DHBW
Duale Hochschule
Baden-Württemberg
Mosbach

Assignment 1 – Typescript-Rollup

Im Studiengang Onlinemedien

(Jahrgang 2020)

an der Dualen Hochschule Baden-Württemberg Mosbach

in Kurs T2 – Frontend-Entwicklung

bei Herr Josef Slezak

von

Jonas Bloching

Ulm, den 06. Mai 2021

Inhaltsverzeichnis

1 Dokumentation der Funktionalität

Von der ersten simplen Idee, Montagsmaler, bis hin zur komplexen letzten, Tetris, bestand der erste Schritt, bevor das Entwickeln beginne konnte, darin eine meinen Kenntnissen entsprechende Idee zu finden. Die Entscheidung fiel auf eine abgeänderte Art von Tetris, bei der immer 4 Bausteine zur Verfügung gestellt werden und dann auf einem 10x10 Kästchen Feld so platziert werden müssen, dass die Reihe komplett gefüllt ist. Ist eine Reihe voll wird diese komplett geleert und es entsteht wieder Platz für neue Bausteine.

Der erste Schritt war klar: das Spielfeld erstellen. Ich habe schnell festgestellt, dass das Erstellen von 100 Kästchen mit jeweils eigener ID doch von Hand sehr umfangreich und mühselig ist. Jedes Feld hätte im Anschluss mit einer boolean Funktion versehen werden müssen, um zu überprüfen ob sich bereits ein Baustein auf dem Feld befindet oder nicht. Darauf hätte dann die restliche Logik aufgebaut. Mir war klar, dass es eine Möglichkeit geben musste, das Spielfeld mit Hilfe einer Funktion erstellen zulassen. Hier war das erste und letzte Problem der Idee. Nach mehreren Stunden Recherche habe ich es leider nicht geschafft, passende Hilfestellungen/Ideen zu finden (mir ist nicht ganz klar, woran das lag, da es ja so einige Spiele mit solch einer Funktion gibt).

Aus dem Frust und der Verzweiflung des Moments heraus warf ich die Idee hin und suchte stattdessen doch wieder nach einem Canvas, um die erste Idee zu verfolgen und endlich etwas zu sehen. Hierbei bin ich schneller erfolgreich gewesen, passenden Code zu finden. Das Adaptieren des kopierten Codes war nicht so leicht, ohne Fehler, zu kopieren. Das erste Code Beispiele wies Fehler auf, die ich nicht beheben konnte. Den zweiten Code von Chris Mytthon[1] konnte ich endlich ohne Fehler einfügen, allerdings funktionierte der Canvas noch nicht (erster Commit).

Nachdem ich für eine ordentliche HTML und CSS-Struktur gesorgt hatte, gab ich mich dem Problem hin und stellte fest, dass es wohl ein Fehler bei der Kompilation der Typescript Datei in die bunde.js geben muss. Aus dem Verdacht, beim Kopieren des Canvas versehentlich etwas gelöscht zu haben, erstellte ich eine neue Datei und kopierte den Code Stück für Stück heraus um im Anschluss den Fehler durch

[1] https://github.com/chrismytton/art/blob/1f34e99d121f2c7dd698eb8c9d45da1e9b7f450c/src/random.ts

einfaches Googlen behoben zu bekommen. Es scheint ein häufiges Problem bei erstmaligem Installieren eines globalen npm packages zu sein[2].
Das Problem, dass der Canvas nicht funktionierte, war leider dadurch noch immer nicht behoben und ich machte mich erneut auf die Suche nach Code. Nach gut einer Stunde war dieser dann funktionsfähig[3]. Der Code bietet außerdem die Eigenschaft „touch“, auf entsprechenden Endgeräten, zum Zeichnen zu verwenden, was bei einem Canvas ja nicht von Nachteil ist.
Da nun der Canvas funktionsfähig war und auch der Button zum Löschen des Canvas funktionierte, machte ich mich an zusätzliche Funktionen: ein Random Tiernamen Generator, ein Eingabefeld für den Namen der Person, die zeichnet und eine Funktion, die das Drucken der Seite vereinfacht. Die Druckfunktion war schnell integriert, bei dem Inputfeld gab es hingegen einige Probleme. Das Inputfeld in HTML erstellen und in CSS zu stylen war natürlich kein Problem, doch ich hatte lange Schwierigkeiten die Eingabe dann, in Typescript, in eine Variable zu importieren und dann wieder auszugeben. Es ist sehr frustrierend solch einfache Funktionen nicht gemeistert zu bekommen, da mir ja an sich klar ist wie die Idee umgesetzten ist. Auf der Suche nach der Lösung des Problems schaute ich mir eine Reihe von Videos generell zu Typescript und besonders dem DOM und Eingabe Elementen an, da mir dies bislang offensichtlich noch nicht ganz verständlich war[4].
Nach einer längeren Pause und mit dem Fokus auf das Stencil Gruppenprojekt beschloss ich neben dem „Löschen“ Button noch weitere Buttons für das Ändern der Farbe und Breite zu erstellen. Die Umsetzung der Funktion der Buttons musste ich allerdings vorerst hintenanstellen, da ich das Problem mit dem Inputfeld noch nicht lösen konnte. Den Ursprung des Problems vermutete ich zuerst in der Art der Eingabe, über „Enter“. Also beschloss ich noch einen Button zum Bestätigen zu erstellen, welchen ich dann mit Hilfe eines eventListeners in Typescript verwenden könnte, so die Theorie. Leider half auch dies nicht bei dem Problem. Durch die Hinzufügung des Buttons in Kombination mit dem eventListener wurde mir nun anstelle von „nichts passiert“ immer die index.ts Datei heruntergeladen. Während weiteren Lösungsversuchen, und im Gespräch mit Julia Morawietz, stellte ich fest, dass beim direkten Aufruf der HTML Datei garnichts auf der Seite funktioniert, auch nicht der Canvas. Zurück im Live Server, versuchte ich mit Hilfe des Debuggers in den DevTools

[2] https://letscodepare.com/blog/npm-resolving-eacces-permissions-denied
[3] https://kernhanda.github.io/tutorial-typescript-canvas-drawing/
[4] https://www.youtube.com/playlist?list=PL4cUxeGkcC9gUgr39Q_yD6v-bSyMwKPUI

den Ursprung des Downloads zu finden. Das Debugging ergab, dass der Download immer an der Stelle gestartet wird, an der die .innerHTML des p-Tags mit der des Inputfelds gleichgesetzt (und somit überschrieben) werden soll. Einer weitere Vermutung lag darin, dass ich beim Kopieren des Canvas ebenfalls Anpassungen in der tsconfig.json Datei vorgenommen hatte. Mich hat außerdem gewundert, dass im Code an sich keine Fehler angezeigt wurden. All diese Erkenntnisse führten für mich allerdings zu keiner logischen Schlussfolgerung, was genau den Fehler auslöste, also wandte ich mich schlussendlich in einer E-Mail an Dich. Während ich auf eine Antwort wartete, versuchte ich weiter auf eigene Faust den Ursprung des Problems ausfindig zu machen, sowie die nächste Funktion, Random Tiernamen durch Drücken eines Buttons anzuzeigen, zu integrieren. Auch hierbei war ich leider nicht erfolgreicher und auch alle Bemühungen durch Laura Schramm mir bei der Lösung meines Problems zu helfen waren leider ohne Erfolg.

Die doch sehr einfache Lösung folgte wenige Tage später per E-Mail: Ich hatte unnötigerweise ein Formular für das Inputfeld und den Button angelegt, welches direkt auf die Typescript Datei verweist. Da diese natürlich unkompiliert nicht gelesen werden kann geht der Browser davon aus, dass die Datei zum Downloaden gedacht ist und lädt folglich dann die index.ts herunter. Die Einbindung der kompilierten Datei der index.ts, also die bundle.js, im header-Bereich reicht aus, um die Funktion in Typescript auszuführen. Zusätzlich wurde durch das Hinzufügen des Attributs „defer“ noch spezifiziert, dass das Script erst nach vollständigem Parsing der HTML Datei ausgeführt wird.

Die beiden Funktionen, des Inputfelds und dem Tiernamen Generator, funktionierten leider trotz alledem noch nicht. Dieses Mal wandte ich mich an Bassit Agbéré, welcher mir nicht nur mit dem Problem an sich sehr weitergeholfen hatte. Er zeigte mir außerdem wie ich meine Variablen auslagen kann, wies mich auf semantisch passende Variablenbezeichnungen hin, zeigte mir wie ich CSS auch in Typescript schreiben kann, wie genau arrow-Funktionen eingesetzt werden und funktionieren, etc.. Fortan wurde das Recherchieren zunehmend effizienter und auch die Entwicklung zielführender. Ich strukturierte die HTML um, fügte alt-Attribute hinzu und passe das CSS an. Ich entschied mich außerdem dafür anstelle der Buttons zur Änderung der Farbe und Breite des Canvaspinsels einen Dark Mode, sowie einen Local Storage zur Speicherung des Zeichenfortschritts, zu entwickeln.

Die Umstrukturierung der HTML dachte ich einfach neben der Entwicklung des Dark Modes zu vollziehen. Diese Idee stellte sich als sehr schlecht heraus. Ich war mir sehr sicher, wie ich in der Umsetzung vorgehen musste allerdings habe ich durch verschiedene Änderungen mein Programm komplett. Dies war der Moment als ich mir der Wichtigkeit von Commit bewusst geworden bin. Ich musst in allen Dateien (mit .html, .css und .ts) mit Hilfe von cmd + „z“ wieder auf den richtigen Ursprungszustand zurückfinden. Glücklicherweise war ich nach einer guten halben Stunde erfolgreich. Im Anschluss führte ich also die einzelnen Entwicklungsschritte nacheinander durch.
Die Entwicklung des Dark Modes funktionierte, nach kurzer Recherche ohne große Probleme mithilfe der toggle Methode. Diese aktiviert und deaktiviert bei der Betätigung des Buttons das entsprechende CSS-Element, welches dann auf den HTML body angewandt wird bzw. wieder entfernt wird. Der Dark Mode Button wir außerdem erst nach der Bestätigung des Namens sichtbar. Ich habe mich dafür entscheiden, um die User Journey sinnvoll zu gestalten: Man erhält durch Klicken des Tiernamen Buttons ein Tier, das gezeichnet werden kann, bestätigt anschließend seinen Namen und kann dann mit dem Zeichen beginnen (im Dark Mode oder eben nicht). Ist man fertig mit der Zeichnung, kann die Seite über den a-Tag am Ende einfach gedruckt werden.
Trotz der Erfahrungen aus den beiden Gruppenprojekten hatte ich meine Schwierigkeiten bei der Integration des Local Storage. Nach längerem Googeln und der eigenständigen Entwicklung der saveDrawingApp() und loadDrawingApp() Methoden entschloss ich mich dazu den Code von Dominik Thurau aus dem Stencil Gruppenprojekt zur Hilfe zuziehen. Anhand diesem war dann auch die richtige Platzierung der Methodenaufrufe erfolgreich. Die Methode saveDrawingApp() speichert den aktuellen Stand des, als Bild, nach dem Beenden des releaseEventHandlers, also dem mouseup bzw touchend, sowie dem Betätigen des „Löschen“ Buttons, in den Local Storage der DataURL. Die loadDrawingApp() Methode ruft bei der Erstellung des Canvas (createUserEvents()), nach neu laden der Seite, das Bild aus dem Local Storage wieder ab und bringt diesen in den CanvasRenderingContext2d.
Die Entwicklung wurde an dieser Stelle, nach finalen Anpassungen der Variablen Semantik und CSS-Struktur, beendet. Als nächste Entwicklungsschritte wären weitere Funktionen, wie dem Ändern der Farbe oder der Dicke des Pinsels, einer Radiergummi Funktion sowie der Integration beispielsweise der paper.js library (also weiteren

Funktionen für den Canvas), denkbar und sinnvoll. Auch wäre die Erweiterung des Local Storage für z.B. das Tier und den Namen, welche in der Bearbeitung sind, noch möglich. Diese konnten allerdings aus Komplexitätsgründen zeitlich nicht mehr umgesetzt werden.

2 Persönliche Reflexion

Ich hatte große Schwierigkeiten mit dem Assignment, da ich keinerlei JavaScript Vorkenntnisse (ganz abgesehen von Typescript, Stencil und Co) vorweisen kann. Ich hatte bislang weder privat, in meiner ersten Praxisphase noch in Kursen des ersten Semesters das Glück, mit der Thematik in Kontakt zu kommen.
Die Vorlesungen an sich waren gut strukturiert, dennoch war es für mich, und meine meisten KommilitonInnen das erste Mal, sich mit den Themen auseinander zu setzen. In fünf Vorlesungen wurden die Skriptsprachen JavaScript und Typescript aber auch das Konzept von Umgebungen, wie NodeJS, und bis hin zum Web Component Compiler Stencil eingeführt und vertieft. Ich persönlich hatte meine Schwierigkeiten damit die einzelnen Lerninhalte direkt zu verstehen und dann auch im Anschluss direkt allein anzuwenden. Daher begannen die Assignments für mich erst einmal mit der Nacharbeitung und einer genaueren Einarbeitung in die einzelnen Themen an sich. Dieser Prozess war bereits sehr zeitaufwendig und sorgte für Schwierigkeiten in der Themenfindung (die ja dann auch zeitnah erfolgen musste). Trotz beispielhafter App-Ideen in Moodle und github war die Entwicklung einer Idee für mich sehr mühselig, da ich ohne Vorkenntnisse die Komplexität meiner Ideen nicht sehr gut einschätzen konnte. Ich habe lange mit dem Gedanken gespielt eine TutorInnenstunde zu buchen um direkt, während ich arbeite, Rückmeldung zu bekommen sowie zusätzliche Kniffe, die man durch Googlen vielleicht nicht herausfinden würde, zu lernen. Besonders nach der Hilfestunde durch Bassit hatte sich diese Hypothese bestätigt. Es war überaus effizient, lehrreich und zielführend (Crashkurs). Im Anschluss fühlte ich mich bereits sehr viel besser in das Thema eingearbeitet.
Ein großes Problem, wenn nicht sogar das größte Problem, stellt das langwierige und oftmals erfolglose scheinende Recherchieren dar. Durch die niedrigen Grundkenntnisse fiel mir das schnelle und zielführende Googlen sehr schwer, was sehr frustrierend war. Mehrmals kam ich selbst nach einer Stunde googlen nicht zu dem gewünschten Ziel. Als Ausweg blieb nur das Fragen von KommilitonInnen bzw. das Weglassen der Funktion oder eine E-Mail zu schreiben. Der Entwicklungsprozess war dadurch sehr mühselig, äußerst Zeit intensiv und nicht wirklich zielführend.
Durch Tabea Wöhr und eine YouTuberin bin ich außerdem auf zwei sehr nützlichen Erweiterungen in VS CodeBracket Pair Colorizer 2 und TabNine aufmerksam geworden. TabeNine arbeitet mit einer künstlichen Intelligenz, die den Schreibstyle

erkennt und dir dementsprechend passende Ergänzungsvorschläge bietet, die über die von VS Code selbst hinaus gehen.
Mit dem Ergebnis meines Einzelprojekts bin ich schlussendlich sehr zufrieden. Trotz der vielen Schwierigkeiten im Prozess kann sich das Ergebnis durchaus sehen lassen da die Applikation soweit funktionsfähig ist.
Alles in allem habe ich dieses Semester vor allem HTML und CSS durch die drei Projekte stark verinnerlicht (Einzelprojekt, Stencil Projekt und besonders dem Projektmanagement Projekt) und erste Erfahrungen und Grundlagen in Javascript etc. gelernt. Ich hoffe im Laufe meines Studiums noch weiter in dieses Feld einzutauchen, da es nach dem Verinnerlichen der Grundlagen bekanntermaßen auch zunehmend Spaß macht. Allgemein habe ich jedenfalls Interesse mein Wissen und Können in der Frontend Entwicklung weiter voran zu treiben.

3 Quellen

- [1] 13.04.2021: https://github.com/chrismytton/art/blob/1f34e99d121f2c7dd698eb8c9d45da1e9b7f450c/src/random.ts
- [2] 20.04.2021: https://letscodepare.com/blog/npm-resolving-eacces-permissions-denied
- [3] 21.04.2021: https://kernhanda.github.io/tutorial-typescript-canvas-drawing/
- [4] 24.04.2021: https://www.youtube.com/playlist?list=PL4cUxeGkcC9gUgr39Q_yD6v-bSyMwKPUI

Input: Josef Slezak zu paper.js library:

- http://paperjs.org/tutorials/interaction/working-with-mouse-vectors/
- https://github.com/paperjs/paper.js/blob/develop/dist/paper.d.ts

Allgemeine Recherche:

- https://www.w3schools.com
- https://developer.mozilla.org/de/

Bilddate (GIF):

- https://lottiefiles.com/53164-light-dark-mode-button?lang=de#