BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Übungsblatt 2

Veröffentlicht am	21.04.2017
Anzahl der Seiten	3
Anzahl Punkte im Pflichtteil (entspricht maximal erreichbaren Punkten)	10
Anzahl Punkte im Bonusteil	5
Abgabetermin und Demonstration in der Übung	Übungen 04./09.05.2017

Anmerkungen

Bitte beachten Sie die folgenden Hinweise zur Bearbeitung der Übungsaufgaben und dem Ablauf im aktuellen Semester.

- Lesen Sie bei einem Übungsblatt stets alle Aufgaben durch, bevor Sie beginnen.
- Nach dem Unterricht wird vor der Übung das ggf. neue Übungsblatt in Moodle veröffentlicht.
- Sofern zum Aufgabenblatt Code-Bausteine (Vorgaben) dazugehören, werden diese ebenfalls auf Moodle zum Download angeboten und sind als Ausgangsbasis bei der Bearbeitung zu verwenden.
- Ihre Lösung der Aufgaben laden Sie ~*vor*~ Ihrer persönlichen Demonstration in Moodle hoch. Dateiname: Ü[Nr]__Nachnamen__Matrikelnummern.ZIP
  Beispiel: Ü2__Mueller_Meier__12345678__87654321.ZIP
- Erfordern die Teil-Aufgaben eines Übungsblattes, dass Sie mehrere Anwendungen, HTML-Seiten oder Code-Pakete erstellen, dann legen Sie bitte Unterordner in Ihrem ZIP mit den Nummern der Aufgaben an.
- Persönliche Demonstration und Erklärung in der Übung durch alle Gruppenmitglieder. Jedes Gruppenmitglied kann die Abgabe erläutern, sonst keine Punkte.
- Bei den Aufgaben ist jeweils angegeben, ob diese Pflicht- oder Bonus-Aufgaben sind, sowie die maximal erreichbaren Punkte der Teil-Aufgabe.
- Eine Übung gilt als bestanden, wenn mind. 50% der Pflichtpunkte erreicht wurden, sonst gibt es 0 (Null) Punkte.
- Bei verspätete Abgabe von bis zu maximal 2 Wochen können nur noch 50% der möglichen Punkte des Übungsblattes erreicht werden (bei einer Woche verspäteter Abgabe 75% der möglichen Punkte).

Ziel und Zweck der Übung:

Sie können nach der Übung eine eigene Installation von `node.js` aufsetzen und diverse Pakete mittels der Paketmanager `npm` bzw. `yarn` installieren. Sie verwenden die `package.json` Deklaration für Abhängigkeiten des Paketmanagements. Es ist Ihnen möglich einen ersten minimalistischen Webserver in `node.js` mittels des Frameworks `express` zu entwickeln, der statische Dateien als Antwort auf Anfragen (Requests) ausliefert.

Referenzen:

- https://nodejs.org/
- http://expressjs.com/
- https://docs.npmjs.com/files/package.json
- https://yarnpkg.com/en/docs/usage
- https://nodejs.org/api/fs.html#fs_fs_readfile_filename_options_callback (Bonusteil)
- https://de.wikipedia.org/wiki/Memoisation (Bonusteil)

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

Aufgabe 0 Bonusaufgabe (Optional, keine Punkte)

Installieren Sie die Entwicklungsumgebung WebStorm von JetBrains und erwerben Sie dafür eine Studierenden-Lizenz für kostenlose Nutzung. (Link: siehe Moodle)

Aufgabe 1 (Pflicht, 5 Punkte)

1. Installieren Sie `nodejs`, sofern es noch nicht verfügbar ist[1].
2. Installieren Sie den `yarn` Paketmanager global (bspw. mittels `npm` wie im Unterricht gezeigt)
3. Legen Sie einen Projektordner im Dateisystem für das Übungsblatt 2 an (Name beispielsweise `.\U2`). Installieren in diesem neuen Projekt-Ordner mittels des `yarn` Managers das Erweiterungsmodul `express`.
4. Legen Sie die Datei `app.js` an und schreiben Sie darin den Servercode für Ihre erste Hello-World-Serveranwendung.

Die Aufgabe 1 gilt als erfüllt, wenn Ihr mit `nodejs` geschriebener Server nach dem Starten auf alle Anfragen (HTTP Requests) immer die gleiche HTML5-Webseite ausliefert, welche nur eine Überschrift „Hello World“ enthält.

Aufgabe 2 (Pflicht, 1 Punkt)

Editieren Sie die `package.json` Informationen für Ihr Projekt entsprechend der `nodejs` Dokumentation (siehe Referenzen). Ergänzen Sie alle relevanten Informationen, als würden Sie ihre Serverlösung anschließend als Paket veröffentlichen wollen. Begründen Sie bei Abgabedemonstration, warum die gemachten Angaben relevant sind. (Die Datei sollte mind. sieben Angaben enthalten).

Aufgabe 3 (Pflicht, 4 Punkte)

Erweitern Sie Ihren node.js Servercode mit dem Ziel, dass beim Abrufen von URLs mit dem Pfad-Prefix `/staticfiles/` entsprechende statische Dateien aus dem Unterordner `.\public` ausgliefert werden. *Kopieren Sie dazu Ihre Dateien der Lösung von Übungsblatt 1 in dieses Verzeichnis (oder andere statische Inhalte).*
Beispiel: Sie fragen mit dem Browser `http://localhost:3000/staticfiles/index.html` an und die statische Seite `index.html` aus dem Verzeichnis `.\public` wird ausgeliefert.

Aufgabe 4 Bonusaufgabe (Optional, 1.5 Punkte)

Finden Sie heraus, wie Sie mit `nodejs` bzw. `express` den Content-Type im HTTP-Header der zurückgelieferten Antwort von `text/html` auf `text/plain` ändern. Nutzen Sie die Lösung um auf einer neuen Route `/time` die aktuelle Systemzeit als `text/plain` zurückzuliefern.
Beispiel: `http://localhost:3000/time` liefert dann die aktuelle Systemzeit des Servers.

Aufgabe 5 Bonusaufgabe (Optional, 3.5 Punkte insgesamt)

5.a Bonusaufgabe (Optional, 2 Punkte)

Finden Sie heraus, wie Sie mit `node.js` Inhalte aus einer Datei auslesen (nicht blockierend! also asynchron) und liefern Sie den Inhalt einer Textdatei auf einer neuen Route `/file.txt` aus. Senden Sie dabei als letzte Zeile zusätzlich die Zeit in Nanosekunden[2], die für das I/O-Auslesen der Datei benötigt wurde.

---

[1] Testen Sie ggf. mit *node* oder *npm* ob diese Programme installiert sind (mindestens Version 4.6). Sollte *node* oder *npm* trotz Installation nicht gefunden werden, fügen Sie es noch dem Pfad ihrer Umgebungsvariablen hinzu.
[2] Dazu können Sie kein Date-Objekt benutzen, da es die Zeit nur in Millisekunden liefert. Schauen Sie, wie Sie die globale node.js Variable *process* nutzen können, um Nanosekunden zu messen.

5.b Bonusaufgabe (Optional, 1.5 Punkte)

Nutzen Sie das Memoization Pattern (siehe Referenzen) um das wiederholte Auslesen der Datei zu sparen, wenn die Route `/file.txt` öfter aufgerufen wird. Anhand der geringeren Zeit für das Auslesen (letzte Zeile Ihrer Antwort) sehen Sie, wieviel schneller dieser Ansatz ist. Erläutern Sie bei Abgabedemonstration, welchen Nachteil das Memoization Pattern hat. Warum wird das Abrufen der Datei auch ohne Memoization (Aufgabe 5.a) schneller?

Hinweis zu Aufgabe 1, 3, 4,5 zusammen

Die URLs zum Abrufen der Lösungen sollen alle gleichzeitig funktionieren in einer `app.js`-Datei.
Mit *alle Anfragen* sind in Aufgabe 1 also alle gemeint bis auf die HTTP-Anfragen aus Aufgaben 3-5, die natürlich parallel auch funktionieren sollen.
Beispiel: http://localhost:3000/staticfiles/index.html liefert also aus .\public die index.html, während gleichzeitig http://localhost:3000/time die Zeit als text/plain liefert und http://localhost:3000/file.txt liefert den Dateiinhalt.