Assignment 1

DHBW Mosbach, Frontend-Entwicklung, Herr Slezak, HSS 2020, Alischa Thomas

Vorkenntnisse und Vorbereitung

Meine Vorkenntnisse im Themenbereich Programmieren mit JavaScript (JS) belaufen sich auf ein theoretisches Grundverständnis von Programmieren im Allgemeinen und praktischen Grundlagenkenntnissen mit HTML und CSS.   
Deshalb arbeitete ich vor der Bearbeitung dieses Assignments nochmal den Foliensatz zu Vorlesung 1 durch. Des Weiteren beschäftigte ich mich nochmal mit den Übungen aus dem ersten Foliensatz. Darüber hinaus arbeitete ich den interaktiven Kurs J*avaScript Fundamentals for ES6* auf der Plattform *Pluralsight* durch. Das Ziel dabei war ein praktisches Verständnis der Syntax und Logik von JavaScript zu erarbeiten.  
Vor dem Einstieg ging ich nochmal systematisch die Anforderungen an das Assignment in Moodle durch. Daraus identifizierte ich für mich nochmals klar das Ziel des Assignments für mich und Den Dozierenden. Ich überlegte mir was ich im Rahmen dieses Assignments erlernen möchte und welches Vorgehen am Besten geeignet ist um diese Ziele zu erreichen.   
Aus Erfahrungen mit der Konfrontation neuer Programmieraufgaben weiß ich, dass ich mich in der Vergangenheit schnell erschlagen fühlte und daher auch vorzeitig an meine Grenzen stieß. Deshalb schrieb ich die Anforderungen und meinen Plan erstmal schriftlich nieder um somit besser den Fokus während des Arbeitsprozesses wahren zu können.

* Zielvorgabe: Manipulation eines Templates mit JS: im Vordergrund an UI Elementen, im Hintergrund, DOM-Manipulation (5 Interaktionselemente), Dokumentation inkl. Persönliche Reflexion, Abgabe versioniertes Git Repository
* Meine Ziele: Ajax, Animation, Canvas, Buttoninteraktion
* Bedingungen: ES6 Standard verwenden, kein jQuery
* Bewertung: JS-Logik

Um nicht den Überblick zu verlieren wurde das Assignment Schritt für Schritt aufgebaut. Die einzelnen Manipulationen mit JS wurden also chonologisch bearbeitet. Dies diente dazu sich intensiv mit den einzelnen Bestandteilen auseinandersetzen zu können und sich nicht gleich überfordert zu fühlen. Dabei wurde das Ziel gesetzt sich mit jedem Schritt in der Komplexität zu steigern.   
Das Git-Repository wurde mit GitHub umgesetzt, der Code mit in der Umgebung von Webstorm (IntelliJ) geschrieben.

# 1 Button Manipulation

Idee  
Das Template weist bereits 4 klassische Buttons auf. Das Ziel der Buttonmanipulation ist es das Erscheinungsbild im User Interface beim Abfeuern eines Events zu verändern.

## Theoretischer Hintergrund

Dies dient ein der Praxis z.B. einem entscheidungsfreundlichen Design oder der besseren orientierung für NutzerInnen. Eine Entscheidung kann zum Beispiel unterstütz werden, wenn der Button die Farbe ändert bei einem Mouseover. Dadurch wird die Klickbarkeit des Buttons hervorgehoben. Bei der Orientierung kann es helfen, wenn bereits geklickte Buttons farblich gekennzeichnet werden wie zum Beispiel auf der Suchergebnisliste von *Google*.   
Umgesetzt wird diese Manipulation mit den beiden Methoden *getElementbyID()* (*HTML DOM getElementById() Method*, n.d.) und *addEventListener()*. Die erste methode greift auf das HTML-Element zu. Die zweite Methode weist ihm ein Event zu, auf das reagiert wird. Dabei gibt es unterschiedlichste Events aus den Bereichen User Interface Events, Focus and Blur Events, Mouse Events, Keyboard Events, Form Events, Mutation Events and Observers, HTML5 Events, CSS Events (dfteam7, 2019). Bei dieser Button Manipulation werden Mouse Events umgesetzt. Spezifisch für Klickevents gibt es alternativ noch die *onklick-Methode*. Beide Methoden wären bei dieser Manipulation anwendbar (*javascript—AddEventListener vs onclick*, n.d.). Da die *addEventListener-Methode* allerdings flexibler und breiter aufgestellt ist, wird sich für diese im Folgenden angewendet.   
  
Praktische Umsetzung Programmieren  
Bevor mit dem Schreiben von JavaScript begonnen werden kann, müssen die HTML-Elemente der Buttons mit einer ID versehen werden. Über diese ID erlangt man Zugriff auf das Element in dem JS-Dokument mit der Methode *getElementByID()*. Die zu manipulierenden Buttons wurden über diese Methode einer variable zugewiesen. Im nächsten Schritt wird der Variable über die Methode addEventListener() das Event zugewiesen, auf das reagiert werden soll. In diesem Zuge wird außerdem die Funktion deklariert, die beim Abfeuern des Events ausgeführt wird. **(Auflisten Funktionen und Events).**  
Um den Style der Buttons manipulieren zu können wurden zuerst die Standardstylings von Buttons in CSS recherchiert (*CSS Buttons*, n.d.). Wir das Standardstyling nicht beachtet kann e zu Fehlern in der Manipulation kommen. Möchte dem Rand eine Farbe geben bei Abfeuern eines Events, muss dem Button zusätzlich zu der Farbe ein Borderstyle zugewiesen werden, da dieser im Standardstyling auf *none* steht. Somit würde man keine Veränderung durch das Event sehen können.  
Die Füllfarbe des Buttons wird nun in der Funktion der methode addEventListener() hinzugefügt. Dazu ruft man an der zuvor erstellten Buttonvariable die Stylemethoden auf (*AddEventListener and changing background color*, 2018). Die Farbe wurde passend zur Website mit dem *Adobe Color Tool* ausgewählt (*Farbpalette, das Farbschema für Künstler | Adobe Color*, n.d.).

Beim Anklicken oder Hovern eines oder über einen Button wechselt dieser nun seine Füllfarbe.

## Probleme und Hindernisse

Bei dieser ersten Manipulation bezogen sich die größten Probleme auf den Einstieg. Es fiel mir schwer anzufangen weil ich nicht genau wusste wie. Beim Arbeiten mit HTML wird einem zum Beispiel bereits eine grobe Struktur vorgestellt, das ist hier nicht der Fall. Was mir hier weitergeholfen hat war die Sichtung des Codes anderer auf *Stackoverflow*. Dadurch fing ich dann einfach an zu programmieren und hatte schon schnell meine ersten Code-snippets erstellt.   
Zu Beginn bereitete mir die Arbeit mit dem integrierten Version Control Tool von Webstorm starke Probleme und kostete mich viel Zeit. Durch intensive Recherche und mehrere erstellt und doch wieder gelöschte Repositories, sowie einem Austausch mit meinem Dozierenden entschied ich mich dazu Github über Terminal und die Desktop App von GitHub zu verwenden. Dadurch habe ich gelernt mich für den für mich am besten geeigneten Weg zu entscheiden. Des Weiteren wurde mir bewusst, dass eine Alles in einem Lösung nicht immer die Optimaleste ist, da diese vor allem in diesem Fall weniger Kontrolle heißt.

## Zwischenfazit

Die Manipulation eines Buttons auf der Oberfläche, dem User Interface, empfand ich als sehr gut geeigneten Einstieg in dieses Assignment. Der JavaScript Code andere Developer ist an diesem Komplexitätspunkt gut nachzuvollziehen wodurch ein Einstieg erleichtert wurde.   
Neben der tatsächlich Implementierten Version der Manipulationsversion habe ich viele verschiedene Arten ausgetestet und konnte für mich die Vielfalt auf dieser niedrigen Komplexitätsebene entdecken. Dieses erste Interaktionselement dient in dem weiteren Verlauf des Assignments weiteren Manipulationen **(Manipulation x, y).**

# 2 Animation mit Canvas

Idee  
Die Bühne des Templates bietet viel Platz. Hier sollen ein Satz einige Sekunden nach öffnen der Website erscheinen. Die einzelnen Worte werden dabei zeitverzögert ausgesteuert.

## Theoretischer Hintergrund

Mit JavaScript können animierte Inhalte in Kopplung an das HTML-Canvas tag ausgesteuert werden. Das Canvas-tag stellt einen leeren Container auf der Website dar, in den mit Hilfe von JS Inhalte „gezeichnet“ werden können. Dies betrrifft Wörter, Formen, aber auch ganze Bilder (*HTML Canvas*, n.d.). Das Event, das hierbei i.d.R. die Animation auslöst, ist das Laden der Seite.  
Das Canvas kann in verschiedenen Kontexten verwedndet werden. Deshalb wird dieser nach dem Zugriff auf das HTML-Element mit *getElementByID()* mit der *getConntext()-Methode* festgelegt. Danach stehen alle Methoden dieses Kontextes zur Verfügung. Die meist genutzten Basisfunktionen sind dabei *moveTo(), lineTo, stroke(), beginPath(), arc(), font(), fillText(), strokeText(), drawImage()* (*HTML Canvas*, n.d.). Diese ermöglichen die Basisfunktionen im Canvas.   
Neben dem Canvas wird in dieser Manipulation außerdem die *setTimeout()-Methode* verwendet. Diese dient dazu Inhalte erst nach einer bestimmten Zeit auf der Website anzeigen zu lassen (*Window setTimeout() Method*, n.d.). In diese Methode kann eine Funktion eingebettet werden, die eine bestimmte Aktion nach der definierten Zeit ausführt (*jquery—How can i load javascript after some time?*, n.d.)(*javascript—Delay script loading*, n.d.). Diese Methode kann für unterschiedlichste Elemente angezeigt werden. Ursprünglich ist sie dafür gedacht Alertboxen anzuzeigen. Sie kann aber auch für andere Elemente wie das Canvas angewandt werden(*javascript—SetTimeout with canvas*, n.d.).

Praktische Umsetzung Programmieren  
Für die Umsetzung wurde dem HTML-Dokument ein Canvas Tag inklusive Größen definiert.   
Im JS werden drei Animationen auf dem gleichen Canvas durchgeführt. Diese sind über drei Funktionen definiert.   
In jeder Animationsfunktion wird zuerst die Methode setTimeout eingenestet. Innerhalb dieser wird dann das Canvaselement definiert. Dazu wird über die Methode *getElementByID()* auf das Canvas-Element zugegriffen und über *getContext()* der 2d-Kontext festgelegt. Anschließend werden die Methoden font(), fillStyle, filltext() an der zuvor initialisierten Canvas-Variablen aufgerufen. Diese definieren Schriftgröße und -art, Füllfarbe, Textinhalt und Position im Canvas. Anschließend wird die Funktion aufgerufen.  
Zum Ende wird nun noch das die Zeitverzögerung in Millisekunden angegeben. Diese ist aufsteigend gestaffelt damit die Textinhalte nacheinander auf der Website animiert werden.   
Somit liegt wird nun der Text I NEED YOU in großen roten lettern auf der Bühne des Templates mit einer Zeitverzögerung von einer Sekunde für I, zwei Sekunden für need und drei Sekunden für YOU ausgesteuert.

## Probleme und Hindernisse

Die Erstellung des Canvas an sich stellte keine Probleme dar. Allerdings bedurfte es einiger Recherche bezüglich der Positionierung. Das Canvas beitet nämlich zwei Arten der Position an: die des Canvas an sich und die der Inhalte. Erstere wird über CSS-Eigenschaften, Zweitere über JS gesteuert(*css—How to position canvas using relative/absolute positioning*, n.d.). Ein weiteres Problem stellte das Styling des Textes dar. Nach einer kurzen Recherche stellte ich fest, dass es verschieden Methoden für die Outlines *strokeText()* und die Füllung des Textes *fillText()* (Chinnathambi, n.d.) (*HTML Canvas Text*, n.d.)gibt.  
Nachdem das Canvas umgesetzt wurde fiel beim Neuladen des Inhaltes in einem neuen Tab auf, dass der Loop nicht die richtige Schrift von Anfang an abgriff und die Animation des ersten Buchstabens I zweimal aussteuerte. Somit erschien einmal ein I in der Basisschrift und einmal eins in der neu definierten.

## Zwischenfazit

Die zweite Manipulatoin des Assignments stellte mich vor komplexere Herausforderungen, die ich durch einige Code Sichtungen aber solide lösen konnte. Die größte dabei war die Methode setTimeout mit zu integrieren und eine verschachtelte Funktion zu erstellen. Nach dem Bewältigen der Aufgabe habe ich nun nicht nur gelernt wie ich Animationen mit Hilfe des Canvas-Tag einfügen kann, sondern auch wie ich selbst Funktionen schreiben und miteinander verbinden kann.

Goal Button:   
Get a text file with names of the working areas of the company. So use Ajax to integrate json data to display it on the webpage.

Overall Goal: 5 Interaktionselemente mit JS, die entweder an UI auf der Oberfläche gekoppelt sein können oder im Hintergrund ablaufen. ES6-features.

Funktionalität

Persönliche Reflexion

Fazit

Quellen

**Canvas Interaction**

<https://fontsgeek.com/fonts/franklin-gothic-demi-bold>

<https://stackoverflow.com/questions/12087874/settimeout-with-canvas>

stop timeout

<https://www.w3schools.com/jsref/met_win_cleartimeout.asp>

git

<https://intellij-support.jetbrains.com/hc/en-us/community/posts/115000445990-File-names-highlighted-in-red-Problem->

ajax und fetch api

<https://www.w3schools.com/xml/ajax_xmlhttprequest_create.asp>

<https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch>

<https://www.youtube.com/watch?v=rJesac0_Ftw>

<https://github.com/LearnWebCode/json-example/blob/master/animals-1.json>