript I	Angular	Angular	Mobile Apps mit Web-Technologien
JavaScript Sprachkonzepte I & II	Client-Side- Javascript II Frameworks & Typescript	Angular	Server-Side- JavaScript REST	Security @ Web Apps

Agenda

- **Intro Node.js**
- **Intro Node.js HTTP Module & Express**
- **Intro MongoDB Anbindung via Node.js**
- **REST Architekturparadigma**

Server-Side JavaScript

Intro Node.js Konzepte



Node.js – Übungsaufgabe I

1. Gib das Folgende auf der Kommandozeile aus:

> “Hallo Web-Programming-Lab nach 4 Sekunden”

> “Hallo Web-Programming-Lab nach 8 Sekunden”

<https://gist.github.com/bitmuggler/faf104b597046675491235ddaefc87cd>

2. Gib das Folgende wiederholend nach 3 Sekunden aus:

> “Hallo Web-Programming-Lab nach 3 Sekunden”

<https://gist.github.com/bitmuggler/afdba6f962bd23a30b73d7c4af8aab5e>

3. Gibt das Folgende wiederholend nach jeder Sekunde aus aber nur 5 Mal:

> “Hallo Web-Programming-Lab”

... anschliessend gib “Done” aus und ermögliche einen Exit.

Rahmenbedingung: Kein setTimeout nutzen.

<https://gist.github.com/bitmuggler/8b89b779e4e17f20c6fff609dad9977b>

Node.js – Übungsaufgabe II (10')

- **Schreibe ein CommonJS Modul und ein ECMAScript Module, welches jeweils die Datei kopiert.**
- Kopiert ein File mit der Endung .copy
- **Rahmenbedingungen:** Mit Promises / async await gelöst!
- Interfaces:

```
const filecopy = require('./filecopy.cjs');  
filecopy('filecopy-commonjs.js'); // => filecopy-commonjs.js.copy
```

exercises/filecopy-commonjs.js

```
import {filecopy} from './filecopy.mjs';  
filecopy('filecopy-esm.mjs'); // => uebung-filecopy-esm.mjs
```

exercises/filecopy-esm.mjs

- Lösung cjs: <https://gist.github.com/bitmuggler/f2bb537aefc285b8228cdc7ed0234d69>
- Lösung mjs: <https://gist.github.com/bitmuggler/30f29439232661e269a7467a07a1e34e>

Server-Side JavaScript

Http Module von Node.js, Express, Debugging & Persistieren mit MongoDB

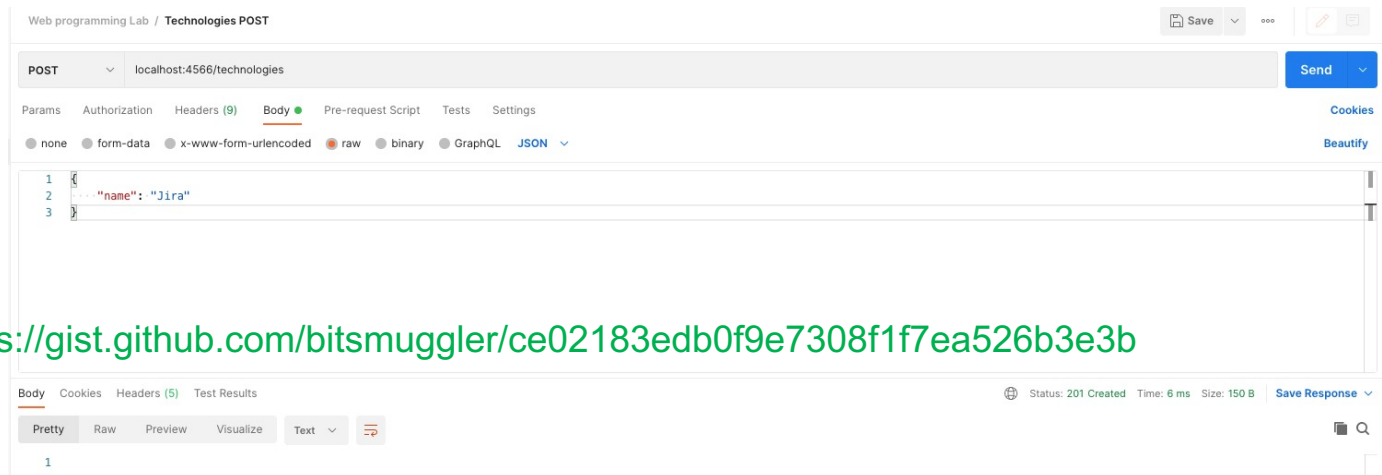


Node.js – Übungsaufgabe (20')

Node.js App mit den folgenden HTTP APIs erstellen (Speichern Sie die Werte In-Memory, z.B. in einem Array):

- POST /technologies → speicherte eine Technologie in der Liste
- GET /technologies → gibt die Technologien Liste zurück
- GET /technologies/{id} → gibt das Detail einer Technologie zurück
- Verzeichnis exercises / tech-radar verwenden (benötigte Dependencies sind bereits deklariert)

Via [Postman](#) können die APIs getestet werden.



- Lösung: <https://gist.github.com/bitsmuggler/ce02183edb0f9e7308f1f7ea526b3e3b>

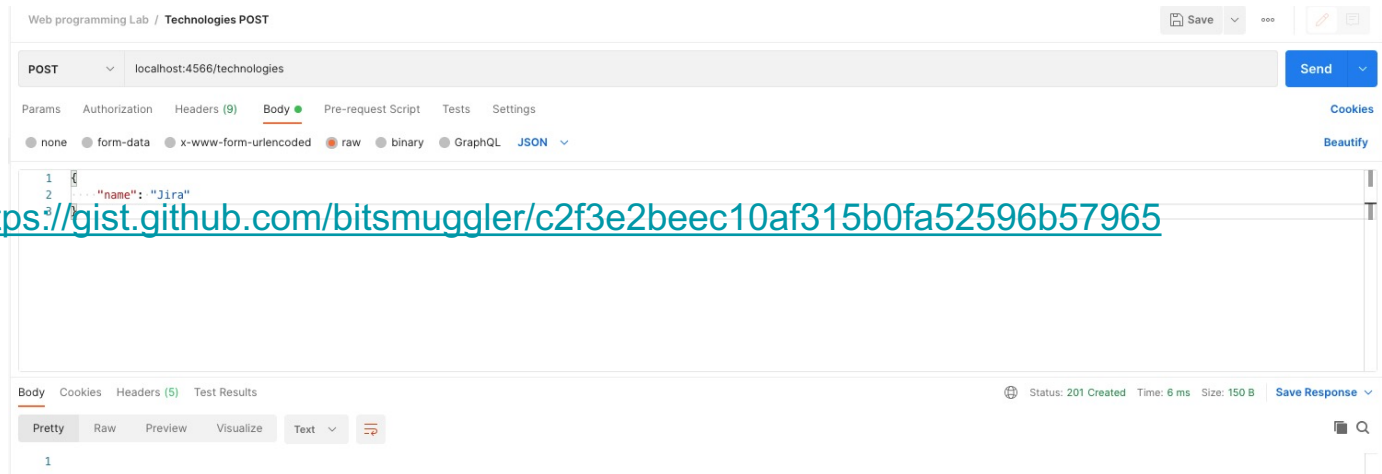
Node.js – Übungsaufgabe II (20')

Ergänzen Sie nun Ihre Node.js App mit Persistenz (der Fokus liegt auf dem Durchstich):

- POST /technologies → persistiert eine Technologie in der MongoDB (falls geklappt Return Code 201)
- GET /technologies → gibt die Technologien aus der DB zurück (falls geklappt Return Code 200)
- GET /technologies/{id} → gibt das Detail der selektierten Technologie zurück
- Verzeichnis exercises/mongodb verwenden (benötigte Dependencies sind bereits deklariert)

Via [Postman](#) können die APIs getestet werden.

- Lösung: <https://gist.github.com/bitsmuggler/c2f3e2beec10af315b0fa52596b57965>



Server-Side JavaScript

API Integration Testing



Node.js – Übungsaufgabe (20')

Teste Sie nun Ihre Node.js App mit mindestens einem Test pro Verb mit dem vorgestellten API Integration Testing Toolkit – SuperTest und Jest.

- Lösung: <https://gist.github.com/bitsmuggler/77195512a22585fc854c9b9bfd2e83d7>