# Dokumentation

## Web-Komponente 1 – Footer

Zu Beginn habe ich zuerst die Komponente mit „@Component“ definiert. Da ich hier in dieser Komponente den Footer darstellen wollte, benannte ich die Komponente „my-footer“. Die styleUrl habe ich ebenfalls „my-footer“ genannt. Ich habe den shadow auf true gesetzt, da ich hier in dieser Komponente ein abgegrenztes Styling unabhängig vom Styling der HTML Seite haben wollte.

Die zu exportierende Klasse heißt MyFooter. In ihr wird der Footer gerendert und gibt das Div mit dem Impressum als Schrift und Anker zurück, welches dann den Inhalt der Komponente bilden.

Hier kam ich zu einem großen Problem, welches mich im fixen etwas Zeit gekostet hat. Ich habe das Rendering mit einem div gebaut, welches ein <a>-Tag und ein <p>-Tag beinhaltet. Das <p>-Tag besitzt den Inhalt „Impressum“ und soll durch das <a>-Tag zum Impressum leiten. Jedoch kam man, wenn man die Seite per Klick aufrufen wollte nur auf eine Fehlerseite und die eigentliche Seite des Impressums wurde nicht gefunden.

Daher wurde zuerst eine leere Variable initialisiert. Diese Variable verändert sich, da folgendes Problem besteht. Wenn der User der Seite auf der Startseite ist, benötigt er einen anderen Link, wie wenn sich der User auf einer Unterseite befindet. Wenn der User auf der Startseite ist, dann kommt der User zum Impressum, da nur einmal HTML vorangestellt wird. Wenn der User sich allerdings auf einer Unterseite befindet wird zusätzlich ein HTML vorangestellt und der User gelangt nicht mehr auf das Impressum. Daher benötige ich die if-Bedingung. Ich frage hier ab, ob sich der User auf der Startseite befindet mit dem „window.location.href=="html"“. Dann soll der Link leer sein + der Link zum Impressum wie im Code Zeile 18 veranschaulicht ist. Wenn der User allerdings auf einer Unterseite ist, soll die Variable link auf die Startseite zurückdeuten und dann erst das Impressum aufrufen. Dadurch umgeht man das Problem, dass dem Link zweimal HTML vorangestellt wird.

Daraufhin habe ich den Footer gestylt per CSS. Ich habe den Footer fixed am unteren Bildschirmrand angebracht. Wir haben uns in der Gruppe zuvor darauf geeinigt die Farben Schwarz, Weiß und Rot zu verwenden. Daher ist die Farbwahl des Footers auf schwarz gefallen und die Schrift dazu im Kontrast weiß. Ich habe die Schrift auf die vertikale Höhe platziert und auf die rechte Seite, sodass das Wort zirka rechts auf einer Höhe mit dem äußersten Wort der Nav-Bar ist.

Der Grund für diese Komponente ist, dass man von jeder Seite auf das Impressum zugreifen kann.

## Web-Komponente 2 – Flexdiv

Bei der zweiten Webkomponente habe ich ebenfalls zu Beginn wieder die Webkomponente mit @Component definiert. Der Tag und die styleUrl wurde auf das flexdiv angepasst und ebenfalls aus den gleichen Gründen wie bei der 1. Webkomponente wurde wieder der shadow auf true gesetzt.

Dieses Mal ist die Klasse etwas kleiner als bei der 1. Webkomponente. Die Webkomponente besitzt ebenfalls wieder eine zu exportierende Klasse, welche dieses Mal den Namen MyFlexdiv besitzt. In ihr wird das zu rendernde Element beschrieben. Das Element besteht aus einem Div, welches die Inhalte der Unterseiten strukturieren soll in einer Flexbox. Ich habe hier im <div>-Tag ein <slot> eingebaut. Dieser Slot sorgt dafür, dass der Inhalt selbst eingesetzt werden kann.

Entscheidend für die Webkomponente ist allerdings das CSS der Komponente. Ich habe hier ein „display: flex;“ eingebaut. Die Richtung sollte in einer Reihe sein, da die Inhalte in Divs nebeneinander angeordnet werden sollen und immer am Beginn der Flexbox angeordnet sein sollen.

## Zusätzliches nach der Erstellung

Wir haben uns als Gruppe darauf geeinigt die Überarbeitung und die Strukturierung der Seiten (sofern benötigt) aufzuteilen. Ich für meinen Teil arbeitete dann mit der Startseite und der Unterseite Geschlossenheit.

Angefangen mit der Startseite habe den Inhalt der Startseite nochmal in ein Div mit der Klasse „.contentAlign“ gepackt. Diese Klasse platzierte den Inhalt der Seite mittig per Flexbox-Struktur und platzierte den Content der Seite in die Mitte des Bildschirms. Ebenfalls hatte die <my-headline> ein Margin, welches ich um einen Minus-Wert verändert habe, sodass der gesamte Content in der Mitte platziert ist.

Als nächstes kümmerte ich mich um die Anordnung auf der Unterseite Geschlossenheit. Ich habe hier auch eine Beispielstruktur für die Verwendung der Elemente in der richtigen Reihenfolge erarbeitet und versucht das Ganze im HTML-Code anschaulich darzustellen. Die bereits erstellten Tags aller Kommilitonen aus der Gruppe habe ich in der richtigen Reihenfolge eingebaut. Diese waren bereits in einem Template-Versuch vorhanden, welchen wir verwarfen, da es für die Aufgabe nicht relevant war verworfen haben. Die Struktur bezüglich der Flexbox hat allerdings nicht funktioniert. Die Inhalte wurden stupide untereinander angezeigt. Also habe ich manuell die Divs erstellt, worin die Inhalte eingepflegt werden müssen um von der Flexbox organisiert zu werden. Das funktionierte soweit auch, jedoch war kein Abstand zwischen den Elementen vorhanden. Ich habe also den Divs Abstände gegeben und ebenfalls eine feste Breite, da die Breite von Div zu Div ansonsten unterschiedlich war.

Den Inhalt für die Startseite habe ich mir selbst ausgedacht. Den Inhalt für das Gesetz der Geschlossenheit habe ich von folgenden Websites kopiert:

-https://www.klickkomplizen.de/blog/print-design/gesetz-der-geschlossenheit/

-https://www.toushenne.de/design/gestaltgesetze-der-wahrnehmung.html

-https://www.webmasterpro.de/design/article/gesetz-der-geschlossenheit.html

# Reflexion

Vor der Vorlesung hatte ich noch nie etwas von dem Framework Stencil gehört. Für mich war es in der Vorlesung sehr schwer den Überblick zu behalten über die Bereiche JavaScript, TypeScript und Stencil. Daher habe ich zu Beginn unseres internen Assignment-Starts erneut die Vorlesung angeschaut und das Vorlesungsskript zu Stencil durchgelesen. Für mich war vor allem die Schwierigkeit zu verstehen wie die Komponenten zusammenhängen, warum eine Komponente so erstellt wird, wie sie erstellt wird und wieso man überhaupt solche Webkomponenten benötigt.

Als ich mich dann selbst rangemacht hatte an die Erstellung meiner Komponenten habe ich zuerst etwas Zeit gebraucht um zu verstehen wie ich eine Komponente überhaupt funktionstüchtig bekomme. Daher habe ich auch für die Erstellung der ersten Komponente viel mehr Zeit benötigt als für die Erstellung der 2. Komponente.

Besonders gut gelaufen für meinen Teil ist vor allem das Erarbeiten der Flexbox. Der Inhalt hat sich mit ein paar Nachbesserungen super organisieren lassen.

Als Recap zu meinem Wissenstand vor dem Assignment kann ich von mir behaupten, sehr viel gelernt zu haben. Ich kann jetzt eigene Komponente anlegen und verstehen den Sinn und Verwendungszweck von Stencil.

Für die Zukunft und zukünftige Projekte würde ich mich an die Erstellung eines Templates wagen und versuchen das Template funktionstüchtig zu bekommen. Sozusagen als Steigerung zur Erstellung von Webkomponenten. Ich habe mit unserem Dozenten eine Stunde versucht unser bisher fehlerhaftes Template zu fixen, jedoch rannte die Zeit weg und deshalb beschlossen wir ohne das Template an sich zu arbeiten. Ich bedanke mich also in dieser Ausarbeitung nochmal für die Hilfe bei unserer Arbeit, da Probleme und offene Fragen gut geklärt werden konnten.