

# Evaluation moderner Webtechnologien für die Entwicklung eines modularen Supportportals

## MASTER PROJEKT

für die Prüfung zum

Master of Science

des Studienganges Angewandte Informatik

an der

Fachhochschule Erfurt

von

**Sebastian Rieger**

Abgabedatum 01.03.2017

Bearbeitungszeitraum	24 Wochen
Matrikelnummer	10286908
Ausbildungsfirma	PDV Systeme Erfurt
Betreuer der Ausbildungsfirma	Dipl. -Inform. FH Nico Kaiser
Gutachter der Fachhochschule	Prof. Rolf Kruse

## Erklärung

Ich, Sebastian Rieger, versichere hiermit, dass ich die vorliegende Bachelorarbeit mit dem Thema

Evaluation moderner Webtechnologien für die Entwicklung eines modularen Supportportals

selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

---

Ort    Datum

---

Unterschrift

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>5</b>
<b>2</b>	<b>Portalserver/ CMS-Systeme im Vergleich</b>	<b>6</b>
2.1	Typo3 . . . . .	6
2.2	Typo3 Neos . . . . .	7
2.3	Joomla . . . . .	8
2.4	Drupal . . . . .	9
2.5	Auswertung der möglichkeiten . . . . .	9
<b>3</b>	<b>Typo3</b>	<b>10</b>
3.1	Typo3 8.2 . . . . .	10
3.2	Extensions . . . . .	10
3.3	TypoScript . . . . .	10
3.4	Fluid . . . . .	10
<b>4</b>	<b>Modulare Extensions</b>	<b>11</b>
<b>5</b>	<b>Neue Webtechnologien</b>	<b>12</b>
5.1	Web Components . . . . .	12
5.1.1	Shadow DOM . . . . .	13
5.2	Google Polymer . . . . .	13
5.2.1	Wozu Polymer Elemente . . . . .	13
5.2.2	Was macht Polymer aus . . . . .	14
5.2.3	Ein eigenes Polymer Element erstellen . . . . .	14
5.2.4	Zusammenfassung . . . . .	16
5.3	AngularJS . . . . .	16
5.4	HTML5 . . . . .	17
5.5	CSS3 . . . . .	17
5.5.1	Bootstrap . . . . .	17
5.5.2	MaterializeCSS . . . . .	18
5.5.3	Vergleich beider CSS-Frameworks . . . . .	18
5.6	PHP7 . . . . .	18
5.7	Google Dart . . . . .	18
<b>6</b>	<b>Zusammenspiel der Technologien</b>	<b>19</b>
6.1	Integration von Polymer in Typo3 . . . . .	19

6.2	MaterialzeCSS . . . . .	22
6.3	Nutzer Anmeldung und Registrierung . . . . .	22
<b>7</b>	<b>Zusammenfassung</b>	<b>22</b>
<b>8</b>	<b>Fazit</b>	<b>22</b>
<b>9</b>	<b>Abkürzungsverzeichnis</b>	<b>23</b>

# 1 Einleitung

In einer vernetzten Welt wie unserer, werden unablässig neue und bessere Web-Technologien entwickelt. Diese neuen Technologien bringen zum einen eine bessere Programmierfreundlichkeit mit sich, aber sie sind zum anderen durch neue Ansätze auch schneller als ältere Technologien.

Heutige Webanwendungen müssen eine Vielzahl an Kriterien erfüllen. Eine gute Webseite sollte heute möglichst auf einen Handy, sowie auf einem Desktop-PC oder Fernseher darstellbar sein. Um dies zu ermöglichen, müssen besondere Anforderungen an das Design gestellt werden. Natürlich muss sie auch von möglichst allen Browsern anzeigbar sein. Ein weiterer Punkt ist, dass Webseiten sich heute so flüssig und performant wie ein natives Programm verhalten sollen. Hierbei ist es natürlich auch wichtig vorhandene Hardware anzusprechen und mit ihr zu interagieren.

Das Ziel dieser Arbeit ist es herauszufinden, wie unter Zuhilfenahme moderner Technologien alle diese Kriterien möglichst gut erfüllt werden können. Es sollen Programmieransätze wie Google Polymer, AngularJS, HTML5, CSS3, PHP7 und Google Dart unter dem Portalserver vereint werden, welcher die Webseite beziehungsweise die Anwendung bereitstellt.

Es soll geprüft werden, wie und ob es möglich ist diese verschiedenen Technologien in möglichst modularen Portalserver-Erweiterungen unterzubringen, um eine möglichst performante und zu gleich leistungsstarke Webseite zu entwickeln. Dieses Oxymoron aufzulösen ist der Schwerpunkt der vorliegenden Arbeit, obwohl doch schon jetzt ersichtlich ist, dass ein auf beiden Seiten ein Kompromiss gefunden werden muss. Wie dieser genau aussieht, wird im weiteren Verlauf eine wichtige Rolle spielen.

Diese Arbeit soll der theoretischen Grundstock für eine weitere Arbeit sein, in der das Support-Portal der PDV System Erfurt GmbH neu entwickelt wird. Im Verlauf dieser Arbeit, wird evaluiert werden, welcher der gängigen Portalserver für ein Supportportal am besten geeignet ist. Außerdem wird geprüft werden, wie es möglich ist, modulare Erweiterungen für einen solchen Server zu entwickeln, um unnötige Datenredundanz zu vermeiden.

Durch einen modularen Aufbau, soll außerdem die Kommunikation zwischen verschiedenen Erweiterungen vereinfacht werden. Des Weiteren soll es möglich werden, nur beliebige Erweiterungen zu verwenden, wenn für den jeweiligen Use-Case nicht alle benötigt werden.

In den nun folgenden Abschnitten werden Programmierbeispiele und Hinweise gegeben, wie eine solche Neuentwicklung unter den Gesichtspunkten Performance, Umsetzbarkeit und Usability vorgenommen werden kann. Um Mitarbeitern und Kunden ein möglichst performantes Supportportal bieten zu können, welches den Arbeitsalltag eines jeden Nutzers erleichtert.

## 2 Portalserver/ CMS-Systeme im Vergleich

Das zukünftige Supportportal der PDV Systeme GmbH soll auf Basis eines Portalserver bzw. *Content Management System* (CMS)-Servers aufgebaut werden. Hierfür werden im folgenden einige Möglichkeiten genauer betrachtet.

Ein Portalserver, welcher für das Projekt heran gezogen wird muss die folgenden Eigenschaften aufweisen.

- Webseiten müssen frei gestaltbar sein
- Es muss die Möglichkeit bestehen Anwendungen für den Server zu entwickeln
- Das System muss Open-Source sein, um ggf. in den Quellcode eingreifen zu können
- Die Nutzer-Community sollte möglichst groß sein, damit Probleme leicht diskutiert und behoben werden können
- Der Server muss die Möglichkeit bieten Dateien zu verwalten, welche als Download oder Kontent in das Portal einfließen
- Lauffähig unter einer SQL-Datenbank wie MySQL oder MariaDB

Auf die Betrachtung reiner CMS-System wird an dieser Stelle verzichtet, da diese nicht die gewünschten Anforderungen eines Portalserver erfüllen. Ein Vergleich verschiedener reiner CMS-Systeme ist in der Bachelorarbeit „Konzept und prototypische Implementierung eines übergreifenden Dokumenten- und Medienmanagements“ zu finden. [Rie15]

### 2.1 Typo3

Typo3 ist ein Verwaltungssystem für Internetseiten. Es basiert in der neusten Version 8.2 auf der PHP Version 7. Seit der Version 7.0 welche zugleich eine *Long Term Support* (LTS) Version ist, wird es unter dem Namen Typo3 CMS vertrieben. [Wik16a]

Es ist möglich unter Verwendung von PHP, Extbase und Fluid eigene Erweiterungen für den Portalserver zu entwickeln. Die jeweiligen Erweiterungen wirken im Fronten von Typo3 wie ganz normale Webseiten. Auf Extbase und Fluid wird im Kapitel 3 genauer eingegangen.

Ein Vorteil von Typo3 ist, dass es eines der am häufigsten verbreiteten Portalserver auf dem Markt ist. Selbst große Firmen wie „Sixt“ setzten auf Typo3 bei der Erstellung ihrer Internetportale. Durch die große Verbreitung von Typo3 ist auch die Community um die Software sehr groß und man findet schnell zu fast jedem Problem im Internet eine Lösung.

Typo3 kann im Zusammenspiel mit MySQL, MariaDB, PostgreSQL oder Oracle als Datenbank betrieben werden. Nur mit Hilfe einer dieser Datenbank im Hintergrund ist Typo3 stabil Lauffähig und kann produktiv eingesetzt werden.

Durch ein übersichtliches Backend (siehe Abbildung 1), ist es auch für Laien möglich qualitativ hochwertige Webseiten zu erstellen, ohne viel Kenntnis von Webtechnologien wie HTML oder CSS zu haben.

Ein Upload von Dateien für den Download beziehungsweise als Seiten-Kontent ist ebenfalls unter Typo3 möglich. Zusätzlich dazu ist es möglich das Nutzer Dateien auf den hochladen. Hierdurch entsteht auch die Möglichkeit ein Austauschportal für Dateien zu schaffen. Nutzer können so in einer sicheren Umgebung sensible Daten untereinander oder mit Mitarbeitern austauschen. [Lob16]



Abbildung 1: TYPO3 Backend im Seitenbearbeitungs Modus [Tea16]

## 2.2 TYPO3 Neos

TYPO3 Neos ist ein relativ neuer Internetportal Server, welcher 2012 aus dem Wunsch heraus entstand TYPO3 zukunftssicher zu gestalten.

Als erkannt wurde, dass eine zukunftssichere TYPO3 Entwicklung basierend auf dem *Model View Controller* (MVC)-Prinzip eine komplette Neuimplementierung erfordert wurde der Grundstein für TYPO3 Neos gelegt. Neben der Entwicklung von TYPO3 CMS wurde die Entwicklung von TYPO3 Neos begonnen. Das Ziel war es, einen Nachfolger für TYPO3 zu entwickeln. Dies gelang jedoch nicht wie vorgesehen. [Wik16a]

Früh wurde festgestellt das sich diese die beiden Projekte in unterschiedliche Richtungen entwickeln. TYPO3 Neos ist heute in der Version 2.0 erhältlich und hat sich zum Ziel gesetzt Webseiten im gegensatz zu TYPO3 live im Frontend zu bearbeiten. Einfach gesagt ist TYPO3 ein Internet-Portalserver welcher auf dem *What you see is what what you get* (WYSIWYG)-Prinzip aufbaut. Es ist möglich Webseiten live oder getrennt vom Livesystem im Frontend zu erstellen. Die Frontend-Bearbeitung bingt viele Vorteile für die Webseitenerstellung für Laien. Diese können eine Seite direkt bearbeiten und Live nehmen. [Lob16]

Neos ist bei weitem nicht so verbreitet wie TYPO3 CMS und die Community um das Projekt ist deutlich geringer. Dies ist ein klarer Nachteil zum großen Bruder TYPO3 CMS ist.

Ähnlich wie bei TYPO3 CMS ist es in Neos möglich sogenannte Plugins zu entwickeln. Diese basieren auf dem „TYPO3 Flow“-Framework, welche zusammen mit Neos entwickelt wurde. Flow ist ein Framework, welches es erlaubt Templates mit Hilfe von „Fluid“ zu entwickeln. Auf die Fluid wird im Abschnitt 3.4 näher eingegangen. [Wik16b]

Ähnlich wie TYPO3 CMS benötigt auch Neos eine SQL-Datenbank zur Datenhaltung und ist Quelloffen verfügbar. Nach der Abspaltung von TYPO3 Neos vom TYPO3 Projekt, wird es heute

von einer relativ kleinen Gemeinde entwickelt. Aus diesem Grund, gibt es keine Roadmap für das Projekt und die Veröffentlichung von neuen Version ist träger als die von Typo3 CMS.

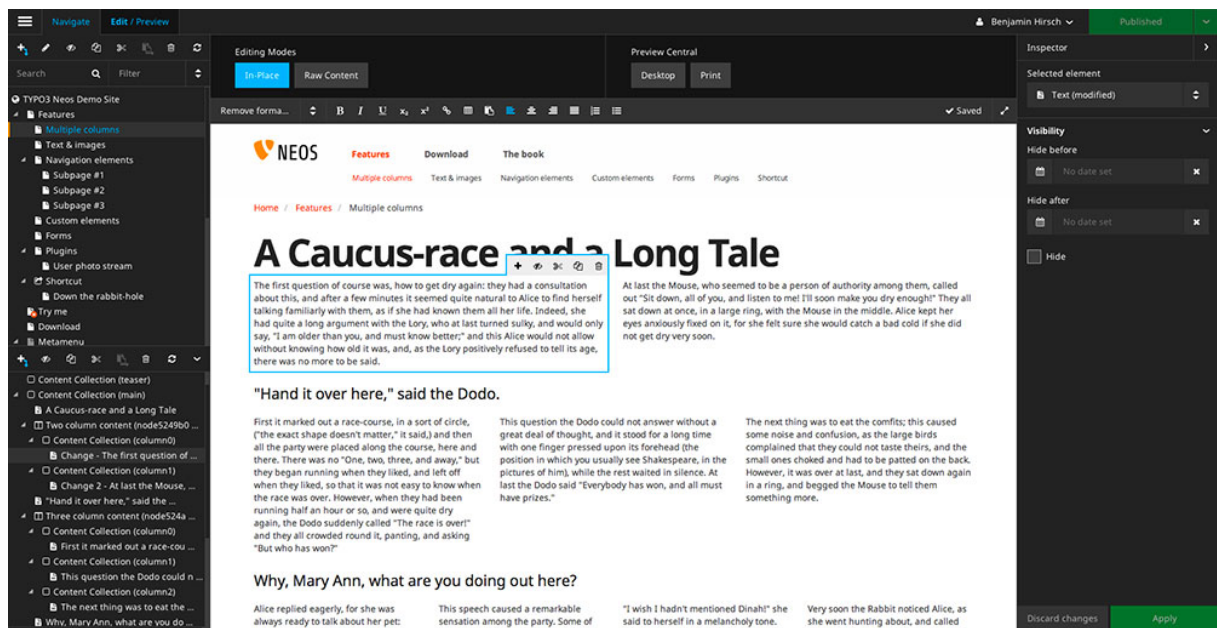


Abbildung 2: Typo3 Neos Backend im Seitenbearbeitungs Modus [HDB14]

In Abbildung 2 ist der Seitenbearbeitungs-Modus von Neos zu sehen. Es ist gut zu erkennen, dass Inhalte einer Webseite direkt bearbeitet werden können. Dies macht eine Webentwicklung für Laien einfacher.

## 2.3 Joomla

Joomla ist ein weiteres CMS-System für die Erstellung von Webseiten. Ähnlich wie Typo3 ist es Open Source und nutzt eine MySQL-Datenbank im Hintergrund. Das System ist in PHP5 geschrieben und dient in erster Linie zur Erstellung von Webseiten. Am besten kann Joomla mit Typo3 Neos verglichen werden, da es einen ähnlichen Ansatz zur Erstellung von Webseiten aufgreift.

Ein Vorteil gegenüber Neos ist, dass Joomla weit verbreitet ist und eine beachtliche Community hat, welche bei Neos geringer ausfällt. Nachteilig an Neos ist jedoch, dass es eine Bearbeitung von Web-Inhalten nur im Backend zulässt. Dies bedeutet wiederum, dass mehr Erfahrung benötigt wird, um eine Webseite zu erstellen.

Weitere Nachteile sind zum einen die kaum existente Roadmap, welche zur Zeit der Bearbeitung nicht aktuell war. Zum anderen ist die Entwicklung von Erweiterungen für Joomla nicht so gut strukturiert wie es bei Typo3 mit Fluid der Fall ist.

Eine freie Gestaltung der Webseiten ist zwar gegeben, aber für die Entwicklung eines komplexen Supportportals wie es später einmal entstehen soll ist Joomla nicht oder nur bedingt geeignet. Es gibt momentan keine garantierte Weiterentwicklung des Portalservers und eine Unterstützung von MariaDB ist nicht offiziell gegeben.

Auch wenn Joomla durchaus seinen Charm hat, so ist es für die Entwicklung eines Supportportals, welches modular aufgebaut werden soll nicht zu empfehlen. Dies bedingt sich zum einen aus der abgelaufenen Roadmap, aber auch durch die schlechtere Erweiterungsentwicklung im Gegensatz zu Typo3 CMS oder Neos.



## **2.4 Drupal**

## **2.5 Auswertung der möglichkeiten**

Auch wenn Typo3 Neos Typo3 CMS in nichts nachsteht, so wurde sich dennoch gegen die Verwendung von Neos entschieden, da eine vorranschreitende Entwicklung nicht gewährleistet. Im Punkto Zukunftssicherheit ist Typo3 CMS deutlich besser aufgestellt.

## **3 Typo3**

### **3.1 Typo3 8.2**

### **3.2 Extensions**

### **3.3 TypoScript**

### **3.4 Fluid**

## 4 Modulare Extensions

## 5 Neue Webtechnologien

Im folgenden Kapitel wird auf neue Webtechnologien eingegangen, welche für die Verwendung eines modularen Supportportals in Frage kommen.

### 5.1 Web Components

Web Components sind eine neue Technologie zur Erstellung von Web-Seiten. Die Spezifikation des W3C für Web Components sehen, wie der Name sagt, vor Komponenten speziell für Web-Seiten zu erstellen. Diese Komponenten können entweder vom Programmierer selbst oder aus Fremdquellen stammen.

Durch die Verwendung soll die Programmierung von Webtechnologien vereinfacht werden. Komponenten kapseln spezielle Elemente wie zum Beispiel Tabellen oder Button mit ihren Funktionen zusammen in ein Element. Komponenten können also auf einer Vielzahl von Technologien aufbauen. Die Nutzung einer Komponente durch einen Programmierer geht somit auch ohne die Kenntnis aller in der Komponente verwendeten Technologien. Ein Programmierer muss sich nicht um die Funktion oder den Aufbau einer Komponente kümmern. Er kann sie einfach benutzen.

Dies bringt zum einen den Vorteil, dass Programmierer sich nicht mehr mit allen Technologien beschäftigen und auskennen muss. Zum anderen können Komponenten schnell in bestehende und neue Projekte integriert werden. Hierdurch kann die Entwicklungszeit einer Web-Seite im Idealfall drastisch gesenkt werden. [Ihl13]

Web Components bestehen nach der Definition des W3C aus fünf Bestandteilen:

- Templates
- Decorators
- Custom Elements
- Shadow DOM
- Imports

**Templates** sind HTML-Teile, welche beim rendern einer Web Komponente zum tragen kommen. Sie werden innerhalb der Komponente definiert (siehe Listing 1) und sind innerhalb von HTML5 standardisiert.

**Decorators** beschreiben das Aussehen eines Templates innerhalb einer Komponente mit Hilfe von CSS. Zum aktuellen Zeitpunkt sind sie jedoch nicht spezifiziert.

**Custom Elements** sind Komponenten, die vom Programmierer selbst erstellt wurden. Sie bilden den Grundstein für wiederverwendbare und eigene Web Components.

**Shadow DOM** ist eine HTML-Struktur, welche vom normalen HTML-DOM aus nicht beeinflusst werden kann (siehe Abschnitt 5.1.1).

**Imports** sind Anweisungen, die dem Browser sagen welche Komponenten er zusätzlich laden muss. Sie sind essentiell, damit Web Components überhaupt in eine DOM-Struktur eingebunden werden können. Imports sind wie Templates ebenfalls schon in HTML standardisiert. [CG13]

### 5.1.1 Shadow DOM

Damit Komponenten nicht durch CSS-Klassen von der eigentlichen Web-Seite oder untereinander beeinflussen, wurde der Shadow DOM erfunden.

Der Shadow DOM ist ein konstrukt um Komponenten vor dem eigentlichen HTML-DOM zu verbergen. Dieses vorgehen ist notwendig, um Komponenten verwenden zu können, denn Komponenten sollen sich weder untereinander Beeinflussen noch sollen sie von der Web-Seite beeinflusst werden.

Komponenten im Shadow DOM werden vor dem eigentlich HTML-DOM verborgen, wie der Name schon sagt, liegen sie im Schatten. Der Browser rendert also zuerst das HTML-DOM und anschließend den Shadow DOM, ohne das dieser vom eigentlichen DOM beeinflusst wird. JQuery aufrufe auf der Hautseite zum Beispiel, können sich nicht auf Elemente im Shadow beziehen, da sie diese nicht kennen.

Zur aktuellen Stunde, können alle modernen Browser den Shadow DOM umsetzen. [Pol16b]

## 5.2 Google Polymer

Das Polymer Projekt wird seit 2012 von Google entwickelt und basiert auf dem Web Component-Standard des W3C. Polymer ist eine Library, welche mit Hilfe des Shadow DOM arbeitet.

Die erste stabile Version von Polymer wurde im März 2016 veröffentlicht und ist stark von AngularJS (siehe Abschnitt 5.3) beeinflusst worden.

Zur Zeit der Bearbeitung ist Polymer in der Version 1.0 veröffentlicht. Jedoch wird momentan schon an der Version 2.0 gearbeitet, welche auch als Beta verfügbar ist.

In den folgenden Abschnitten wird nun weiter auf Polymer in der Version 1.0 eingegangen. [Pol16a]

### 5.2.1 Wozu Polymer Elemente

Polymer Elemente versuchen sogenannten Boiler-Code unter Zuhilfenahme von Web Komponenten zu vermeiden. Als Boiler-Code wird die Verschmelzung verschiedener Programmiersprachen innerhalb einer Datei oder eines Elements bezeichnet. Boiler-Code ist häufig in Web-Seiten zu finden, was historisch bis heute bedingt ist. Eine weiter gängiger Name für Boiler-Code im Web ist „DOM Pollution“, was so viel aussagt wie, das der DOM mit verschiedenen Sprachen verschmutzt ist.

Um eine HTML-Element zu erstellen wird als Grundgerüst HTML verwendet. Zusätzlich enthält das HTML-Element aber auch CSS um das aussehen zu beeinflussen. Um ein HTML-Element dynamisch oder interaktiv zu gestalten wird meist zusätzlich auch noch Java-Skript oder AngularJS verwendet. Typischer weise findet mal alle drei Sprachen in einer HTML-Datei oder in getrennten Dateien.

Beide Ansätze haben jedoch Nachteile. Bei der Vereinigung verschiedener Sprachen in einer Datei, leidet meist die Übersicht, denn CSS, Java-Skript und zugehöriges HTML muss nicht an festen Positionen sein. So kommt es öfter vor, dass zusammengehörige Elemente innerhalb einer Datei weit auseinander stehen. Hier leidet die Übersicht und Zusammenhänge sich wenn überhaupt oft nur schwer für fremde Programmierer zu erkennen.

Der Ansatz die verschiedenen Sprachen zu trennen, beseitigt zwar das durcheinander innerhalb der HTML-Dateien, jedoch entstehen hier wieder neue Probleme. Wenn alle Sprachen strikt in verschiedenen Dateien sind, ist es oft nur schwer nachzuvollziehen, welches Java-Skript zu welchem HTML-Element gehört. Für außenstehende Programmierer ist dies oft eine unlösbare

Aufgabe. Nachteilig ist auch, dass immer alle Dateien benötigt werden, um das Gesamtbild verstehen zu können.

### 5.2.2 Was macht Polymer aus

Polymer wiederum geht, im Gegensatz zu den beiden im Abschnitt 5.2.1 genannten, einen dritten sehr eleganten Ansatz. Den der Web Components. Grundsätzlich kommen bei Polymer verschiedene Programmiersprachen und Konzepte wieder zusammen in eine HTML-Datei.

Der entscheidende Unterschied jedoch ist, das HTML, CSS und Skripte ihre festen Bereiche haben, in denen sie implementiert werden. Dies gestaltet die HTML-Datei des jeweiligen Elements sehr übersichtlich und wurde so in den Web Components vom W3C spezifiziert.

Ein weiterer Vorteil ist, dass in der Web-Seite, in der das Element Verwendung findet nur noch ein einzelner Tag zu sehen ist, der auf das Polymer Element verweist.

Damit der Browser das entsprechende Element mit Hilfe der Polymer Library parsen kann muss vor der Verwendung das jeweilige Element über den HTML-Link-Tag eingebunden werden.

Das Polymer Projekt bietet eine ganze Reihe fertige Elemente, welche einfach verwendet werden können<sup>1</sup>.

Wem diese Elemente mit ihren Funktion noch nicht reichen, kann entweder die bestehenden Elemente erweitern oder neue Elemente erstellen.

### 5.2.3 Ein eigenes Polymer Element erstellen

Um zu zeigen, wie einfach es ist auf Basis von Polymer Elemente selbst zu erstellen folgt in den nun kommenden Abschnitten nun Beispiel, welches im Kapitel 6 in Typo3 eingebunden wird.

```
1 <link rel="import" href="../../bower_components/polymer/polymer.html">
2 <dom-module id="button-up">
3   <template>
4     <button id="scrollToTopFAB"
5       class="btn-floating btn-large waves-effect waves-light waves-
6         circle half-out-button tooltipped fix-scroll-button"
7       data-position="top" data-delay="50" data-tooltip="Nach oben
8         Scrollen"
9       on-click="handleClick">
10       <i class="material-icons">navigation</i>
11     </button>
12   </template>
13   <script>
14     Polymer({
15       is: "button-up",
16       handleClick:
17         function (e){
18           $("html, body").animate({scrollTop: 0}, "slow");
19         },
20     });
21   </script>
22 </dom-module>
```

Listing 1: Beispiel für ein eigenes Polymer-Element

Als erstes wird der Polymer-Link hinzugefügt, welcher immer vorhanden sein muss. Danach folgt der Tag „dom-module“, welcher die genaue Beschreibung des Elements enthält. Im „template“-Tag wird das HTML eingefügt, welches das Element beinhaltet. Hierauf kann optional ein „style“-Tag folgen, in welchem der Style des HTML-Elements per CSS angegeben wird.

<sup>1</sup><https://elements.polymer-project.org/>

**CSS für ein Element** Die Definition von CSS hat nur Einfluss auf das jeweilige Template. Elemente außerhalb des Element-DOM werden von dem hier definierten CSS nicht beeinflusst.

Es ist jedoch umgekehrt möglich, CSS-Klassen zu nutzen, im globalen CSS beschrieben sind. Eine erneute Einbindung der CSS-Datei in das Polymer-Element ist nicht notwendig. Die Verwendung von CSS-Abhängigkeiten in Polymer ist jedoch unschön, da dies ein Element gegebenenfalls auf Web-Seiten beschränkt, welche diese bestimmte CSS-Datei verwenden. Es ist somit in den meisten Fällen besser keine CSS-Abhängigkeiten in Elementen zu verwenden, vor allem dann wenn diese öffentlich sind.

Im Listing 1 wurden CSS-Abhängigkeiten zu MaterializeCSS (Abschnitt 5.5.2) aufgebaut, eine Begründung hierfür ist im Kapitel/Abschnitt (????) zu finden.

**Element-Script** Der „script“-Tag enthält das jeweilige Skript zum Element. Es ist JSON-Notation gehalten und basiert auf Callbacks.

**Die Registrierung** eines Elements steht immer zu beginn des Skripts. Dieser Teil muss also immer vorhanden sein, damit das jeweilige Element überhaupt dargestellt werden kann.

```
1 Polymer({  
2   is: "button-up",  
3 });
```

Listing 2: Polymer-Element Registrierung

**Handler** werden in Polymer verwendet um einzelne Skripte aufzurufen. Innerhalb des Handlers kann ein Java-Skript definiert sein, das ausgeführt wird, wenn der jeweilige Handler getriggert wird.

```
1 handleClick:  
2   function (e){  
3     $("html, body").animate({scrollTop: 0}, "slow");  
4   },
```

Listing 3: Polymer-Element Handler

Damit ein Handler überhaupt zum Tragen kommt, muss er innerhalb des HTML-Tags für das dass jeweilige Skript sein soll registriert werden.

```
1 on-click="handleClick"
```

Listing 4: Polymer-Element Handler Registration

**Attribute**, welche das Polymer-Element haben soll, müssen ebenfalls im Skript definiert werden.

```
1 properties: {  
2   owner: {  
3     type: String,  
4     value: "Daniel"  
5   }  
6 }
```

Listing 5: Polymer-Element Attribute

Das Attribut „owner“ wird im Beispiel als String definiert, und hat den Standardwert „Daniel“. Jedes im Skript definiertes Attribut lässt sich bei Verwendung des Elements wie ein normales XML-Attribut verwenden. Wird bei Verwendung des Elements das „owner“-Attribut nicht angegeben, so ist es „Daniel“. Um den Inhalt des Attributs im Template zu verwenden, muss es in doppelten geschweiften Klammern aufgerufen werden. So kann zum Beispiel ein B-Tag wie folgt geschrieben werden.

```
1 <b>{{owner}}</b>
```

Listing 6: Polymer-Element Attribute benutzen

**Der Aufruf** dieses Polymer-Elements muss wie im Beispiel umgesetzt werden

#### 5.2.4 Zusammenfassung

Polymer bietet wie gezeigt wurde eine echte Alternative zum normalen Web-Seiten-Boiler-Code. Durch die Verwendung von Polymer-Elementen entsteht keine nennenswerte Verzögerung beim parsen einer Web-Seite durch den Browser. Es bietet sich daher an, bei Neuentwicklungen auf Polymer zu setzen.

Ob und wie eine Intergration von Polymer in Typo3 möglich ist, wird im Kapitel 6 näher beschrieben.

### 5.3 AngularJS

AngularJS oder kurz Angular ist ein von Google entwickeltes Javascript Framework, welches sich sehr gut für Single-Page-Web-Sites eignet. Angular ist nach dem *Model View ViewModel* (MVVM)-Prinzip aufgebaut, welches eine Trennung zwischen Darstellung und Logik bietet und arbeitet Client-Seitig.

Die Darstellung wird über HTML abgebildet, welches Angular durch eigene Attribute ergänzt. Die Logik wiederum wird durch Javascript auf Basis von Angular abgebildet.

Angular ist in der Lage das HTML-DOM entsprechend der Befehle der Logik umzubauen ohne dafür auf JQuery zurückgreifen zu müssen.

Aufgaben, wie das Suchen oder Sortieren einer Tabelle sind typische Anwendungsbeispiele von Angular. [TB14]

Die momentan aktuelle Version von Angular ist 1.5.6, welche auch in dieser Arbeit verwendet wird. Zur Zeit wird aber auch der Release von Angular 2.0 vorbereitet, wobei die neue Version nicht abwärtskompatibel ist, da sich sehr viel an der Codebasis geändert hat.

Durch das neue Framework, soll das ohnehin schon schnelle Framework noch einmal an Geschwindigkeit zunehmen.

Im folgenden Beispiel der W3C-School wird gezeigt wie Angular eingesetzt werden kann.

```
1 <div ng-app="myApp" ng-controller="myCtrl">
2
3 First Name: <input type="text" ng-model="firstName"><br>
4 Last Name: <input type="text" ng-model="lastName"><br>
5 <br>
6 Full Name: {{firstName + " " + lastName}}
7
8 </div>
```

Listing 7: Angular View im HTML-DOM [W3C16]

Die im Div-Tag zu sehenden Attribute „ng-app“ und „ng-controller“ werden von Angular genutzt. „ng-app“ gibt dabei an, wie die Angular Applikation heißen soll. „ng-controller“ gibt den Namen des Controllers an, welcher für das HTML-Element zum Einsatz kommen soll.

Innerhalb des Div-Tags, welcher die Applikation darstellt, sind zwei Input-Tag zu sehen, welche das Angular-Attribut „ng-model“ beinhalten. „ng-model“ sagt aus, dass diese Elemente zum Datenmodell gehören und welchen Namen sie tragen.



In doppelten geschweiften Klammern werden Modell-Attribute, welche zur Laufzeit von Angular durch die konkreten Werte der Modell-Attribut ersetzt werden.

```
1 <script>
2 var app = angular.module('myApp', []);
3 app.controller('myCtrl', function($scope) {
4     $scope.firstName= "John";
5     $scope.lastName= "Doe";
6 });
7 </script>
```

Listing 8: Angular ViewModel und Model im HTML-DOM [W3C16]

Im Listing 8 ist das ViewModel beziehungsweise das Model des im Listing 7 gezeigten Views zu sehen.

Als erster Schritt, wird der zu referenzierende Angular View angegeben und in der Variable „app“ abgelegt. Daraufhin wird ein Controller innerhalb der Applikation, unter Verwendung des Namens, angelegt.

Innerhalb des Controllers werden dann für die Model-Attribute angegeben. Im Beispiel sind das die Werte „John“ und „Doe“.

Wird nun die entsprechende HTML-Seite im Browser aufgerufen, füllt Angular die beiden Input-Tags mit den Standardwerten und ersetzt auch die Platzhalter in geschweiften Klammern. Ändert ein Nutzer nun den Inhalt der Input-Tags, so passt Angular automatisch das HTML-Dom an und aktualisiert die Werte innerhalb der geschweiften Klammern.

Das vom Browser geparsete Dom enthält keine Angaben darüber, wo sich Platzhalter befinden. Einzig Angular weiß, welche Elemente bei einer Änderung angepasst werden müssen. [W3C16]

## 5.4 HTML5

## 5.5 CSS3

*Cascading Style Sheets* (CSS) ist eine Designsprache für HTML und ist deshalb eine der Hauptkomponenten der Webentwicklung. Mit Hilfe von CSS ist es möglich einzelne HTML-Elemente oder Gruppen zu stylen. Seit der ersten Version die 1993 erschien, wird CSS kontinuierlich weiterentwickelt und ist heute auf praktisch jeder Web-Seite im Einsatz.

Auf der Basis von CSS entwickelten sich im Laufe der Jahre immer mehr Frameworks, welche einen Designansatz umsetzen und fertige Klassen für die Verwendung bereitstellen. Eines der heute am häufigsten anzutreffenden CSS-Frameworks ist „Bootstrap“.

Innerhalb der PDV Systeme Erfurt wurde beschlossen das neue Supportportal im „Material Design“ aufzubauen. Als „Material Design“ werden Gestaltungsrichtlinien von Google bezeichnet, welche angeben wie eine Android-Applikation oder eine mobile Web-Seite für Android aussehen sollte.

Das „Material Design“ ist ein flaches Design, und geht von der Metapher aus, dass der Bildschirm Papier ist. Jedes Element soll sich also wie reales Papier verhalten und auch so aussehen. Obwohl das Design sehr schlicht gehalten ist, so kann mit viel Farbe gearbeitet werden. [Mat16a]

Die nun folgenden Betrachtungen beziehen sich auf die Umsetzung im „Material Design“.

### 5.5.1 Bootstrap

Bootstrap ist ein CSS Framework, welches von „Twitter“ entwickelt wird und das unter der „MIT-Lizenz“ frei erhältlich ist. Durch den modularen Ansatz von Bootstrap ist es sehr leicht möglich das Framework um eigene Style-Anweisungen zu ergänzen. [Boo16] [Wik16c]

Bootstrap baut auf dem Less-Parser auf. Less ist eine Sprache, die es sich zum Ziel gesetzt hat das schreiben von CSS möglichst einfach und effizient zu machen. Geschriebener Less-Code muss in CSS geparkt werden.

Das Bootstrap Framework kann entweder direkt als fertiges CSS-Framework oder als Less-Code heruntergeladen werden.

Ein Nachteil von Bootstrap ist, dass es nicht leichtgewichtig ist. Zwar kann es sehr gut erweitert werden, dies macht jedoch das Framework auch Schwerfällig. Soll zum Beispiel ein Bootstrap-Template für das „Material Design“ verwendet werden, muss zunächst das „standard Framework“ eingebunden werden. Ein Template wie „Material Design for Bootstrap“ überschreibt dann nach der Einbindung zum Teil das standard Framework und ergänzt es um eigene Klassen. Der Vorteil bei diesem Vorgehen ist ganz klar, dass ein bestehendes Template relativ leicht ausgetauscht werden kann, ohne dass der eigentliche HTML-Code der Web-Seite angepasst werden muss.

Dieser Vorteil wird schnell zum Nachteil, wenn eine Web-Seite erstellt werden soll bei der es auf Geschwindigkeit ankommt. Durch das überschreiben des standard Frameworks wird zusätzliche Zeit beim parsen der Web-Seite benötigt, was vorallem bei älteren PCs auffällt. Zusätzlich steigt der Overhead beim laden einer Seite, da mehr CSS geladen wird, als tatsächlich benötigt wird.

Standardmäßig enthält Bootstrap zum Beispiel keinen Date-Picker, welcher jedoch in einem Supportportal unerlässlich ist. Auch „Bootstrap Material“ enthält keinen Date-Picker im „Material Design“. Durch den modularen Aufbau von Bootstrap kann natürlich schnell und einfach ein entsprechendes Template ergänzt werden. Dies bedeutet aber wieder zusätzlichen Overhead. [Mat16b]

### 5.5.2 MaterializeCSS

Ein anderes Framework, welches die Design-Richtlinien des „Material Design“ umsetzt ist das noch recht neue Framework „MaterializeCSS“. MaterializeCSS ist ein eigenständiges Framework, welches versucht alle gegebenen Gestaltungsrichtlinien möglichst elegant und performant umzusetzen.

Die MaterializeCSS Quellen sind mit *Syntactically Awesome Stylesheets* (SASS) geschrieben, und müssen vor der Verwendung in CSS komiliert werden. Dieses Vorgehen hat meherer Vorteile. So ist es zum einen sehr einfach möglich das Framework um eigene Style-Objete zu erweitern. Zum anderen hat es den Vorteil, dass die Farbgebung des gesamten Frameworks an einer Stelle zusammen gefasst ist. Soll also die Farbgebung geändert werden, so werden in den SASS-Quellen des Frameworks die Farben zentral angepasst. Nach einer erneuten kompilierung stehen dann die Farben im gesamten Framework zur Verfügung. Ohne dieses Vorgehen müssten die entsprechenden Farben an unzähligen Stellen innerhalb der CSS-Datei angepasst werden. [Mat16c] Ein weitere Vorteil von MaterializeCSS ist, das viele Templates wie ein Date-Picker im „Material Design“ schon vorhanden sind. Diese Templates können ohne zusätzlichen Overhead verwendet werden.

Auch wenn MaterializeCSS noch sehr jung im Vergleich zu Bootstrap ist, so ist es doch eine gute alternative, wenn eine Web-Seite im „Material Design“ umgesetzt werden soll.

### 5.5.3 Vergleich beider CSS-Frameworks

????? Hier geht es noch weiter

## 5.6 PHP7

## 5.7 Google Dart

Vergleichspunkte	Bootstrap Material	MaterializeCSS
Abhängigkeiten	<ul style="list-style-type: none"> <li>• Bootstrap</li> <li>• JQuery</li> </ul>	<ul style="list-style-type: none"> <li>• JQuery</li> </ul>
Umfang	<ul style="list-style-type: none"> <li>• Standard Layout</li> <li>• Dialoge</li> </ul>	<ul style="list-style-type: none"> <li>• Standard Layout</li> <li>• Dialoge</li> <li>• Date-Picker</li> <li>• Card-Views</li> <li>• Side-Navs</li> <li>• viele weitere</li> </ul>
Farbanpassung	über CSS Klassen (17 Farben)	Farben werden in SASS-Quellcode definiert (kein Limit)
Icons	Verwendung von Material Icons	Verwendung von Material Icons
Subjektive Geschwindigkeit	teilweise Träge und Langsam	immer schnell und flüssig

Tabelle 1: Vergleich von Bootstrap Material und MaterializeCSS

## 6 Zusammenspiel der Technologien

### 6.1 Integration von Polymer in Typo3

Im Abschnitt 5.2 wurde ausführlicher auf die Verwendung und die Erstellung von Polymer-Elementen eingegangen. Nun soll untersucht werden wie sich Polymer möglichst elegant in Typo3 integrieren lässt.

Der beste Weg Polymer-Elemente in Typo3 zu integrieren ist, diese in ViewHelper zu kapseln (Siehe Kapitel ?????).

Das Ziel ist es also, einen ViewHelper zu erstellen, welcher einen ViewHelper-Tag in einen Polymer-Tag umwandelt.

In Abbildung 3 (rechts) ist die Aufrufhierarchie zu sehen, welche innerhalb des Projektes entwickelt wurde, um Polymer Elemente elegant in Typo3 einzuarbeiten.

Die beim laden einer Seite aufgerufene „Action“ eines bestimmten „Controllers“ wird aufgerufen. Der „Controller“ stellt alle zum Rendern des Templates benötigte Daten bereit und übergibt sie dem Template.

Das Template wiederum beinhaltet einen für das Polymer Element entwickelten ViewHelper, welcher innerhalb des Templates aufgerufen wird.

Das Fluid-Framework ruft beim Rendern eines Templates den entsprechenden ViewHelper auf.

Der ViewHelper wiederum gibt den Link zum entsprechenden Polymer Element und den Polymer Tag zurück.

Ist das Template fertig gerendert, so wird es an den aufrufenden Client gesendet.

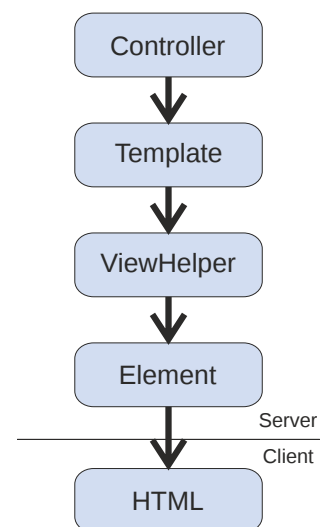


Abbildung 3: Polymer ViewHelper Hierarchie

Der Client wiederum ersetzt das Polymer Element durch HTML.

Am Beispiel des Button-Up Elements (siehe Abschnitt 5.2.3), wird im folgenden erklärt, wie dieser Prozess im Quellcode umgesetzt wird.

```

1 <?php
2 namespace PdvPolymer\ViewHelpers\Custom;
3 use TYPO3\CMS\Fluid\Core\ViewHelper\AbstractTagBasedViewHelper;
4 class ButtonUpViewHelper extends \TYPO3\CMS\Fluid\Core\ViewHelper\
    AbstractTagBasedViewHelper
5 {
6     //the tagname which is also the name
7     protected $tagName = 'button-up';
8     /**
9      * this method is used to initialize the specific tag arguments
10     */
11     public function initializeArguments()
12     {
13         parent::initializeArguments();
14     }
15     /**
16      * generate the tag for the element and the link for the element.
17      * @return string the link and the tag for the element
18     */
19     public function render(){
20         $tag = '<link_rel="import" href="typo3conf/ext/pdvpolymer/Resources/
                Public/Elements/Custom/button-up.html">';
21         return $tag . $this->tag->render();
22     }
23 }

```

Listing 9: Polymer ViewHelper

Der ViewHelper erbt von der abstrakten Klasse „AbstractTagBasedViewHelper“. Dieser ist in der ViewHelper ist anschließend in der Lage, einen HTML-Tag zu verändern.

Mit dem Attribut \$tagname, wird definiert wie der neue Tag heißen soll, welcher den ViewHelper-Tag ersetzt. Dieser Tag muss nun so heißen wie das Polymer-Element.

Die Methode „render“ des ViewHelpers gibt den String zurück, der den ViewHelper-Tag ersetzt.

Die „return“-Anweisung gibt immer den Link zum entsprechenden Polymer-Element zurück, und den Polymer-Tag.

Der ViewHelper wird nun wie folgt aufgerufen.

```

1 <p:custom.buttonUp></p:custom.buttonUp>

```

Listing 10: Polymer ViewHelper aufrufen

Beim ausliefern der Web-Seite ersetzt Typo3 nun den ViewHelper-Tag gegen die folgenden HTML-Elemente

```

1 <link rel="import" href="typo3conf/ext/pdvpolymer/Resources/Public/Elements/
    Custom/button-up.html">
2 <button-up />

```

Listing 11: Polymer ViewHelper Ersetzung

Wird die Web-Seite nun vom Browser gerendert wird, ersetzt er diesen Polymer-Tag schließlich durch das definierte Polymer-Element.

Durch die Verwendung von ViewHelpers, kann jedes beliebige Polymer-Element auch unter Typo3 zum Einsatz kommen. Es ist also sehr elegant möglich Boiler-Code auch in Typo3-Templates und -Partial mit der Hilfe von Polymer zu umgehen.

Durch die Kapselung der Polymer-Elemente in ViewHelpers verhalten sich diese sich außerdem wie standardmäßige ViewHelper mit Boiler-Code.

Um nun Polymer-Elemente in einem Template zu verwenden, ist kein extra Wissen notwendig, was eine Verwendung so einfach wie möglich macht. Der Programmierer eines Templates muss sich keine Gedanken über Links zu Polymer-Elementen machen und muss sich auch nicht mit

Polymer auskennen. Es wird einfach der ViewHelper verwendet und alles andere passiert im Hintergrund.

## 6.2 MaterializeCSS

Die Integration von MaterializeCSS in Typo3 ist genau so einfach, wie die Integration von Bootstrap oder jedem anderem CSS-Frameworks. Um MaterializeCSS in Typo3 verwenden zu können, muss im Template nur angegeben werden, welche CSS-Dateien Typo3 einbinden soll.

Zusätzlich zu der CSS-Datei benötigt MaterializeCSS noch eine eigene Java-Script-Datei und eine aktuelle Version von JQuery. Diese Dateien werden zusammen mit dem CSS im Template definiert.

Außerdem ist es möglich mit MaterialieCSS die standard Icons und Schriftarten von Google zu nutzen. Sollen dieses Sachen verwendet werden, so müssen sie nur zusätzlich zum CSS und Java-Script im Template definiert werden.

Im Listing 12 ist ein Typo-Script-Beispiel zu sehen, mit welchem MaterializeCSS in Typo3 integriert wird. Das angegebene Typo-Script an entweder in der Konfiguration einer Extension angegeben sein oder es wird direkt in einem Typo3-Template definiert.

```
1 page{
2     includeCSS{
3         icons = https://fonts.googleapis.com/icon?family=Material+Icons
4         material = EXT:pdvtemplate/Resources/Public/Materialize/css/materialize.
            css
5     }
6     includeJS{
7         query = EXT:pdvtemplate/Resources/Public/jquery-2.1.1.min.js
8         materialjs = EXT:pdvtemplate/Resources/Public/Materialize/js/materialize
            .js
9         webcomponents = EXT:pdvtemplate/Resources/Public/webcomponents-lite.js
10    }
11 }
```

Listing 12: MaterialieCSS Anbindung

## 6.3 Nutzer Anmeldung und Registrierung

## 7 Zusammenfassung

## 8 Fazit

## 9 Abkürzungsverzeichnis

**CMS** *Content Management System*

**LTS** *Long Term Support*

**MVC** *Model View Controller*

**WYSIWYG** *What you see is what what you get*

**CSS** *Cascading Style Sheets*

**SASS** *Syntactically Awesome Stylesheets*

**MVVM** *Model View ViewModel*

## Abbildungsverzeichnis

1	Typo3 Backend im Seitenbearbeitungs Modus [Tea16] . . . . .	7
2	Typo3 Neos Backend im Seitenbearbeitungs Modus [HDB14] . . . . .	8
3	Polymer ViewHelper Hierarchie . . . . .	19



**Tabellenverzeichnis**

1      Vergleich von Bootstrap Material und MaterializeCSS . . . . . 19

## Literatur

- [Boo16] Bootstrap Homepage. <http://getbootstrap.com/>, September 2016.
- [CG13] Dominic Cooney and Dimitri Glazkov. Introduction to Web Components. June 2013.
- [HDB14] HDBlog TYPO3 Neos: Das CMS der Zukunft? <https://www.hdnet.de/blog/typo3-neos-das-cms-der-zukunft/>, Januar 2014.
- [Ihl13] Jens Ihlenfeld. HTML-Elemente selber bauen. <http://www.golem.de/news/web-components-html-elemente-selber-bauen-1305-99318.html>, May 2013.
- [Lob16] Parick Lobacher. *TYPO3 Extbase: Moderne Extensionentwicklung für TYPO3 CMS mit Extbase & Fluid*. CreateSpace Independent Publishing Platform, 2 edition, Februar 2016.
- [Mat16a] Material Design. Technical report, Google Inc., September 2016.
- [Mat16b] Material Design for Bootstrap. Technical report, September 2016.
- [Mat16c] Materialize. Technical report, September 2016.
- [Pol16a] Polymer Project. <https://www.polymer-project.org/1.0/>, November 2016.
- [Pol16b] What is shady DOM? <https://www.polymer-project.org/1.0/blog/shadydom>, November 2016.
- [Rie15] Sebastian Rieger. Konzept und prototypische Implementierung eines übergreifenden Dokumenten- und Medienmanagements. Technical report, Karlsruher Institut für Technologie, August 2015.
- [TB14] Philipp Tarasiewicz and Robin Böhm. *AngularJS : eine praktische Einführung in das JavaScript-Framework*. dpunkt-Verl., Heidelberg, 2014.
- [Tea16] TYPO3 News. <https://typo3.org/news/article/typo3-v81-tightening-the-screws/>, September 2016.
- [W3C16] AngularJS Introduction. [http://www.w3schools.com/angular/angular\\_intro.asp](http://www.w3schools.com/angular/angular_intro.asp), November 2016.
- [Wik16a] TYPO3. <https://de.wikipedia.org/wiki/TYP03>, September 2016.
- [Wik16b] TYPO3 Flow. [https://de.wikipedia.org/wiki/TYP03\\_Flow](https://de.wikipedia.org/wiki/TYP03_Flow), September 2016.

[Wik16c] Bootstrap (Framework).  
(Framework), September 2016.

[https://de.wikipedia.org/wiki/Bootstrap\\_](https://de.wikipedia.org/wiki/Bootstrap_)