Assignment 2

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

Table of Contents

[Projektvorhaben 1](#_Toc39868148)

[Vorkenntnisse und Vorbereitung 1](#_Toc39868149)

[Dokumente und Klassen 3](#_Toc39868150)

[Dokument- und Klassenstruktur 3](#_Toc39868151)

[Dokumentenfunktionalität 3](#_Toc39868152)

[Klassenfunktionalität 3](#_Toc39868153)

[Probleme und Hindernisse 3](#_Toc39868154)

[Persönliche Reflexion 3](#_Toc39868155)

[Fazit 3](#_Toc39868156)

[Literaturverzeichnis 3](#_Toc39868157)

# Projektvorhaben

Ziel des Projektes ist die Entwicklung einer funktionalen Typescript-Applikation, die im Browser durch die Kompilierung des verwendeten JavaScript über eine *index.html* verwendet werden kann. Die Art der Applikation liegt dabei im Entscheidungsfreiraum der Autorin. Diese entschied sich für ein Registrierungsformular, das einen *Usernamegenerator*, sowie einen *Password-Checker*. Diese Applikation *form-validator* wurde aus vom Dozierenden empfohlenen Liste mit Projekten ausgewählt(Traversy, 2020/2020). Dieses wurde lediglich zur Ideenfindung genutzt. Es wurde sich allerdings nicht an dem darin verwendeten Code orientiert, sondern ein eigener Ansatz frei von dem Beispielprojekt entwickelt. Das Vorhaben wurde außerdem mit dem Dozierenden abgeklärt und von diesem bestätigt.   
Der Usernamegenerator aus dem zuvor eingegebenen Uservor- und nachnamen inklusive einer zufällgi erzeugten Zahl einen Usernamen. Der Password-Checker überprüft zwei eingegebene Passwörter auf Similarität. Die Überprüfung wird durch zwei Aktionen der NutzerInnen ausgeführt: Verlassen des Feldes, sowie Klicken des Registrierungsbuttons. Dieser Button führt vor dem Absenden der Formulars die Passwortüberprüfung statt. Ist diese erfolgreich wird kurz vor dem Absenden ein entsprechender Text auf der NutzerInnenoberfläche ausgegegeben und das Formular wird abgesendet. Ist die Überprüfung nicht erfolgreich verliert der Button seine Funktion Daten zu übermitteln und teilt der NutzerInnen mit, dass sie die Passworteingabe korrekt wiederholen sollten um die Registrierung abzuschließen.  
Kriterien für die Entwicklung der App sind die Verwendung von Typescript (*folgend TS*), was die Anwendung typescirpttypischer Features wie z.B. Typisierung mit einschließt (*Angaben in Moodle*). Aus den in der Vorlesung genannten Kriterien geht hervor, dass Interfaces oder Klassen verwendet werden sollen. Des Weiteren sollen die verschiedenen TS-Dokumente mit einem Modulebundler gebundled werden.

# Vorkenntnisse und Vorbereitung

Die Vorkenntnisse zum Thema TS belaufen sich auf die theoretischen Vorlesungsinhalte, die zuvor vom Dozierenden vermittelt wurde. Da TS auf der Logik von Javascirpt und Java aufbaut sind zumindest Grundkenntnisse vorhaben. Jedoch müssen die Kenntnisse über entsprechend um die spezielle TS-Syntax und TS-features erweitert werden. Dazu wurde hauptsächlich ein sehr ausführliches Videotutorial mit zusätzlichen Übungen verwendet (*TypeScript Course for Beginners 2020*, n.d.). Zustäzlich wurde sich nochmal über die Vorteile der Nutzung von TS für die Entwicklung informiert (Pramono, 2019). Da die Typisierung der hauptsächliche Vorteil ist wurde nochmal tiefer über die in TS zu verwendenden Typen recherchiert(*Basic Types in TypeScript*, n.d.). Dadurch konnte das theoretische Wissen um einen ersten Überblick in der praktischen Anwendung erweitert werden. Dies erleichterte den Einstieg in das praktische Aufsetzen des Projektes immens und ist sehr weiterzuempfehlen für TS-EinsteigerInnen.  
Die grundlegende Struktur für das Projekt wurde sich durch Clonen des, vom Dozenten zur Verfügung gestellten, Git-Repositories auf *GitHub* angelegt(Slezak, 2020/2020). Da es keine Vorkenntnisse im Umgang mit Formularen generell gibt wird zuerst eine allgemeine Recherche rund um das Thema *Form Validation* durchgeführt. Die Recherche zeigt, dass es verschiedene Arten, sowie unterschiedliche Herangehensweisen gibt. Es wird sich für eine *Clinet-side Form Validation.* Diese wird außerdem extern mit TS durchgfeführt, was an wenigen Stellen zusätzlich mit einer Built-In Form Validation innerhalb des HTML-Dokumentes ergänzt wird(*Client-Side Form Validation by MDN*, n.d.).  
 Um einen ersten Überblick über das Vorhaben zu erhalten wird im ersten Schritt eine größtenteils funktionslose HTML-Datei inklusive zugehörigem CSS-Dokument erstellt. Die Kenntnisse im grundlegenden Umgang mit HTML und CSS sind an diesem Punkt des Studiums bereits verfestigt. Trotzdem bedurfte es einiger kleiner Nachrechercher in bezug auf den Aufbau eines Formulars mittels Formular- und Input-tags (*Html - Background Text in Input Type Text*, n.d. ) (*Change Input Placeholder Color*, n.d.)(*HTML Input Required Attribute*, n.d.) Damit werden die Bestanteile des Projektes in ihren Grundzügen gelegt und im nächsten Schritt kann ihnen eine Funktion mit Hilfe von TS zugeordnet werden.  
Damit diese Funktionen auch abgerufen werden können wird der TypeScript-Compiler im Projekt installiert. Darüber hinaus wird das Projekt mit einem *lite-server*, ein zur freien Verfügung stehender Server für Entwicklungsvorhaben, verbunden und gleichzeitig eine automatische Synchronisierung zwischen diesem und den Dateien des Projektes eingebaut. Dazu werden die entsprechenden Anpassungen in der package.json Datei im TS-Projekt gemacht. Dies kostet im ersten Schritt Zeit, erspart aber das händische Kompilieren und Aufrufen im browser bei jeder Änderung während des Projektes (*TypeScript Course for Beginners 2020*, n.d.).  
Da bei Bearbeitung des Projektes keine bis wenige Kenntnisse über die praktische Durchführung des Bundlens von Modulen vorliegt werden auch dazu weitere Informationen eingeholt (*Rollup.Js Dokumentation*, n.d.)(*Java Script Rollup Bundle Tutorial*, n.d.). Diese beziehen sich auf die Installation des verwendeten Modulebundler rollup.js, sowie die Verknüpfung von Dateien und Klassen mit Hilfe von *import* und *export* innerhalb der Dateien(*Modules in TypeScript*, n.d.). Auf Basis dessen kann der gewählte Modlebundler erfolgreich eingebaut und die Module gebundled werden.   
Anschließend wird seine Konfiguration in der rollup.config.js-Datei so angepasst, dass bei jeder Kompilierung Datei index.ts zu einem kombinierten bundle.js-Datei wird. Diese kann dann vom Browser ausgelesen und somit die Funktion ausgesteuert werden. Des Weiteren werden in der ts.config-Datei die Root- und Sourceordner für die Kompilierung angegeben, sowie der Version des im Projekt angewandten ECMAScript auf ES6 eingestellt.  
Mit diesen Grundeinstellungen als Basis können im nächsten Schritt die Funktionalitäten der App-elemente mit TS implementiert werden. Zur leichteren Verfolgung der weiteren Schritte dient folgender Screenshot der Applikation:

A picture containing sitting, computer, monitor, photo

Description automatically generated

Abbildung 1: Screenshot Typescript Applikation, 09.05.2020

Anfangs: schwierigkeiten zu verstehen wo ich dort nun typescript unterbrigen kann weil man das auch über normale html-pattern und normale code snippets in css lösen könnte. Bei so einem kleinen projekt scheinen klassen etwas überflüssig, aber okay.

05.05: Meeting Malte Weiersbach: hat mit mir den code angesehen und geschaut wie man ts dort einbringen könnte.

Aufbau Usernamegenerator: Input Vorname + Input Nachname + Randomnumber  
  
Problems: understanding how function and variables work together and how I can grasp the input of input fields and show my results to an HTML-input field.

Only anable letters in input field würde man eig mit HTML machen: <input onkeyup = „lettersonly(this)”>.

# Dokumente und Klassen

## Dokumentenstruktur- und funktionalität

Die Dokumentenstruktur ist folgendermaßen aufgebaut:

|  |  |
| --- | --- |
| A screenshot of a cell phone  Description automatically generated  Abbildung 2: Ordnerstruktur innerhalb der IDE: oberste Ebene | A screenshot of a cell phone  Description automatically generated  Abbildung 3: Ordnerstruktur innerhalb der IDE: gesamt |

Das Projekt aus 3 Ebenen: auf der untersten Ebene befinden sich die kompilierten JS-Dokumente der einzelnen Klassen (src-Ordner). Diese wurden aus den darüber, auf der zweiten Ebene, liegenden TS-Dokumenten kompiliert. Auf dieser Ebene befindet sich außerdem die index.ts. Diese führt alle einzelnen Module in einem TS-Dokument zusammen. Ausgehend von diesem wird auf der ebenfalls zweiten Ebene innerhalb des darüberliegenden build-Ordners das gebundlete bundle.js-Dokument erstellt. Des Weiteren befinden sich auf der zweiten Ebene weitere verwendete Inhalte wie das Hintergrunddokument im assets-Ordner.   
Auf der oberen Ebene sind nun die übergreifenden Ordner wie assets, build, src. Eine Besonderheit stellt der node-modules-Ordner dar, der in der Grundeinstellung der DIE versteckt wird. Er beinhaltet alle Node-Module, die zusätzlich verwendet werden können. Dies trifft in diesem Projekt z.B. auf den Lite-Server zu. Da der Ordner entsprechend viele Unterordner enthält wird er hier nur in eingeklappter Version dargestellt. Auf der obersten Ebene gipfelt dann die Kette des Modulebundlings in der index.html. In dieser ist die darunter, im build-Ordner liegende, bundle.ts mittels eines <script>-tags verbunden. Somit werden hier die aus den einzelnen TS-Klassen kompilierten und gebundelten Dateien in die HTML-Datei, die im Brwoser ausgesteuert wird integriert. Der Bundleprozess kann noch einmal bildlich im Abschnitt *Klassenfunktionalität* nachvollzogen werden. Außerdem ist nebend er Dokumentation eine datei zu finden, die die Installation ders Typescirpt-Compilers beschreibt. Daneben sind des Weiteren die Konfigurationsdateien für den Modulebundler und Typescript zu finden.

## Klassenfunktionalität

Um den Umgang mit mehreren Typescript-Modulen demonstrieren zu können wurden für das projekt mehrere Klassen angelegt, die teilweise voneinander erben, also Funktionalität übernehmen können. Die Typescript-Klassenhierarchie baut sich folgendermaßen auf:

Abbildung 4: Klassenstruktur und Bundleprozess

**Hierarchie 1**  
MethodsPasswordCheck

ApplyPasswordCheck

**Hierarchie 2**  
MethodsUsernameGenerator

ApplyUsernameGenerator

index.ts 🡪 bundle.js 🡪 index.html

Die Methods-Module legen dabei die grundlegenden Methoden fest, die an die Apply-Klassen weitervererbt werden. Diese führen dann diese Methoden an HTML-Objekten aus. Die beiden Modularten wurden miteinander verbunden, indem sie exportiert bzw. importiert wurden(*Modules in TypeScript*, n.d.). Hierbei ist wichtig zu beachten, dass nicht die gesamte Datei importiert wird, sondern lediglich die Klasse in dieser Datei (*Import Typescript Files*, n.d.). Der Dateienimport wird in der index.ts verwendet. In dieser werden dann die Anwendungsklassen, die über Vererbung mit den Methodenklassen verbunden sind als Dateien importiert. Diese index.ts bildet nun die Grundlage für das Bundlen in die bundle.js-Datei, die wiederum das HTML-Dokument im Browser speist.   
Nun liegt die grundlegende Verbidnung zwischen Typescirpt, JavaScript und HTML. Allerdings muss der Prozess nicht nur in eine Richtung zu dem HTML-Dokument hin funktionieren, sondern auch zurück. Die HTML-Objekte müssen innerhalb der TS-Klassen angesprochen und mit Funktionen ausgestatte werden. Dazu werden die in der jweilig verwendeten Klasse erst Variablen deklariert und typisiert und anschließend mit dem entsprechenden HTML-Element durch document.getElementByID() initialisiert (*Connect InnerHTMl with JavaScript*, n.d.). Wichtig ist hierbei der verwendete Typ. Grundlegend werden die in diesem Projekt die HTML-Objekte als Typen HTMLElement, HTMLButtonElement und HTMLInputElement (*HTMLInputElement*, n.d.) verwendet. Dabei wurde eine spezielle Schreibweise bei der Initialisierung verwendet. Mit Hilfe des Ausrufungszeihens *!* wird dem Kompilierer zugesichert, dass dieses HTML-Element auch ind er HTML-index existiert. Ohne dieses kam es im Verlauf dieses Projektes zu Fehlermeldungen. Das Ausrufungszeichen ist ein sogenanntes Workaround um diese(*TypeScript Course for Beginners 2020*, n.d.). das HTML-Elemetn ist mit diesen Schritten initialisiert und typisiert und liegt in der Klasse als weiter zu verndende Variable ab.

Grundlegend besteht jede TS Klasse aus einem Konstruktor, den zu verwendenden Variablen und Methoden (Motto, n.d.). Die Methoden wurden hierbei, wie bereits erwähnt, in separate Klassen ausgelagert.

## Klasse MethodsUsernameGenerator

Diese Klasse bezieht sich auf den Umgang mit den Eingabefelder Vorname, Nachname und Username. Durch build-in Form Validation wurde den ersten beiden Feldern das HTML-Attribut *required* und dem letzteren *disabled* zugewiesen. Dadurch wird die NutzerInneneingabe in den ersten beiden Feldern verppflichtend, wogegen es ihnen nicht erlaubt ist etwas in das dritte Feld einzutragen.

.A picture containing sitting, computer, monitor, photo

Description automatically generated

Abbildung 5: Durch Klasse MethodsUsernameGenerator bediente Felder

Im ditten Feld wird basierend auf der vorangegangenen Eingabe aus dem Vornamen, Nachnamen und einer zufällig generierten Zahl ein Username erstellt.

A close up of a sign

Description automatically generated

Abbildung 6: generierter Username

Dazu wird eine statische Methode *getRandomInt()* erstellt, die eine zufällig generierte Zahl erstellt (Cardillo, 2018). Diese Methode ist statisch, da sie ausshcließlich in dieser Klasse verwendet wird. Diese wird inerhalb die Methode *addInput()* aufgerufen. An diese werden die beiden NutzerInneneingaben als string übergeben und mit der zufällig generierten Zahl kombiniert. Wurde der Username dadurch generiert wird außerdem ein Text in der Konsole ausgesteuert, der darüber informiert, ob dieser Prozess erfolgreich war. Ausgeführt werden die beiden Methoden in der erbenden Klasse *ApplyUsernameGenerator.*

## ApplyUsernameGenerator

Damit die Klasse die Methoden von *MethodsUsernameGenerator* erben kann, wird dieser importiert. Anschließend werden die zu anzusprechenden HTML-Elemente (die ersten drei EIngabefelder) mit document.getElementByID() Variablen zugewiesen und entsprechend typisiert. Anschließend wird dann die Klasse *MethodsUsernameGenerator* instanziiert und einer Konstanten überwiesen. Dabei werden die Werte der Eingabefelder an die Methoden der Klasse weitergegeben, sowie diese aufgerufen und ausgeführt. Der Wert des dritten Eingabgefeldes, das für NutzerInnen nicht zugänglich ist wird anschließend mit dem soeben generierten Username überschrieben und somit auf dem Screen sichtbar gemacht. Diese gesamte Funktionalität wird einem *EventListener* am zweiten Eingabefeld zugeordnet, der auf das Event *blur* reagiert. Sobald NutzerInnen das Eingabefeld für den Nachnamen verlassen erscheint der neue Username im dritten Eingabefeld automatisch. Damit diese gesamte Klassenfunktinalität ausgeführt wird bei Seitenaufruf wird die gesamte Klasse *ApplyUsernameGenerator* abschließend mit dem new Operator instanziiert.

<https://stackoverflow.com/questions/34110766/how-to-show-hide-multiple-divs-based-on-input-value>

Aufbau Klassen: Konstruktoren etc.

welche events gibt es? <https://www.w3schools.com/jsref/dom_obj_event.asp>

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

MethodsPasswordCheck

<https://stackoverflow.com/questions/14702190/to-show-error-message-without-alert-box-in-java-script>

<https://stackoverflow.com/questions/38639604/how-do-i-activate-deactivate-css-on-button-click>

<https://stackoverflow.com/questions/21070101/show-hide-div-using-javascript>

ApplyPasswordCheck

<https://stackoverflow.com/questions/16117290/display-the-result-of-a-javascript-function-in-a-div-element/16117312>

<https://stackoverflow.com/questions/48100152/hide-elements-using-typescript/48100271>

<https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement/hidden>

<https://stackoverflow.com/questions/24834595/hide-html-element-using-javascript>

wie wird submittet: damit dozent auch sachen sehen kann etc:

<https://developer.mozilla.org/en-US/docs/Web/API/HTMLFormElement/submit_event>

# Probleme und Hindernisse

Die Erstellung der Klassenstruktur erfolgte nicht intuitiv und benötigte einige Iterationen. Es fiel besonderes schwer Klassen für eine solch kleine Applikation zu bauen. Nach Rücksprache mit meinem Betreuer aus dem Ausbildungsbetrieb wurde die Klassenstruktur, so wie sie im projekt vorhanden ist letzten Endes auch angelegt. Allerdings konnten wir beide und nach Rückmeldung aus dem Studiengang auch viele andere die Idee dieser „künstlich erzeugten“ Zerstückelung in Module nicht ganz nachvollziehen. Bis zum ende des Projektes wurde mir der Mehrwert dessen in einem so kleinen Projekt nicht ganz bewusst. Die Idee ist klar, dass der generelle Umgang mit dem Modulebundler und Klassen erlernt werden soll. Allerdings war es eine Herausforderung Code in einzelne Teile zu zerlegen, die ich intutiv nicht zerlegt hätte.

Zu Beginn nicht klar wie TS und HTML interagieren, da Komponenten von Dozierendenbeispiel verwirrend. Jetzt klar, dass TS quasi nur die Funktionen Modularisiert.

Konnte node modules nicht finden.

<https://intellij-support.jetbrains.com/hc/en-us/community/posts/207297985-Can-t-see-excluded-folders>

<https://stackoverflow.com/questions/26107125/cannot-read-property-addeventlistener-of-null>

<https://stackoverflow.com/questions/12930049/how-do-i-import-other-typescript-files>

<https://stackoverflow.com/questions/16117290/display-the-result-of-a-javascript-function-in-a-div-element/16117312>

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

<https://stackoverflow.com/questions/52325814/why-we-are-using-htmlinputelement-in-typescript>

Zu Beginn: Probleme mit Klassen erstellen: <https://ultimatecourses.com/blog/typescript-classes-and-constructors>

Verständnis Variablen:

<https://www.tutorialspoint.com/typescript/typescript_variables.htm>

Erkenntnis mit private und public etc.

<https://stackoverflow.com/questions/34110766/how-to-show-hide-multiple-divs-based-on-input-value>

# Persönliche Reflexion

Der häugiste Satz, den ich während dieses Assignments gehört habe war „ das ist eigentlich JAvascript, nur typisiert“. Was auch stimmt, bis es zu den 100 Ausnahmen kommt, die es in Typescript gibt…

# Fazit

# Literaturverzeichnis

Interfaces

<https://www.logicbig.com/tutorials/misc/typescript/type-assertions.html>

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

Backgroundbild: <https://stock.adobe.com/de/search/images?filters%5Bcontent_type%3Aphoto%5D=1&filters%5Bcontent_type%3Aillustration%5D=1&filters%5Bcontent_type%3Azip_vector%5D=1&filters%5Bcontent_type%3Avideo%5D=0&filters%5Bcontent_type%3Atemplate%5D=0&filters%5Bcontent_type%3A3d%5D=0&filters%5Bcontent_type%3Aimage%5D=1&filters%5Binclude_stock_enterprise%5D=0&filters%5Bis_editorial%5D=0&order=relevance&price%5B%24%5D=1&safe_search=1&serie_id=302616645&search_page=1&search_type=see-more&get_facets=0&asset_id=302616645>