# Dokumentation

Ziel dieser Web-App ist es, durch die Gestaltgesetze zu navigieren. Zu jedem wird mittels Beispiele und Begleittext das Gesetz erklärt.

## My-button.tsx

Der Tag wird dem Shadow-DOM mittels „shadow = true“ zugewiesen. Sinn des Buttons ist es, von der aktuellen auf die nächste Seite zukommen. Dadurch soll ein Rundlauf durch die Gestaltgesetze ermöglicht werden. Dazu ermittle ich zunächst die URL. Das passiert jeweils zu Beginn der Funktionen. In der Funktion „setButtonTagID()“, die beim rendern aufgerufen wird, begrenze ich sie durch regex ausschließlich auf den Dokumentnamen + Dateiendung. Durch einen control-flow weise ich dann die passende Zahl der Variable „buttonID“ zu. In der Funktion „goToNextPage()“, die durch „onclick“ aufgerufen wird, überschreibe ich dann die aktuelle URL, nachdem ich je nach buttonID den passenden Pfad angegeben habe. Hier arbeite ich wieder mit einer „.replace“-Funktion und regex. Wenn der User auf der letzten Seite des Rundgangs angekommen ist, erscheint ein Hinweis und er landet wieder auf der Startseite. Den Text des Buttons setze ich, ebenfalls wie das Ziel, variable. Dazu wird in der Funktion „setContent()“ je nach URL der passende Content generiert.

## My-Headline.tsx

Dieser Component wird mittels „shadow = true“ ebenfalls dem Shadow-DOM zugewiesen. Bei my-headline ermittle ich die URL über die Funktion „getURL()“, damit ich den Inhalt variabel korrekt anpassen kann. Befindet sich der User auf einer Seite eines Gestaltgesetzes, wird zusätzlich „Gesetz“ im Content angezeigt. Im Render-Prozess werden dann alle „ae“s zu „ä“s, bei Fortsetzung wird noch „guten“ angehängt und die passenden Artikel werden vergeben. Durch mehrere Control-Flows wird dadurch jede einzelne Headline korrekt vergeben, ohne, dass der User Text einfügen muss.

## My-subheadline.tsx

Auch dieser Component wird wieder dem Shadow-DOM zugewiesen. Ich habe einen Slot-Tag eingebaut, um bei der Erstellung eines Templates trotzdem noch Text variabel hinzufügen zu können. Außerdem kann der Content so ohne Properties zwischen den Tags eingefügt werden.

# Quellen

<https://angular.io/guide/template-syntax>

(zuletzt aufgerufen am 18.04.20)

<https://upmostly.com/tutorials/react-onclick-event-handling-with-examples>

(zuletzt aufgerufen am 18.04.20)

and of course: <https://stackoverflow.com/>

(zuletzt aufgerufen am 18.04.20)

# Reflexion

Ich habe bislang noch nicht mit StencilJS gearbeitet und musste mich gerade zu Beginn erst mal durch den Aufbau kämpfen. Die Arbeit mit den Components selbst war dann gar nicht mehr so schwer. Allerdings galt es auch hier wieder, sich mit den einzelnen Eigenschaften auseinanderzusetzen. Gerade die Interpolation-Notation hat mir bei der Gestaltung der Buttons sehr geholfen. Dadurch konnte ich sowohl die ID variabel übergeben, als auch die Funktion beim Klicken ausführen. Ansonsten habe ich diese Notation eigentlich in jedem Component für den Inhalt genutzt. Wenn ich irgendwas Sinnvolles mit regex machen kann, freut es mich auch immer. Ich habe hier noch Übungsbedarf und nutze solche Gelegenheiten dann sehr gerne, um meine Fähigkeiten zu verbessern.

Mein größtes Problem galt dem Methodenaufruf innerhalb anderer Funktionen. Da ich hier keine zufriedenstellende Lösung gefunden habe, bin ich ein Kompromiss eingegangen. Anstatt unterschiedliche Schritte in Funktionen auszulagern, habe ich sie in mehrfach in anderen Funktionen aufgerufen. Das hat mich zwar nicht ganz so zufrieden gestellt, aber dafür hat es funktioniert.