Dokumentation Webengineering – 4793868

Matrikelnummer: 4793868

Inhaltsverzeichnis

Was wurde bearbeitet

Es wurden die folgenden Elemente bearbeitet

1. Anzeige von Aktien Verlaufskurven [1]
9. Benutzer Login mit dem Einsatz von Cookies (ein Passwort für alle Nutzer) [2]
16. Eigene Page mit völlig eigenen Inhalten, einige Elemente implementiert [1]
17. Installation der Komponenten läuft über Docker [1]

Gesamt: [5] Punkte

Erfahrungsbericht

Gut funktioniert

Prinzipiell hat der Aufbau einer HTML Seite und das Hinzufügen von CSS gut funktioniert, da die Vorlesung viel Basisinhalt dazu vermittelt hat. Dazu hilft bei den meisten Problemen oder Herausforderungen die W3School-Webseite.
JavaScript war schon etwas herausfordernder. Das Arbeiten mit dem DOM und modifizieren von CSS oder Inhalten via innerText/innerHTML hat relativ gut und intuitiv funktioniert.
Die Einbindung von Bootstrap war simpel und hat den Aufbau eines Responsive-UI erleichtert.

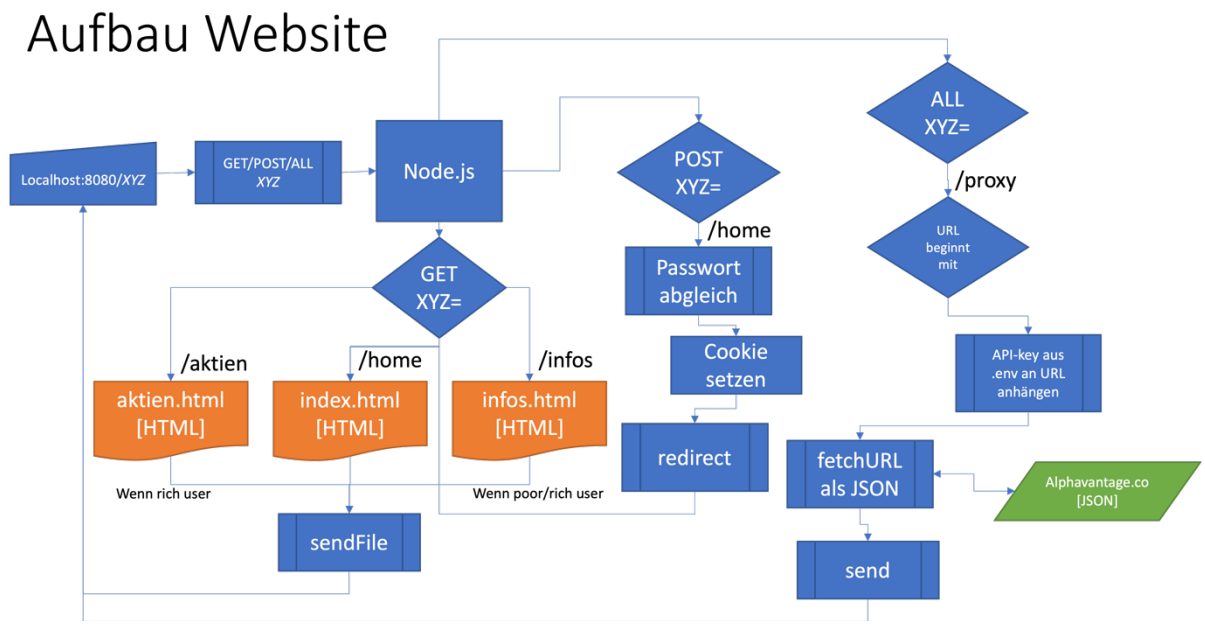
Schlecht funktioniert

Schlecht beziehungsweise komplizierter war es mit JavaScript in Node.js zu arbeiten. Dort neue Anweisungen hinzuzufügen, z.B. die Reaktion auf einen POST von einem Formular oder auch die Bearbeitung von Proxy-Call und den damit erhaltenen JSON-Inhalten.
Dazu war es auch herausfordernd mit Docker zu arbeiten, wie auch

Was habe ich gelernt

HTML und CSS war davor schon bekannt.
JavaScript wurde durch die Vorlesung und die Übungen sehr weit ausgebaut. Es war nicht immer leicht, aber meine Fähigkeiten in JavaScript haben sich nach eigenem Ermessen auf jeden Fall verbessert.
An Node.js habe ich mich vor der Vorlesung nie herangetraut, da es mir immer zu komplex erschien. Die Vorlesung hat mir sehr geholfen mich damit mehr zu beschäftigen und auch bei komplexeren Problemen nicht aufzugeben.
Auch wenn die Arbeit mit Node.js nicht einfach war, war es auf jeden Fall ein Erfolg.

Datenflussdiagramm



Readme zur Installation

Auch auf GitHub verfügbar: https://github.com/LeGlim/DHBW-WE-Webprojekt_4793868

Wenn mit Docker gearbeitet wird:

1. GitHub Repository pullen / Ordner aus E-Mail kopieren.
2. Terminal öffnen, während man sich im Ordner befindet.
3. Sicherstellen, dass Docker installiert und ausführbar ist.
4. Docker Image erstellen
    a. Im Terminal: „docker build . -t yourcontainernamehere“
5. Docker Image ausführen
    a. Im Terminal: „docker run -p 8080:6001 yourcontainernamehere“
6. Auf Webserver zugreifen
    a. Im Webbrowser: http://localhost:8080

Wenn ohne Docker / mit npm gearbeitet wird:

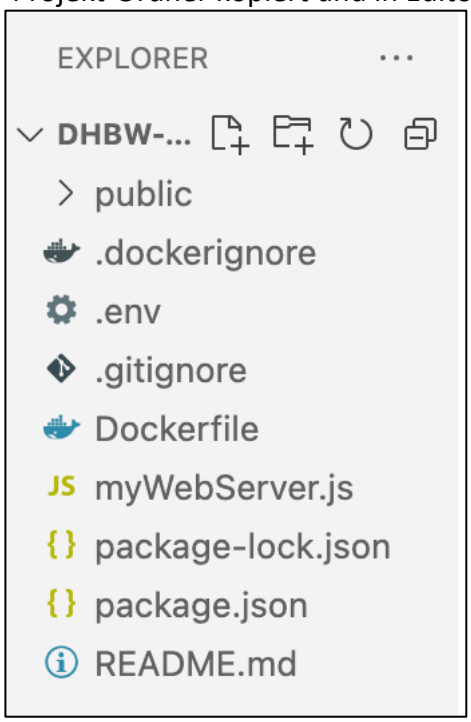
1. GitHub Repository pullen / Ordner aus E-Mail kopieren.
2. Terminal öffnen, während man sich im Ordner befindet.
3. Sicherstellen, dass NPM installiert und ausführbar ist.
4. Alle nötigen Packages installieren

a. Im Terminal: „npm install“

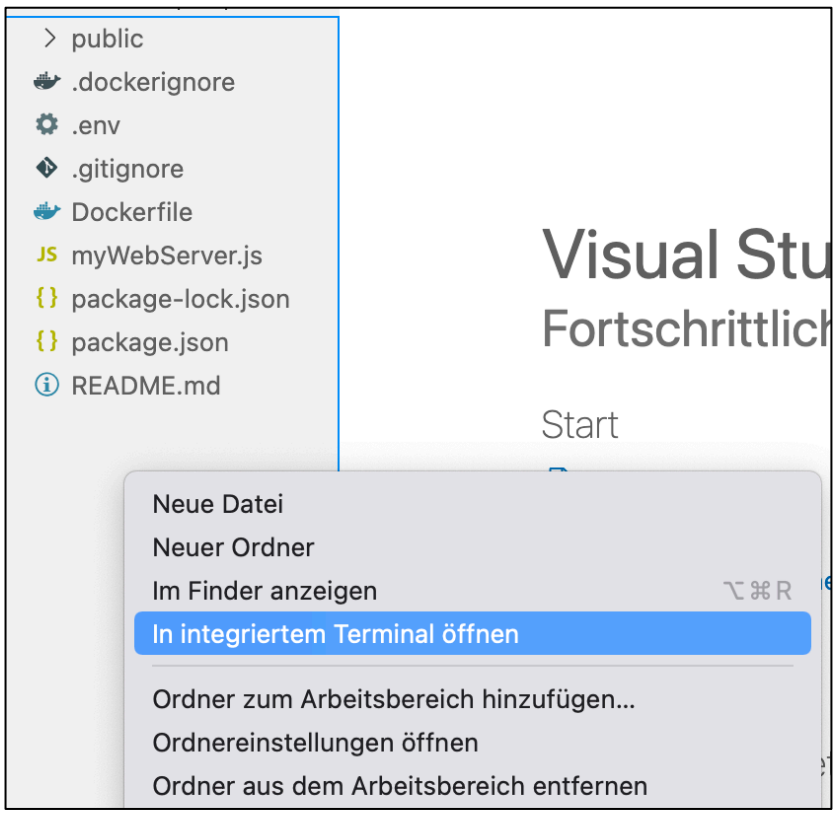
5. Kompilieren für Entwicklung
    a. Im Terminal: „npm serve“
6. Kompilieren für Produktion
    a. Im Terminal: „npm build“

Screenshots zur Installation mit Docker

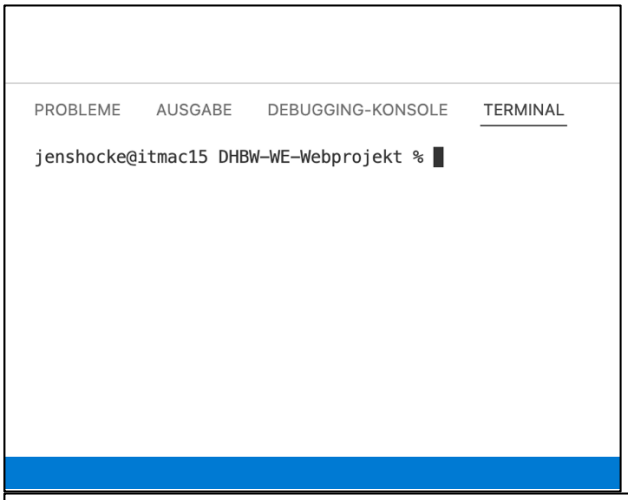
1. Projekt-Ordner kopiert und in Editor (z.B. Visual Studio Code) öffnen

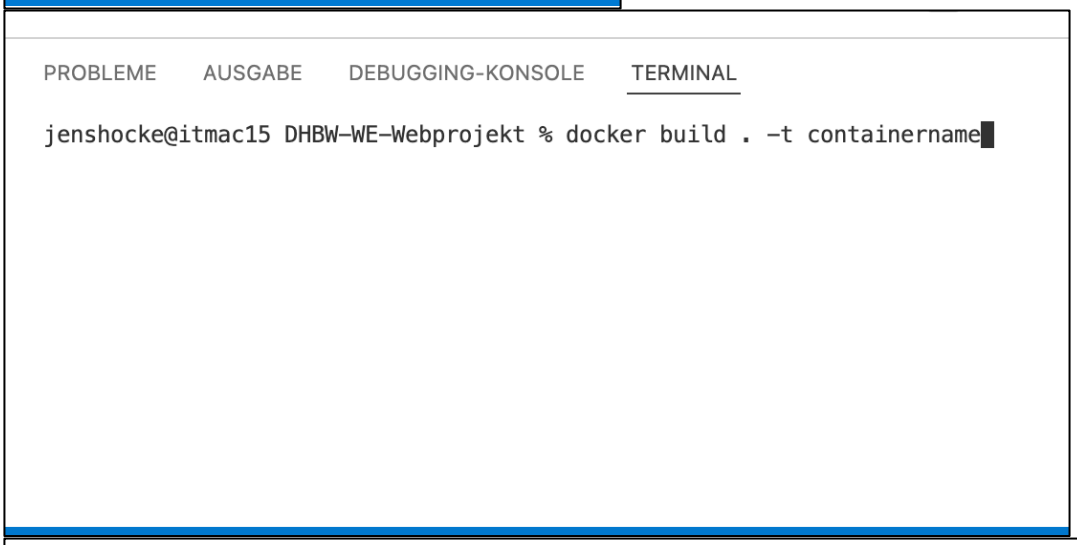


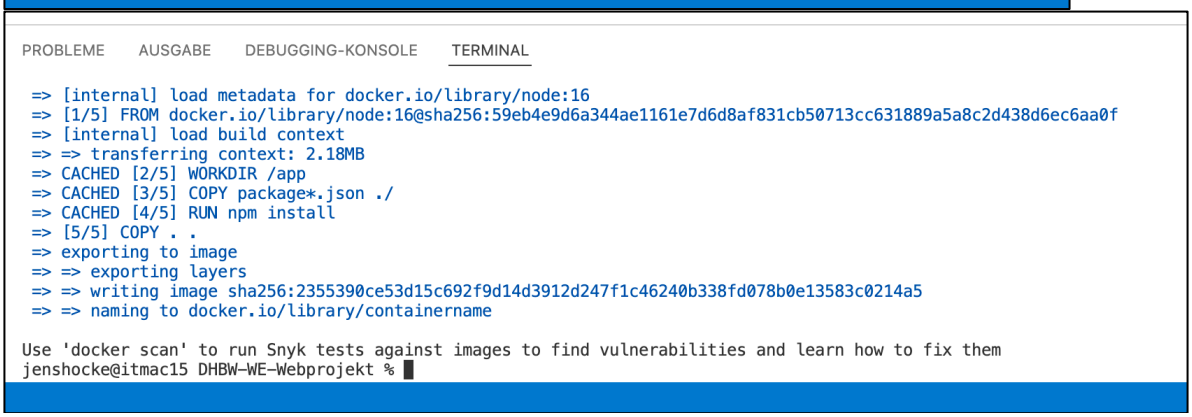
2. Termin im Ordner öffnen



3. Befehl in Termin eintippen „docker build . -t containername“







4. Befehl in Terminal eintippen: „docker run -p 8080:6001 containername“

```
PROBLEME  AUSGABE  DEBUGGING-KONSOLE  TERMINAL

 => [internal] load metadata for docker.io/library/node:16
 => [1/5] FROM docker.io/library/node:16@sha256:59eb4e9d6a344ae1161e7d6d8af831cb50713cc631889a5a8c2d438d6ec6aa0f
 => [internal] load build context
 => => transferring context: 2.18MB
 => CACHED [2/5] WORKDIR /app
 => CACHED [3/5] COPY package*.json ./
 => CACHED [4/5] RUN npm install
 => [5/5] COPY . .
 => exporting to image
 => => exporting layers
 => => writing image sha256:2355390ce53d15c692f9d14d3912d247f1c46240b338fd078b0e13583c0214a5
 => => naming to docker.io/library/containername

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
jenshocke@itmac15 DHBW-WE-Webprojekt % docker run -p 8080:6001 containername
```

5. Server startet

```
PROBLEME  AUSGABE  DEBUGGING-KONSOLE  TERMINAL

 => exporting to image
 => => exporting layers
 => => writing image sha256:2355390ce53d15c692f9d14d3912d247f1c46240b338fd078b0e13583c0214a5
 => => naming to docker.io/library/containername

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
jenshocke@itmac15 DHBW-WE-Webprojekt % docker run -p 8080:6001 containername

> 4793868Portal@1.0.0 start
> node myWebServer.js

************************************
listening: 6001
************************************
```

6. Im Webbrowser auf http://localhost:8080/ gehen