

[Dokumenttitel]

[Untertitel des Dokuments]



Inhalt

[Persönliche Reflexion 2](#_Toc35875026)

[Dokumentation 3](#_Toc35875027)

[Feature 1 – Page Preloader 3](#_Toc35875028)

# Persönliche Reflexion

# Dokumentation

## Feature 1 – Seiten Ladeanimation

Ich habe vor ein paar Monaten eine Page Ladeanimation gebaut mit einem GIF in Verbindung von HTML CSS und JavaScript. Da dachte ich, es ist ein „Fancy“ Feature eine Ladeanimation für das Template zu schreiben. Als erstes ging es darum, welches Ladesymbol ich verwende. Ich hatte keine Lust alles selbst per HTML und CSS zu gestalten, zumal dies ja sowieso nicht Bewertungskriterium ist. Deshalb verwendete ich https://loading.io/ und baute mir damit mein Lade-GIF. Eingebunden habe ich das GIF per HTML <img> Tag. Zunächst ging es dann darum das GIF richtig zu positionieren per CSS und eine Hidden-Animation einzufügen. Hier habe ich für das Fade out die Annotation @keyframes verwendet, mit der man einzelne Schritte einer Animation bearbeiten kann. Schließlich habe ich mich an den JavaScript Teil heranbewegt. Ich erstelle ein Event „load“, welches eine Funktion beinhaltet. In dieser Funktion greife ich per DOM mit dem .querySelector auf die CSS-Klasse .loader zu. Jetzt habe ich sie zwar aufgerufen, aber die Animation ist dauerhaft. Dann rufe ich auf der Klasse loader hidden auf und die Animation wird visibility hidden.

Bei diesem Versuch wurde jedoch .hidden als Klasse direkt aufgerufen und ausgeführt. Ich hatte nach langer Fehlersuche keinen blassen Schimmer, wieso die Ladeanimation trotz setTimeoutFunction nicht funktioniert. Ich wendete mich an Josef (den Dozenten der Vorlesung Front-End Entwicklung). Der Code veränderte sich daraufhin nur minimal. Der Unterschied ist einfach, dass die Funktion hideLoader direkt in der Funktion setTiemoutFunction extern aufgerufen und ausgeführt wird. Ebenfallls wird der Klasse .loader im HTML die Klasse .hidden angehängt nach den zwei Sekunden.

Ich habe den Timer nur zu Testzwecken auf 2 Sekunden gesetzt, damit man sieht, dass die Funktion auch wirklich funktioniert. Auf einer Live Seite im Internet würde ich die Funktion umschreiben bezogen auf die Ladezeit der Seite und die FadeOut Animation bezüglich der Länge um die Hälfte kürzen.

## Feature 2 – Toggle Dark Mode

Ich habe mir überlegt, welches Feature heute mehr Aufmerksamkeit denn je erhält. Ganz klar kam ich auf die Thematik Dark Mode. Ich habe mir zuerst überlegt, welche Features ich umfärben möchte und wie ich das Bewerkstelligen möchte. Ich kam auf die klassische Idee per Button. Habe auch überlegt einen Schieberegler zu verwenden, jedoch aufgrund der Benotungskriterien habe ich den Button gewählt. Er ist schneller erstellt und das CSS Styling spielt keine Rolle für die Benotung. Zu färbende Features waren für mich der Hintergrund und die Schrift. Auf mehr habe ich nicht geachtet, da ich für mich erstmal wissen wollte, ob ich das Feature überhaupt selbst programmiert bekomme. Ich kam schnell auf die Idee des Arbeitens und Ersetzten mit CSS-Styling. Ich habe zuerst einen Button mit einem onclick Event in HTML erstellt und die Styling-Klasse der Buttons übernommen. Ich habe nun auf der CSS-Seite die Darkmode-Farben plus zzgl. Styling ersetzt aus den Originalklassen oder IDs, die das CSS-Styling definieren. Dann habe ich mich an den zu bewertenden Teil gemacht. Ich habe eine Funktion toggleDarkMode definiert, welche vom Button per Klick aufgerufen wird. In der Funktion wird die Variable darkMode definiert, mit der ich auf die Klasse dark-mode-button zugreife und die Schriftfarbe ändere. Ebenfalls greife ich auf die Klasse wrapper zu und ersetze diese durch meinen veränderten Style für den Dark Mode.

Jetzt stoße ich aber auf ein Problem! Bei erneutem betätigen des Knopfes wird die Hintergrundfarbe nicht geändert, sondern nur die Schrift. Ich habe versucht die Funktion umzuschreiben und per jQuery per .one die Betätigung nur einmal ausführen zu können. Ich kam nur zu weiteren Fehlern also suchte ich nach einer anderen Lösung. Es muss ein zweiter Button her für die Aktivierung des Light Mode. Damit löse ich zwar das Problem des Bugs nicht komplett, allerdings biete ich eine Möglichkeit wieder den Light Mode zu verwenden ohne die Seite neu zu laden. Man betrachte also die Verwendung des Dark Mode als vorrübergehender Betatest.

Für die Aktivierung des Light Modes wird dann wiederrum das CSS-Styling per JavaScript angesprochen per onclick. Jetzt kann man also den Light Mode abrufen, ohne die Seite neu laden zu müssen.

## Feature 3 –