OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG

INFORMATIK UND MATHEMATIK

Übungsaufgaben Blatt 10 - Serverseitige Entwicklung mit Node.js

Medieninformatik Webtechnologien - Prof. Dr. Markus Heckner

Aufgabe 1 : Aufsetzen der eigenen Node.js Installation

Um JavaScript-Anwendungen auf dem Server ausführen zu können, benötigen Sie die Node.js Runtime auf Ihrem Rechner.
Gehen Sie wie folgt vor, um Ihren Rechner (im CIP Pool) mit Node.js zum Laufen zu bringen:

1. Laden Sie sich aus Grips die Datei `Node_self_contained.zip` herunter und Kopieren Sie diese in das Wurzelverzeichnis Ihres G-Laufwerks, d.h. direkt nach `G:`

2. Entpacken Sie die Datei durch Rechtsklick, dann *7-Zip*, dann *Hier entpacken*. Sie sollten die folgende Verzeichnisstruktur erhalten:
   `G:\Programme\nodejs`.

3. Testen Sie, ob Node läuft: Starten Sie eine Eingabeaufforderung durch Klick auf *Start* und dann *Eingabeaufforderung* und wechseln Sie in das gerade entpackte Verzeichnis mit den folgenden Kommandos:

   ```
   G:
   cd Programme\nodejs
   node
   ```

   Wenn alles korrekt installiert wurde, dann springt die Kommandozeile in eine neue Zeile und Sie können JavaScript Befehle eingeben (d.h. Sie erhalten eine neue Kommandozeile / Shell, die JavaScript Befehle versteht). Testen Sie das durch Eingabe von:

   ```
   console.log("Hooray, Node successfully installed!");
   ```

   Verlassen Sie die Node-Shell wieder durch zweimaliges Drücken der Tastenkombination *Strg+C*.

4. Node.js ist jetzt korrekt installiert, Sie müssen aber in Ihrer Kommandozeile immer in das Verzeichnis `G:\Programme\nodejs` wechseln, wenn Sie Node.js ausführen wollen. Geben Sie den folgenden Code auf der Kommandozeile ein, um Befehle in diesem Verzeichnis aus allen Verzeichnissen auf Ihrem Rechner ausführen zu können:

   ```
   set path=%path%;G:\Programme\nodejs
   ```

Achtung: Diese Änderung bezieht sich nur auf die aktuell geöffnete Eingabeaufforderung. Sobald Sie das Fenster schließen oder den Rechner neu starten, müssen Sie das obige Kommando erneut eingeben.

Wechseln Sie jetzt auf ein beliebiges anderes Verzeichnis, z.B. durch Eingabe der folgenden Kommandos:

```
cd ..
cd ..
dir
```

Testen Sie jetzt, ob das Node-Verzeichnis korrekt der Pfadvariable hinzugefügt worden ist. Die Eingabe von `node` sollte jetzt zum selben Ergebnis führen wie vorher.

Sie haben Node.js erfolgreich installiert und können Node.js jetzt aus jedem Verzeichnis starten!

Aufgabe 2 : **Eigenen Node.js Server starten: Hello Counter**
In dieser Aufgabe sollen Sie einen ersten, sehr einfachen, http Server erstellen. Der Server soll auf eingehende http Requests (d.h. Aufrufe durch einen Webbrowser) warten und dann einen Text zurückgeben. Der Server `merkt` sich, wie oft er aufgerufen wurde und gibt ein entsprechendes Ergebnis zurück.

Für diese Aufgabe steht Ihnen schon ein fertiger Code (`helloCounter.js`) für den Server auf Grips zur Verfügung.

Gehen Sie wie folgt vor, um den Server zum Laufen zu bringen:

- Laden Sie die Datei `helloCounter.js` aus Grips und speichern diese in einem geeigneten Verzeichnis ab
  (z.B: `G:\Projects\mi\nodecounterhttp\HelloNodeHttpCounter.js`).
- Öffnen Sie eine Eingabeaufforderung und navigieren Sie dort zum gerade erstellten Verzeichnis. Starten Sie den http Server mit dem folgenden Kommando:

  ```
  node HelloNodeHttpCounter.js
  ```

- Testen Sie Ihren Server im Browser.

Aufgabe 3 : **Node.js Server mit verschiedenen Clients testen**
In dieser Aufgabe sollen Sie den zuvor erstellten http-Server mit verschiedenen Browsern testen. Gehen Sie wie folgt vor:

- Finden Sie die IP-Adresse des CIP-Poolrechners mit Hilfe des folgenden Kommandozeilenaufrufs heraus: `ipconfig -all`.

- Versuchen Sie jetzt auf den Server Ihres Nachbarn zuzugreifen indem Sie die URL in Ihrem Browser umstellen, so dass Sie jetzt den Server mit der IP-Adresse des Nachbarn aufrufen.
- Testen Sie die Ausgabe im Browser.

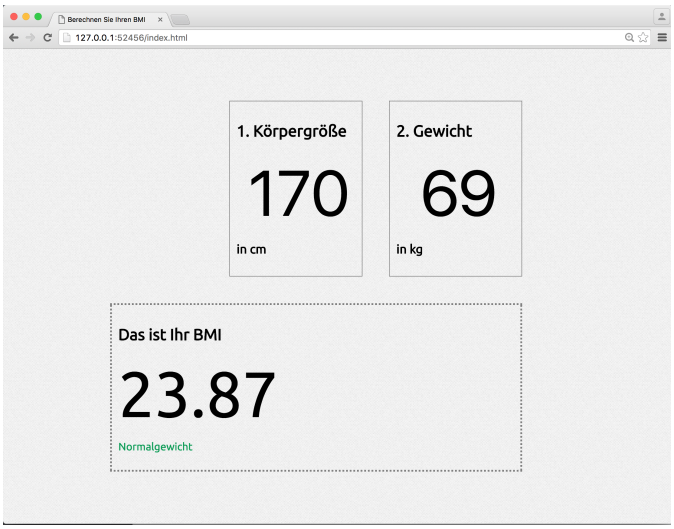
Aufgabe 4 : **Automatischer Neustart des Node-Servers bei Änderungen des Quellcodes: Installation des pakets supervisor**
Die Änderungen am Quellcode werden erst beim Neustart von Node übernommen. Mithilfe des Moduls `supervisor` überwacht Node.js laufend die Dateien des Projekts auf Änderungen und startet den Server bei allen Änderungen neu.
Rufen Sie die Website `https://www.npmjs.com/package/supervisor` auf und installieren Sie das Paket wie beschrieben. Starten Sie anschließend Ihren Server mit `supervisor` anstatt mit `node` neu.
Ändern Sie anschließend den Quellcode und überprüfen Sie, ob Ihr http-Server neu startet.

Aufgabe 5 : **BMI Calculator *goes REST***
In einer früheren Übung haben Sie einen clientseitigen BMI-Rechner erstellt. In dieser Aufgabe sollen Sie diese Berechnung (d.h. die gesamte *Business-Logik*) vom Client auf den Server auslagern. Der Client übergibt jetzt nur noch Größe und Gewicht und aktualisiert die Darstellung, sobald der Server die Antwort zurückgeliefert hat.

Für diese Aufgabe finden Sie in Grips bereits ein Starterprojekt mit den folgenden Inhalten:

- Vollständiger HTML / CSS Code zur Anzeige des BMI-Rechners im Browser
- Vollständiger clientseitiger JavaScript-Code zur Anfrage des BMI an den Server und zur Verarbeitung der Anfrage.
- Verzeichnis `server` mit einer Datei `server.js` - Implementieren Sie hier Ihren REST-Server. In diesem Verzeichnis finden Sie auch ein Modul `bmiCalculator.js`, das die BMI-Berechnung beinhaltet. Bedienen Sie sich aus dieser Datei, um die BMI-Berechnung auf dem Server durchzuführen. Für diese Lösung dürfen Sie den kompletten Code des Servers in die Datei `server.js` schreiben.

Die fertige Lösung sollte sich für den Nutzer identisch verhalten, wie die clientseitige Lösung. Vergleichen Sie dazu den folgenden Screenshot:



Aufgabe 6 : **Nutzung der offiziellen Node.js Dokumentation**

In dieser Aufgabe sollen Sie die offizielle Node.js Dokumentation kennenlernen. Diese Dokumentation finden Sie online unter dem folgenden Link:
`https://nodejs.org/api/index.html`.
Das Objekt *request* erhalten Sie immer dann, wenn der *http*-Server eine neue Anfrage von einem Client erhält. Finden Sie anhand der offiziellen Node.js Dokumentation heraus, wie Sie aus einem *request* die folgenden Informationen ablesene können:

- Die URL des Requests (welche URL wollte der Client aufrufen)).
- Die vom Client an den Server übergebenen Daten.
- Pfad der URL, d.h. alle Informationen nach dem Hostnamen und vor dem Querystring.