Web Engineering 2, Übung 1: Node.js

Inhaltsverzeichnis

Michael Gfeller <michael.gfeller@ost.ch>

Installation & Vorbereitung

1. Installieren Sie die LTS Version von Node.js.
2. Installieren Sie Webstorm oder eine alternative IDE.
3. Optional: Installieren Sie nvm bzw nvm-windows.

Aufgabe 1: Hello World Server

Diese Aufgabe soll überprüfen ob alles richtig installiert/konfiguriert ist.

1. Erstellen Sie einen Server, welcher immer "Hello World" ausgibt. Nutzen Sie dafür https://nodejs.org/api/http.html.
2. Starten Sie den Server über die Entwicklungsumgebung starten.
3. Debuggen Sie den Aufruf.

NOTE | Lösung: Solution/hello-world.mjs

Aufgabe 2: Hello World Server - Version 2

Erstellen Sie einen Server, welcher immer die angefragte URL ausgibt.
z.B. http://localhost:3000/Test5105 Ausgabe: "requested /Test5105".

NOTE | Lösung: Solution/hello-world-2.mjs

Aufgabe 3a: Module - 1

Erstellen Sie ein Module, welches die Zahlen [VON, BIS] ausgibt. Der Start und End-Wert können dem Module übergeben werden.

- Importieren Sie das Module in ein anders und rufen Sie dieses auf.
- Optional: Versuchen Sie das Problem ohne Schleifen (while, for...) zu lösen

NOTE | Lösung: Solution/Routing/number.mjs

Aufgabe 3b: Module - 2

Erstellen Sie ein Module, wessen API ein Filename und einen Text erwartet.

- Erstellen Sie dieses File mit dem übergebenem Text.
- Lesen Sie danach das File.
- Löschen Sie danach das File und geben Sie den gelesen Wert wieder zurück.
- Behandeln Sie auch den Fehlerfall.
- Variante 1: Nutzen Sie die API: https://nodejs.org/api/fs.html
- Variante 2: mit async / await: https://nodejs.org/api/fs.html#fs_promises_api
- Importieren Sie das Module in ein anders und rufen Sie dieses auf.

NOTE

Lösung

Variante 1: Solution/Routing/file.mjs
Variante 2: Solution/Routing/file_v2.mjs

Aufgabe 4: Routing 1

Schreiben Sie einen Node-Server, welcher mit je nach URL eine eine Antwort erzeugt:

NOTE

Die Aufgaben Routing 1/2 kann mit callbacks und/oder async / await gelöst werden.

- Route: /numbers
  - Senden die Zahlen 0 bis 50 zum Client.
  - Nutzen Sie dafür Module 1
  - Mit Query-Parameter (von / bis) soll der Bereich definiert werden können z.B. /numbers?min=10&max=50. Finden Sie die dafür geeignetste API-Funktionalität.
- Route: /file
  - Beim Aufruf soll ein File erstellt werden mit dem aktuellen Datum. Dieser Wert soll ausgelesen werden und zum Client geschickt werden. Am Ende soll das File wieder gelöscht werden.

- Nutzen Sie dafür Module 2

NOTE

Lösung

Callback: Solution/Routing/server.mjs
Async Await: Solution/Routing/server-await.mjs

Aufgabe 5: Routing 2

Ergänzen Sie den Server von Routing 1 wie folgt:

Wenn "/to-send-html.html" angefragt wird, soll dessen HTML übertragen werden; dieselbe Funktionalität soll auch für das JavaScript File implementiert werden (to-send-js.js).

NOTE

to-send-html.html und to-send-js.js befinden sich Ordner Vorlage.

Eine Anfrage an "/to-send-html.html" kann so mit folgenden Schritten beantwortet werden:

- Laden Sie das HTML vom File-System mit 'fs'.
- Schreiben Sie den richtigen Content-Type in das Response-Object: res.writeHeader(200, {"Content-Type": "text/html"});
- Schreiben Sie den den Inhalt des html's in das Response-Object: res.write(html)
- Beenden Sie die Response: res.end();
- Verwenden Sie dieselbe Vorgehensweise für "/to-send-js.js"
- Vermeiden Sie Copy and Paste

NOTE

Lösung

Callback: Solution/Routing/server.mjs
Async Await: Solution/Routing/server-await.mj