Hausarbeit:

Die Aufgabe der Hausarbeit ist die Implementierung eines „Übungs“-Navigators. Ursprünglich habe ich das Projekt in HTML, CSS und JavaScript geschrieben. Nachdem ich aber die Übung zu VueJs bearbeitet habe, habe ich mich dazu entschlossen das Projekt noch mal neu Aufzuziehen und eine Web-App mit Single-File-Components zu erstellen.

Bei der HTML Implementierung habe ich die sehr viel dem DOM (Document Object Model) und Event-Listenern gearbeitet. Dies hat es mir ermöglicht, den Inhalt der Seite neu zu rendern, ohne dass die Seite neu geladen werden muss. Dies hat auch super geklappt nur war das ganze Verfahren sehr kleinkariert und Fehler Anfällig.

Die Implementierung von den Single-File-Components dagegen war sehr angenehm. Ich musste Elemente, die ich in der Web-App mehrmals benutze, nur einmal implementieren und konnte daraufhin die Components beliebig oft benutzen. Ein weiterer Vorteil dieses Verfahrens ist es, dass die Funktionalität (JavaScript), das Design (CSS) und der Aufbau (HTML) immer in den einzelnen Components vorgenommen werden konnte und sich die einzelnen Components so nie in die Quere gekommen sind. Änderungen an der Funktion, am Aussehen oder an dem Aufbau konnten sehr leicht geändert werden, da ich nicht an verschiedenen Stellen auf die Änderungen achten musste, sondern nur im dem jeweiligen Component. Nachdem ich das Grundgerüst der Web-App erstellt hatte, habe ich mich an den Text-Inhalt gemacht. Ich wollte dies so weit wie möglich von der eigentlichen App trennen, sodass der dynamische Aufbau auch hier wiedergefunden werden kann.

Die Umsetzung erfolgte durch eine .json Datei. Dies hat es mir ermöglicht, das „Hard-Coden“ in der Web-App zu vermeiden. Jedes der angelegten Components weiß, wo der für ihn relevante Inhalt liegt, und die Kommunikationen erfolgt ausschließlich über Attribute. Da wir für die Übungen auch teilweise ganze „html‘s“ erstellen sollten, lag mein Augenmerk ursprünglich darauf, dass das html aus der .json Datei richtig gerendert wird.

Da das Erstellen der .json Datei nur Copy-Pasten ist (oder zumindest habe ich es gedacht), habe ich das zunächst bei Seite geschoben um mich auf ein schönes Design konzentrieren zu können. Dabei habe ich versucht das Design möglichst Klassisch – Minimalistisch zu halten. Im Nachhinein betrachtet, könnte es auch langweilig wirken aber mir gefällt das Design sehr gut. Bis auf wenige Animationen (Qullen stehen als Kommentar im Code) habe ich das Design vollkommen selbst umgesetzt. Ich bin ziemlich Stolz darauf, da ich vor dieser Veranstaltung noch nie mit HTML, CSS oder JavaScript in Berührung gekommen bin. Jemand der ein bisschen mehr Erfahrung in dem Bereich hat wird dies Wahrscheinlich eher Belächeln.

Als ich dann die .json Datei vervollständigen wollte, ist mir beim Kapitel „JavaScript“ aufgefallen, dass das <script>-Tag beim Rendering vollkommen ignoriert wird. Es erscheint zwar in der Ausgabe, jedoch wird keine Funktionalität unterschützt und die Konsole gibt einen Fehler aus, der besagt, dass die Funktion „beispiel()“ nicht gefunden wurde. Da ich das erstellen der .json Datei als eine Fließband-Arbeit (Copy-Paste) betrachtet habe, habe ich dies auch bis zum Ende Aufgeschoben. So war ich bei der Entdeckung des Problems schon kurz vor der Abgabefrist. Ich habe dann noch sehr viel Zeit damit verbracht das ganze doch noch hinzukriegen – leider ohne Erfolg.

Da es aber auch in der Praxis eher seltener vorkommt, dass man ganze html Dateien mit den Scripts in ein Component einfügen will, habe ich es dann auch sein lassen. Der ganze Sinn der Single-File-Components ist es ja, html-Elemente mit allem was dazu gehört in einem einzigen Component zu speichern und nicht eine ganze Seite zu Rendern, die ihren eigenen Script mit sich bringt.

Im großem und ganzen bin ich wirklich sehr stolz auf das gesamte Projekt. Besonders Stolz bin ich aber auf die Tatsache, dass das Projekt so dynamisch gestaltet ist. Wie schon erwähnt, kommt der Text-Inhalt aus einer .json Datei, der Style und die Funktionalität ist in den einzelnen Components implementiert und die App.vue fügt dann alles zusammen. So könnte ich mit dem Austausch einer einzelnen Datei (.json) dafür sorgen, dass der gesamte Inhalt der Web-App ausgetauscht wird, ohne die irgendwas in der App selber zu verändern. Um eine Änderung im Design wie z.B. die Hintergrundfarbe, Schriftgröße, Schriftfarbe, usw. zu erzielen muss man nur in der .css Datei etwa umschreiben um dies für die ganze App umzusetzten.

Da ich nun auch so ziemlich am Ende des Studiums bin und ich mir Gedanken über den anstehenden Bewerbungsprozess für das PraxisProjekt oder auch danach für einen Job mache, war meine Idee eine WebSite zu bauen, die mit in meine Bewerbung kommt. Mit dieser Web-App habe ich meiner Meinung nach ein sehr gutes Gerüst geschaffen welches ich in Zukunft mit Sicherheit noch ausbauen werde.