Interaktive Visualisierung von Querbeziehungen in XML-Daten mit D3.js

Bachelorarbeit

In der Studienrichtung „Technische Redaktion“
im Studiengang
„Technische Redaktion und E-Learning-Systeme“

an der
Hochschule Merseburg

Vorgelegt von
Alexander Jache
Kurt-Eisner-Straße 71
04275 Leipzig

Eingereicht bei
Erstbetreuer: Dr. rer. nat. Thomas Meinike
Zweitbetreuer: Dipl.-Phys. Gerrit Imsieke

Merseburg, März 2016

Vorwort

Zuerst möchte ich mich hiermit bei all denjenigen bedanken, die mich während der Anfertigung dieser Bachelor-Arbeit unterstützt und motiviert haben.

Ganz besonderer Dank gilt Dr. rer. nat. Thomas Meinike, welcher mir mittels seines stets begeisterungsfähigen Lehrstils die Einführung in dieses Wissensgebiet ermöglicht und mein Interesse dafür geweckt hat.

Nicht weniger Dank gilt Dipl.-Phys. Gerrit Imsieke, dem ich viele Praxis-Projekte aus dem Alltag von le-tex zu verdanken habe und dadurch mein theoretisches und praktisches Wissen deutlich erweitern konnte.

Zu guter Letzt danke ich meinem geschätzten Kollegen Philipp Glatza, ohne dessen Hilfe ich für die Lösung mancher Problemstellungen deutlich länger gebraucht hätte. Er stand mir während der Arbeit als stets hilfreicher und kritischer Supervisor zur Seite.

Inhalt

1 Einleitung

1.1 Unternehmen

Die Firma le-tex publishing services GmbH wurde 1999 gegründet und hat sich auf verschiedene Dienstleistungen des Verlagswesens spezialisiert. Unter den Begriff „Content Engineering“[1] fallen beispielsweise Dienstleistungen aus dem Bereich Satz, Verlagsherstellung, Copy-Editing oder „Single Source Publishing“[2].

le-tex versteht sich dabei als Dienstleister. Es zählt zur Philosophie des Unternehmens, freie Technologien und Werkzeuge zu verwenden und zu fördern. Eigene entwickelte Werkzeuge werden meist unter freier BSD-Lizenz zur Verfügung gestellt. Um diesen freien Open Source Gedanken aufzugreifen, liegen der Realisierung dieser Arbeit auch freie Technologien zugrunde.

Jährlich ist le-tex auf der Frankfurter Buchmesse mit einem Messestand präsent, um neue Kunden zu akquirieren. Neben persönlicher Beratung und Vorträgen zu aktuellen Projekten und Produkten, gibt es für Interessenten auch die Möglichkeit die Website von le-tex auf einem Touchscreen zu besuchen.

Mit dem Label „XML made in Germany“ wurde von le-tex und zwei weiteren führenden XML-Verlagsdienstleistern eine Initiative ins Leben gerufen, um „[…] die Zukunft des Publizierens aktiv mitzugestalten […]“[3]. In diesen Zusammenhang wurde ein gemeinsames Programm mit Fachvorträgen für das Crossmediale Publizieren aufgesetzt.[4]

1.2 Aufgabenstellung

Das Design der Unternehmens-Website wurde von Marcel Langer[5] für le-tex entwickelt und stellt eine Papieranalogie mit stilisierten Buchrücken in der oberen linken Ecke dar. Die Website wurde im Rahmen eines Praktikums durch den Autor, mittels Bootstrap-Framework, auf ein responsives Layout umgebaut. Neben dem responsiven Layout war eine weitgehende Funktionalität ohne JavaScript gewünscht.

Da das Unternehmen jährlich mit einem Messestand auf der Frankfurter Buchmesse vertreten ist, wurde vor einiger Zeit bereits einmal eine auf JavaScript basierende interaktive Navigation für den Messestand erstellt. Diese erwies sich aber beim Benutzertest als sehr ressourcenhungrig und nicht intuitiv.

Um dies Problem zu beseitigen wurde beschlossen eine neue Navigation zu erstellen, die auf D3.js basiert, um Probleme mit der Performance, als auch die Benutzerfreundlichkeit der Benutzerschnittstelle zu verbessern. Gleichzeitig soll die interaktive Navigation als optionales Feature Benutzern der regulären Website zur Verfügung stehen. Da die Website[6] des Unternehmens bisher keine Suchen-Funktion enthält, soll in diesem Zusammenhang auch eine Schlagwort-Suche implementiert werden. Weiterhin gilt es, die Website aufgrund von Barrierefreiheit bei deaktivierten JavaScript benutzbar zu halten. Um die Benutzerfreundlichkeit noch weiter zu verbessern, soll der Browserverlauf auch bei Verwendung der interaktiven Navigation „gefüttert“ werden.

---

[1] Daher sind die meisten Studiengänge in diesem Bereich auch zertifizierte Ingenieurs-Studiengänge.
[2] Aus einer XML-basierende Datenquelle, werden mehrere Ausgabeformate (automatisiert) erstellt.
[3] (le-tex publishing services GmbH, 2015).
[4] Das Programm von der Buchmesse 2015 ist zu finden unter (XML made in Germany, 2015).
[5] Stiftdesign Marcel Langer (Langer, 2014).
[6] (le-tex publishing services GmbH, 2016).

Als Datenquelle für die Website von le-tex dient ein XML-Dokument nach DocBook-Standard[7], aus dem die einzelnen Webseiten mittels Saxon PE 9.4 und anhand eines einzelnen XSLT-Dokumentes transformiert werden. Es existieren dabei zwei DocBook-Dokumente, jeweils eins für die deutsche Sprache und eins für die englische Übersetzung. Vorteil dieser technischen Umsetzung ist es, dass man auf den Betrieb eines CMS verzichten kann. Als Versionskontrolle kommt Subversion (svn) zum Einsatz.

Die Arbeit setzt sich im theoretischen Teil mit einer Reihe von XML- und Web-Technologien auseinander. Da jede einzelne Technologie problemlos eine Arbeit füllen könnte, erhebt die Arbeit keinen Anspruch auf Vollständigkeit. Stattdessen werden einige Schwerpunkt-Technologien, auf denen sich der praktische Teil der Arbeit besonders stützt, ausführlicher behandelt (z. B. SVG oder D3.js). Alle anderen Technologien werden bestenfalls angerissen. Ebenfalls Teil der Arbeit ist eine Untersuchung, wie die Querbeziehungen sinnvoll hergestellt werden können und welche Datenstrukturen sich für das Konzept eignen.

Wenn in dieser Arbeit von Websites geschrieben wird, so sind stets komplette Internetauftritte mit zahlreichen Webseiten, engl. Webpages, gemeint. Somit soll einer Missdeutung der Begriffe Webseite, Website oder Webpage, die oft als Synonym verstanden werden.

Ein weiterer zu definierender Term ist der Name des Produktes, also der interaktiven Navigation, die mit dieser Arbeit entstanden ist. Vom Autor wurde der Begriff „Navigator“ gewählt, was sich in der Arbeit als Begriff überwiegend wiederspiegelt und auch in die Nomenklatur der Dateinamen einfloss. Da die Vorgängeranwendung aufgrund ihres Fokus für den Messeauftritt als Kiosk bezeichnet wurde, können in der Umsetzung Klassen und Variablen nach der alten Bezeichnung umgesetzt sein. Das wurde so gewählt, da im XSLT bereits Funktionen und Variablen mit dieser Bezeichnung existierten und nicht neu benannt werden sollen.

[7] In DocBook Version 5.0.

2 Theoretische Vorbetrachtung

Ausgehend vom Titel dieser Arbeit verweist sie auf den Aspekt der Querbeziehungen. Es stellt sich die Frage, was Querbeziehung in einer Navigation sinnvoll und interessant erscheinen lassen könnte. Dazu soll eine Betrachtung klassischer Navigationsstrukturen von Websites helfen.

2.1 Hierarchische Menüstruktur

Die meisten Websites, Onepager[8] ausgenommen, bestehen aus einer mehr oder weniger großen Anzahl einzelner Webseiten. Verlinkt sind diese Seiten durch ein Menü in einem oberen oder seitlichen Bereich der Website. Es existieren meist 1-3 Menü-Ebenen. Die Unterseiten sind immer stets einer übergeordneten Seite zugeordnet. Die Hierarchie die sich daraus ergibt, entspricht einer Baumstruktur.

Auch die Website von le-tex entspricht überwiegend dieser Hierarchie. Diese Form der Ordnung hat sich für viele Websites bewährt und stellt kein Problem dar, solange die Hierarchie einer Webseite mit ihren Inhalten sich in diese Hierarchie einbringen lässt. Neue Webseiten müssen sich immer in ein bestehendes System einpassen. Sollte das nicht möglich sein, greift man gern zur Umbenennung einer Eltern-Kategorie in eine allgemeinere Bezeichnung. Die Zahl der Elemente einer Ebene sollte dabei die Millersche Zahl[9] nicht überschreiten.

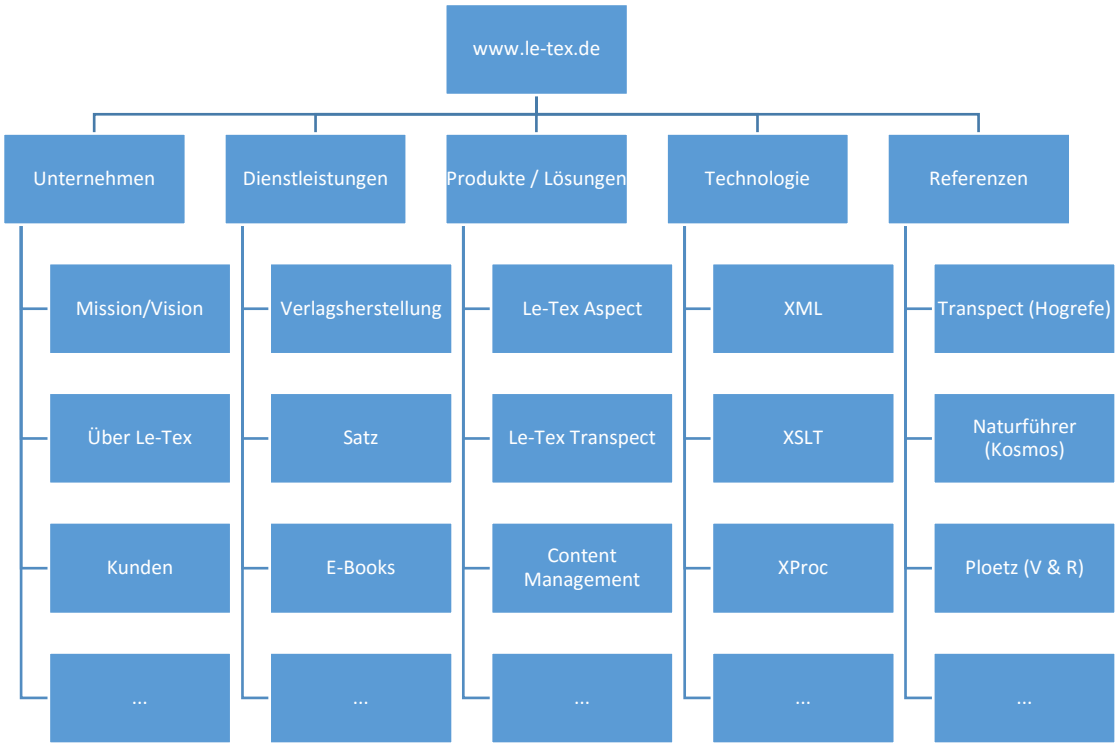


Abbildung 1: Hierarchie von www.le-tex.de (Auszug)

Viele Websites besitzen noch zahlreiche Seiten, die nicht direkt in der Menüstruktur erfasst sind, aber für den Nutzer nicht weniger interessant oder wichtig sind. Typische Beispiele sind die Seiten für

[8] Webauftritte bestehen aus einer einzelnen Webseite. Die Inhalte sind untereinander, meist gestalterisch getrennt, angeordnet. Die Navigation erfolgt durch Anker-Links mittels eines Menüs oder Affix. Beliebt sind diese Single-Page Seiten aufgrund einer leichteren responsiven Umsetzbarkeit und der Dynamik einzelner Elemente (Stichwort Storytelling).

[9] Die Zahl 7 wurde von George A. Miller festgelegt als durchschnittliche Kapazität von sogenannten Chunks, die ein Kurzzeitgedächtnis speichern kann.

das Impressum oder den Kontakt. Die Verlinkungen hierfür befinden sich in der Regel im Footer[10]. Bei der Website von le-tex existieren aber z. B. mehrere Unterseiten für Ansprechpartner, die sich mit der klassischen Footer-Struktur schwierig abbilden lassen. Oft wird das Problem damit gelöst, dass entweder das Hauptmenü einen zentralen Eintrag „Kontakte“ erhält, oder die Footer-Unterseiten sind nicht mehr durch Menü erfasst. Dadurch besteht das Risiko, dass der User die Unterseite nicht findet und die Website vorzeitig verlässt.

2.2 Alternative Navigationsmodelle

Nun stellt sich die Frage, welche Möglichkeiten existieren, um den Nutzer zielgerichtet an die gesuchten Informationen zu leiten. Eine der klassischen Möglichkeiten ist die Integration einer einfachen Stichwort-Suche. Bei großen Internetauftritten kann die Zahl der Treffer aber schnell unübersichtlich groß werden, weshalb meist noch die Treffer nach bestimmten Kriterien[11] einer Relevanz zugeordnet werden.

Eine weitere Möglichkeit den Nutzer schneller zur gewünschten Information zu leiten, ist die Tag Cloud, in deutscher Sprache meist „Schlagwortwolke“ genannt. Meist alphabetisch sortiert, werden Schlagworte in einer Art Wolke dargestellt und je nach Gewichtung werden einzelne Terme unterschiedlich dargestellt. Das heißt häufig aufgerufene oder oft kategorisierte Schlagworte[12] werden größer oder auffälliger dargestellt als seltener aufgerufene Schlagworte.

Abbildung 2: Tag-Cloud-Beispiel aus der Unternehmensbroschüre von le-tex

Empirische Usability-Tests haben ergeben, dass Schlagworte besondere Aufmerksamkeit erlangen, wenn sie auffälliger dargestellt werden und bzw. oder sich in der Mitte oder im oberen linken Bereich

[10] Während der Header in der Webentwicklung den Kopfbereich einer Webseite darstellt, stellt der Footer den Fußbereich einer Webseite dar.
[11] Wenn der gesuchte Term beispielsweise in einer Überschrift H1 zu finden ist, wird der Treffer stärker gewichtet, als bei einem Inhaltstext.
[12] Bei Weblogs fließt die Zahl der unter einer Kategorie zugeordneten Einträge in die Wertung ein.

der Tag Cloud befinden.[13] Eine Tag Cloud kann aber aufgrund ihrer statistischen Grundlage nicht den individuellen Bedürfnissen eines Nutzers gerecht werden. Sie spiegelt entweder die Schlagworte wider, für die die meisten Inhalte vorhanden sind, oder die am häufigsten gesuchten Schlagworte. Untersuchungen haben ergeben, dass vorzugsweise erfahrene Nutzer am besten mit Tag Clouds navigieren können.[14] Ein weiterer Nachteil besteht darin, dass mit zunehmender Anzahl an Schlagworten die Übersicht verloren geht und eine Abbildung umfangreicher Websites gar nicht mehr übersichtlich möglich ist.

2.3 Querbeziehungen

Um zu klären, was man unter Querbeziehung versteht, muss man sie von den hierarchischen Beziehungen abgrenzen. Hierarchische Beziehungen sind die einzelnen Links zwischen einem Eltern-Knoten[15] und einem Kind-Knoten. Neben diesen offensichtlichen Verbindungen gibt es auch Verbindungen, die zwischen Kind-Knoten unterschiedlicher Zweige bestehen können. Auch kann ein Knoten in der 4. Hierarchieebene in direktem Bezug zu einem Knoten in der 2. Ebene stehen. Diese Verbindungen werden durch eine hierarchische Baumstruktur nicht abgebildet.

Bereits zu Beginn der Arbeit enthielt die Website von le-tex Querverweise zu verwandten Themen, welche sich als Navigationsinstrument nutzen lassen.

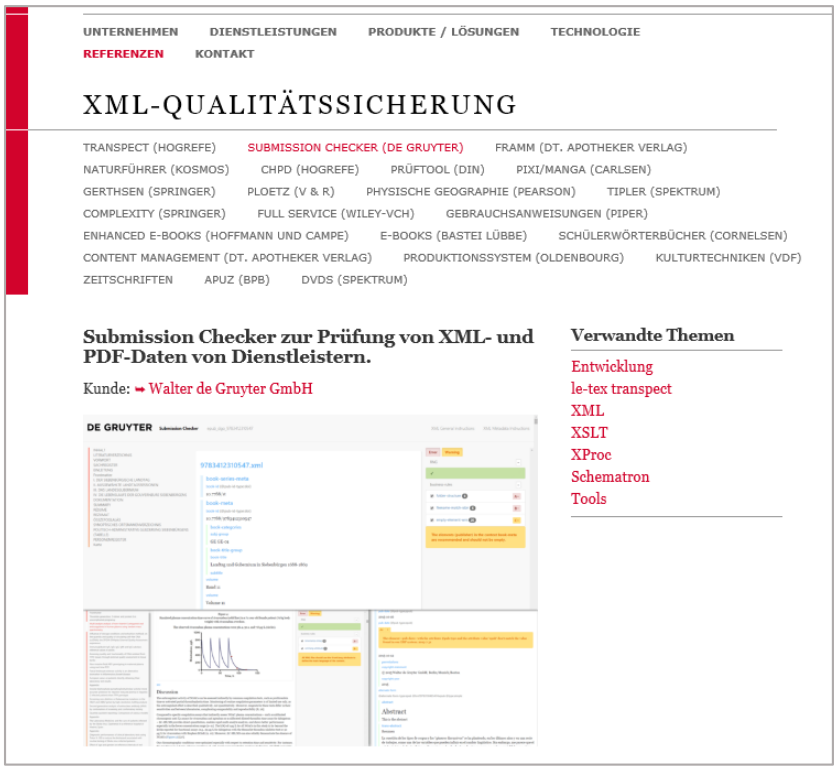


Abbildung 3: Screenshot einer Kunden-Referenz mit Verwandten Themen

Es stellt sich die Frage, was die Querbeziehungen als Navigationsinstrument interessant erscheinen lässt. Betrachtet man das Beispiel der Firma le-tex, so ist man zwangsweise mit einer Vielzahl an Dienstleistungen und Produkten konfrontiert, die den Besucher der Website nur teilweise, womöglich gar nicht vertraut sind, aber in Bezug zueinanderstehen können. Durch eine Visualisierung

[13] (Lohmann, et al., 2009, p. 12).
[14] (Oelke & Gurevych, 2014, pp. 7 - 8).
[15] Ein Knoten (englisch: node) ist ein Knotenpunkt in einer Baumstruktur. Bei einer Hierarchie einer Website kann er eine logische Unterteilung oder eine zugrundeliegende Webseite sein.

von Querbeziehungen hat der Nutzer die Möglichkeit die sonst verborgenen Beziehungen als Navigationsinstrument zu verwenden. Websites werden damit nicht mehr alleine als hierarchisches Konstrukt aus Einzelseiten betrachtet, sondern als Gesamtkonstrukt.

Fragt man sich, warum der Nutzer von den Querbeziehungen profitieren kann, muss man sich mit den Gründen auseinandersetzen, die den Nutzer zur Website geführt haben. Um diese Frage zu beantworten, liegt jetzt keine qualitative oder quantitative Evaluation zugrunde, sondern sollen nur schemenhaft einige Nutzerprofile angerissen werden.

Neben der Laufkundschaft am Messestand wird ein Teil der Nutzer durch Mundpropaganda oder Empfehlungen von Kunden auf le-tex aufmerksam geworden sein.[16] Somit findet er sich eventuell schnell bei den Referenzen von le-tex wieder. Ist der Nutzer nun (teilweise) fachfremd, wird er unvermeidlich mit zugrundeliegenden Produkten bzw. Technologien konfrontiert werden. Um den Informationsbedarf unmittelbar befriedigen zu können, wird durch Links auf die entsprechenden Seiten der unmittelbare Navigationspfad vorgegeben. Dem Nutzer wird also während seines Aufenthaltes ein „Roter Faden“ angeboten, mit dem er schneller zum avisierten Ziel kommt, als er durch Navigation mittels hierarchischer Menüstruktur kommen würde. Wenn dann der Besucher am Ende des Fadens noch auf der Kontaktseite eines Mitarbeiters für das entsprechende Produkt oder Dienstleistung landet und Kontakt aufnimmt, haben beide Seiten davon profitiert.

Diese Form der „Rote-Faden-Navigation“ ist besonders für Erstinteressenten eine hilfreiche Navigationsform. Besonders viele Erstkontakte und Interessenten finden sich gewöhnlich auf Messen. Genau dieser Zielgruppe soll mit der Interaktiven Navigation am Messestand begegnet werden. Dazu ist zweifellos auch ein gewisses Maß an Eyecatcher-Qualität notwendig, dem die Arbeit auch gerecht zu werden versucht.

Spricht man von Querbeziehungen stellt sich die Frage, wie sich diese erzeugen lassen. Einen Teil der Beziehungen ergeben sich aus dem Eltern-Kind-Verhältnis der Hierarchie. Einen weiteren Teil der Beziehungen kann man durch Verweise auf Ansprechpartner erschließen. Man kommt aber nicht um einen fähigen Redakteur herum, welcher die Querbeziehungen bereits im Quell-Dokument manuell herstellt.

[16] Tatsächlich akquiriert le-tex einen Großteil der Kunden durch B2B-Empfehlungen.

3 Technologien

3.1 XML

Wie der Name Extensible Markup Language bereits assoziiert, handelt es sich bei XML um eine erweiterbare Auszeichnungssprache. Anders als bei HTML, wo eine feste Anzahl an Elementen existiert, können bei XML beliebig viele Elemente selbst definiert werden. In Kombination mit Schemasprachen wie DTD oder Schematron, lassen sich Struktur und Inhalte anwendungsspezifisch einschränken und validieren. Erweitern lassen sich Elemente mittels Attributen.

XML basiert auf der Standard-Meta-Sprache SGML von 1986[17]. Während die erste Spezifikation von XML 1998 erschien[18], stammt die aktuelle Version vom 26. November 2008[19].

Grundeigenschaften:

- Es gibt genau ein Wurzelelement.
- Struktur wird durch Verschachtelung von Elementen abgebildet.
- Elemente bestehen aus Start- und End-Tags bzw. leere Element-Tags.
- Elemente können durch optionale Attribute erweitert werden.

In der XML-Spezifikation wird der Begriff Wohlgeformtheit definiert: „Ein Dokument ist dann wohlgeformt, wenn es den grundlegenden Regeln von XML […] entspricht.“ [20] Wohlgeformtheit ist aber meist unzureichend. Meist sollen Dokumente validierbar, also überprüfbar sein. Das erfordert die Verwendung von Schemasprachen wie DTD oder Schematron. Fehlt z. B. ein Pflichtattribut zu einem Element, kann das Dokument wohlgeformt sein, aber es ist aufgrund der Definition in der DTD nicht valide.

3.1.1 Prolog

Auch wenn ein Prolog optional ist, kommt praktisch kaum ein XML-Dokument ohne vor. Im Prolog ist die XML- und DOCTYPE-Deklaration enthalten. Die XML-Deklaration sorgt dafür, dass eine XML-Datei als solche erkannt wird. Sie beginnt mit `<?xml` und endet mit `?>`. Dazwischen steht eine Reihe von Parameter-Werte-Paaren:

- *version* – Gibt die Version der zugrundeliegenden XML-Spezifikation an. Bisher gibt es nur den Wert 1.0.
- *encoding* – Legt die Zeichencodierung des Dokumentes fest. Ist nichts angegeben, wird UTF-8 angenommen.
- *standalone* – Gibt an, ob die DTD extern oder intern (im Dokument) enthalten ist. Standardwert ist `no`, also es wird von einer externen DTD ausgegangen.

Ein Beispiel mit allen Parametern:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
```

Listing 1: XML Prolog

3.1.2 Elemente

Ein Element besteht, mit Ausnahme von leeren Elementen, aus einem Start-Tag und einem End-Tag. Die Tags wiederrum beginnen mit dem Zeichen < und enden mit dem Zeichen >. Zwischen den beiden Zeichen befindet sich der Elementname, sowohl im Start-, als auch im End-Tag. Ein End-Tag

[17] ISO 8879:1986.
[18] XML 1.0 First Edition (W3C, 1998).
[19] XML 1.0 Fifth Edition (W3C, 2008).
[20] (Schraitle, 2004, p. 34).

wird noch durch ein / (engl. Slash) ergänzt, welcher nach dem Zeichen < folgt. Eine Ausnahme sind leere Elemente. Diese haben kein End-Tag. Das Start-Tag wird dann sofort durch ein / vorm Zeichen > geschlossen. Zwischen den Start- und End-Tag befindet sich der Inhalt.

Beispiele:

- Element mit Inhalt: <artist>David Bowie</artist>
- Leeres Element:

Unabhängig von der verwendeten DTD oder dem Schema darf die Benennung von Elementen nicht beliebig erfolgen. Grundsätzlich sind beliebige alphanummerische[21] Zeichen erlaubt. Sogar nicht-lateinische Buchstaben und Ideogramme[22] sind erlaubt. Als Interpunktionszeichen darf auch Unterstrich, Bindestrich und Punkt verwendet werden. Dennoch gibt es Einschränkungen. So ist das Schlüsselwort „xml“ reserviert und auch dürfen Leerzeichen nicht vorkommen. Elementnamen dürfen nicht mit Ziffern beginnen.

3.1.3 Attribute

Elemente können zusätzlich noch Attribute besitzen. Attribute befinden sich immer im Start-Tag bzw. im leeren Element-Tag. Ein Attribut besteht aus einem Attributnamen und einem Wert. „Attributwerte müssen in einfachen (') oder doppelten (") Anführungszeichen eingeschlossen werden […]“[23]. Durch ein Gleichheitszeichen erfolgt die Zuweisung. Die Trennung mehrerer Attribute in einem Element erfolgt durch mindestens einen Lehrraum[24].

Für Attribute gelten dieselben Regeln bezüglich der Benennung, wie sie auch für Elemente gültig sind. Pro Element darf aber ein Attributname nur einmal verwendet werden. Der wesentliche Unterschied zwischen einem Attribut und einem Element ist, dass ein Attribut keine Elemente enthalten darf.

Beispiel:

```
<artist
  facebook-url="https://www.facebook.com/OfficialLemmy">
  Lemmy Kilmister
</artist>
```

Listing 2: XML-Element mit einem Attribut

3.1.4 Kommentare

Wie jede andere Programmier- oder Auszeichnungssprache auch, sind in XML Kommentare möglich. Kommentare lassen sich zwischen den Zeichenfolgen <!-- und --> einfügen. Diese dürfen sich über mehrere Zeilen erstrecken, solange im Kommentar keine zwei Bindestriche -- vorhanden sind. Kommentare dürfen nicht in Tags oder vor XML-Deklarationen stehen. Inkludierte Codeteile werden ebenfalls „auskommentiert“.

Beispiel:

```
<!-- Kondolenzeintrag für <artist>Achim Menzel</artist> -->
```

Listing 3: XML-Kommentar

[21] Klein- und Großbuchstaben von A bis Z, sowie die Ziffern 0 bis 9.
[22] Asiatische Schriftzeichen.
[23] (Schraitle, 2004, p. 15).
[24] Als Leerraum zählen (HEX-ASCII-Code in Klammern) Leerzeichen (0x20), Tabulator (0x09), Zeilenvorschub (0x0A) oder Wagenrücklauf (0x0D).

3.1.5 Namespaces

Ebenfalls, wie in vielen Programmiersprachen und Markup-Sprachen auch, gibt es in XML auch Namensräume (engl. Namespaces). Sobald mehr als eine Markup-Sprache mit unterschiedlichen Vokabular in einer Datei verwendet werden soll, empfiehlt sich die Verwendung von Namespaces. Mittels eines Präfixes vor dem Element- bzw. Attributnamen, gefolgt von einem Doppelpunkt als Trenner, wird der Namespace zugeordnet. Das Präfix vor dem Doppelpunkt wird als „Namensraumpräfix" bezeichnet, der Teil dahinter „lokaler Name". Zusammengenommen bezeichnet man das Konstrukt als qualifizierten Namen[25].

Beispiele:

- `xsl:apply-templates`
- `svg:rect`
- `xs:string`

Die Verwendung des Präfixes alleine reicht dabei nicht, da es nur eine Art Platzhalter ist. Um den Namensraum zu binden, benötigt man das reservierte Attribut `xmlns` (XML Namespace). Folgend ein Beispiel für die Namensraumdeklaration für xLink[26]:

```
<svg height="30" width="200" xmlns:xlink="http://www.w3.org/1999/xlink">
  <a xlink:href="https://www.youtube.com/watch?v=A5CV4NLIJyM">
    <text x="0" y="15" fill="red">The Transformation Song</text>
  </a>
</svg>
```

Listing 4: Bindung des xLink-Namensraumes an den Präfix „xlink"

Deklariert man den Namensraum ohne Präfix, wird er als Standardnamensraum festgelegt, womit das Präfix bei den Kindelementen wegfallen kann. Dieser Standardnamensraum ist solange gültig, bis er an eine andere URI gebunden wird.

3.1.6 Validierung

Immer wenn XML-Daten nicht nur auf Wohlgeformtheit geprüft, sondern auch validiert werden sollen, benötigt man eine DTD oder eine andere Schemasprache. Obwohl die DTD-Syntax selbst kein XML ist, ist sie Teil der XML-Spezifikation. Als XML-basierende Schemasprachen sollen hier das XML-Schema (XSD), Schematron und Relax NG erwähnt werden. Über Vor- und Nachteile dieser parallel gepflegten Technologien soll hier nicht debattiert werden, da lediglich auf fertig erstellte Schemata des DocBook-Standards zurückgegriffen wird. Es soll nur so viel erwähnt werden, dass alle drei XML-basierenden Möglichkeiten die von DTD deutlich übersteigen[27].

3.2 DocBook

DocBook ist ein lizenzfreier, offener OASIS[28]-Standard und wurde ursprünglich für SGML implementiert. Seit der Version 4.0 existiert auch eine Implementierung für XML, welche dann seit Version 5.0 wieder mit der Implementierung von SGML zusammengeführt wurde. Für DocBook gibt es neben der DTD auch Implementierungen für XML-Schema, Schematron und RELAX NG.

[25] Abgekürzt als QName.
[26] Ähnlich Hyperlinks, ermöglicht aber bidirektionale und sogar multidirektionale Verbindungen (W3C, 2010).
[27] Eine kurzes Einführungs- und Vergleichsvideo gibt es unter (video2brain, 2016).
[28] Nicht mit der britischen Popgruppe zu verwechseln, siehe Abkürzungsverzeichnis!

DocBook Prolog mit dem Element <book> als Wurzelelement:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model
  href="http://docbook.org/xml/5.0/rng/docbook.rng"
  schematypens="http://relaxng.org/ns/structure/1.0"?>
<?xml-model
  href="http://docbook.org/xml/5.0/rng/docbook.rng"
  type="application/xml"
  schematypens="http://purl.oclc.org/dsdl/schematron"?>
<book
  xmlns="http://docbook.org/ns/docbook"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  version="5.0">
</book>
```

Listing 5: DocBook V5 Prolog und Wurzelelement

Die Quelldateien für die Website von le-tex wird ebenfalls nach dem DocBook-Dokumenttyp „book“ gepflegt.

3.3 XSL

Extensible Stylesheet Language ist eine Familie von Transformationssprachen, die auf DSSSL zurückgeht, aber auf XML basiert. Zur Familie von XSL gehört XSLT, XPath als auch XSL-FO, eine Stylesheet Sprache vorwiegend für Druckerzeugnisse, die hier nicht näher erläutert werden soll. Hier wird der zweite Unterschied zu DSSSL deutlich. Die einzelnen Sprachen der XSL-Familie sind unabhängig voneinander.

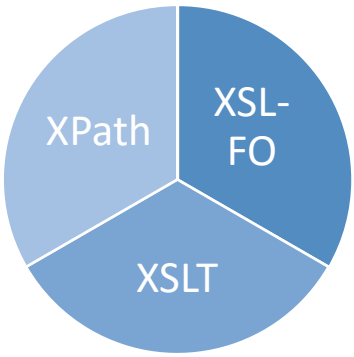


Abbildung 4: XSL-Technologien

3.3.1 XSLT

XSL Transformation, meist XSLT abgekürzt, ist eine Programmiersprache zur Transformation von XML-Dokumenten. Das resultierende Dokument kann dabei wieder ein Dokument sein, welches der XML-Syntax entspricht oder aber ein anderes Format aufweist[29]. Um ein XML-Dokument in ein anderes XML-Dokument umzuwandeln, benötigt man neben der XSLT auch einen Prozessor. Zwei bekannte Prozessoren sind Saxon[30], den es auch als JavaScript-basierte Versionen gibt[31] und Antenna House[32].

[29] Zum Beispiel eine Textdatei oder Binärdatei.
[30] Entwickelt durch die Firma Saxonica, des W3C-Mitglieds Michael Kay.
[31] Saxon CE und im Laufe des Jahres 2016 auch das überarbeite SaxonJS mit XPath3-Unterstützung.
[32] Zu finden auf www.antennahouse.com.

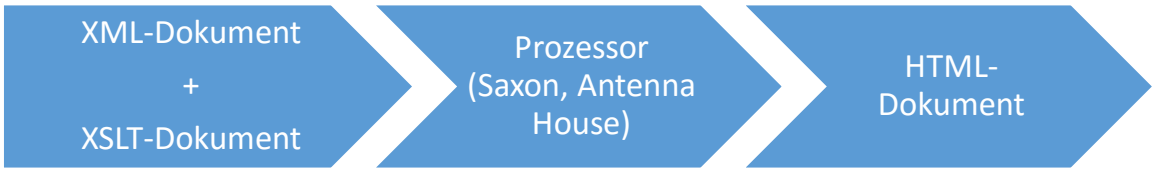


Abbildung 5: XSLT-Transformation von XML zu HTML

Die Transformation erfolgt anhand einzelner Templates, welche aus einem Pattern und einem Inhalt bestehen. Das Pattern ist ein XPath-Ausdruck anhand dessen ausgewählt wird, für welchen Quellknoten es gilt. Der Inhalt gibt an, wie sein Zielinhalt generiert werden soll. Es gibt eine Reihe von Funktionen, die mit einer Ausgabe kombiniert werden können. So kann die Ausgabe vorher sortiert oder eine Summe von nummerischen Werten gebildet werden.

Folgend ein einfaches Template-Beispiel, bei dem mittels XPath-Ausdruck alle `<title>`-Elemente als `<h1>`-Überschriften ausgegeben werden:

```
<xsl:template match="//title">
  <h1>
    <xsl:apply-templates />
  </h1>
</xsl:template>
```

Listing 6: Einfaches Beispiel für ein XSLT-Template

Die aktuelle Version von XSLT ist die Version 2.0.[33] Seit dem 19. November 2015 gibt es die Candidate Recommendation für XSLT 3.0.[34] Zusammen mit der Version 3.0 von XPath[35] ergeben sich dann neue bzw. erweiterte Möglichkeiten z. B. bei der funktionalen Programmierung, neue Funktionen, Error-Handling mit try/catch, XSLT-Streaming und Auswertung von dynamisch erzeugten XPath-Ausdrücken.

3.3.2 XPath

Während XSLT die Transformation von einem XML-Dokument in ein anderen XML-Baum beschreibt, dient XPath mittels sogenannter Lokalisierungspfade zum Ausdruck von Baumbestandteilen. Ein Lokalisierungspfad besteht im Wesentlichen aus drei Bestandteilen:

Bestandteil	Beschreibung	Beispiele
Achse	Beziehung zwischen Kontextknoten und auszuwählenden Knoten	`//book/part`
Knotentest	Abfrage des Knotentyps oder Knotennamens	`//child::book/part`
Prädikate	(Optional) Abfrage von Eigenschaften z. B. unter Verwendung von Funktionen und Relationszeichen	`//book/part[@title eq "Erstens"]`

Tabelle 1: Bestandteile eines XPath-Lokalisierungspfades

Für eine vollständige Referenz aller XPath-Achsen, Knotentests und Prädikate empfiehlt sich der Blick in die XPath-Spezifikation[36]. XPath unterstützt auch Kontrollstrukturen, wie `if-then/else`, mit denen sich die einzelnen Fälle verarbeiten lassen.

[33] (W3C, 2007).
[34] (W3C, 2015).
[35] (W3C, 2014).
[36] (W3C, 2014).

3.4 HTML5

HTML wurde auf Basis von SGML von Tim Bernes-Lee entwickelt.[37] Mit HTML wurde es möglich, unter den Mitarbeitern des CERN schnell Forschungsergebnisse zu teilen und miteinander zu verlinken. Um den Standard HTML offen und lizenzfrei schneller zu verbreiten, wurde bereits 1994 das W3C gegründet, das von nun an Empfehlungen[38] für mehr als 35 technische Spezifikationen[39] festlegt. Obwohl das W3C keine verpflichtenden Normen festlegen kann, gelten diese quasi als Standard. Die aktuelle Empfehlung des W3C bezüglich HTML ist die Version 5 vom 28. Oktober 2014.[40]

Die weitere Entwicklung von HTML wurden durch verschiedene Hersteller wie Microsoft oder Netscape vorangetrieben. Diese herstellerspezifischen Erweiterungen führten zu Inkompatibilitäten und den sogenannten „Browserkrieg“. Aktuell ist die Kompatibilität zwischen den Browsern wieder deutlich höher, wenn auch noch einige Elemente bzw. Komponenten uneinheitlich umgesetzt sind. Zu nennen wäre hier insbesondere die Unterstützung von Audio- und Video-Formaten und die uneinheitlichen `input`-Typen. Wobei auch hier eine langsame Harmonierung stattfindet.[41]

Möchte man in wenigen Sätzen die wesentlichen Neuerungen von HTML5 nennen, fallen schnell Komponenten wie die neuen Abschnitts-Elemente, z. B. `<nav>`, `<main>` oder `<header>`, die `data`-Attribute oder die Unterstützung von Video- und Audio-Formaten ein. Da eine vollständige Auflistung den Kern der Arbeit verlassen würde, wird hier nur kurz auf die Neuerungen bezüglich der Abschnitts-Elemente und der `data`-Attribute eingegangen, da diese für die Umsetzung zum Einsatz kommen.

Ein Großteil von Websites im Internet sind bereits vor HTML5 logisch in Abschnitte wie Header oder Footer eingeteilt gewesen. Somit stellt sich die Frage, welche Vorteile Abschnitts-Elemente wie `<header>` oder `<footer>` gegenüber `<div>`-Containern mit ID bringen sollen. Ein erster Vorteil ist die für Suchmaschinen einfachere Möglichkeit, den Inhalt von Webseiten genauer beurteilen zu können.[42] Der Zweite Vorteil ist eine bessere Übersicht über den Code zu erhalten, böse Zungen sprechen sogar von einer Bekämpfung des „Divitis“-Syndroms.[43]

[37] (Schraitle, 2004, p. 10).
[38] Engl.: Recommendations.
[39] (Schraitle, 2004, p. 11).
[40] (W3C, 2014).
[41] Beim Kampf H.264/H.265 gegen VP8/VP9, hat Microsoft die Unterstützung von WebM für den Browser Edge angekündigt (iX, 2015).
[42] Eine Optimierung dieser Art (genauere Semantik) fällt in den Bereich SEO (Search Engine Optimization).
[43] (P.Hogan, 2011, pp. 23-24).

Eine Übersicht über die neuen Strukturellen Elemente (Abschnitts-Elemente):

Tag	Beschreibung
`<header>`	Kopfbereich eines Abschnitts oder einer Seite
`<footer>`	Fußbereich eines Abschnitts oder einer Seite
`<nav>`	Navigationsbereich eines Abschnitts oder einer Seite
`<section>`	Logischer Abschnitt einer Seite oder Gruppe von Inhalten
`<article>`	Artikel oder in sich geschlossenes Inhaltselement
`<aside>`	Sekundäre Inhalte

Tabelle 2: HTML5 Abschnitts-Elemente

Diese Tags werden von allen modernen Browsern unterstützt. Probleme bereiten alte Versionen vom Microsoft Internet Explorer (IE), die trotz der Deklaration als EOL[44] noch oft verwendet werden.[45] Für den IE 8 existiert aber ein Polyfill[46], welches sich via bedingten Kommentar[47] einbinden lässt.[48] Ab IE 9 werden die Elemente nativ unterstützt.

Eine kleine Veränderung in HTML5 ist die Dokumenttypangabe. Die folgende kurze Angabe im Prolog des HTML-Dokumentes fällt deutlich kürzer und leichter zu merken aus als es noch bei HTML4 oder XHTML der Fall war:

```
<!DOCTYPE html>
```

Listing 7: HTML5 DOCTYPE Deklaration

Durch den Verzicht auf eine Versionsangabe wechselt z. B. der Internet Explorer in den „Standards-Mode“ und wird damit sogar vom IE 6 erkannt. HTML5 versteht sich hierbei als Obermenge aller „HTML-Standards“.

Dritte Neuerung in HTML5 sind die neuen benutzerdefinierten Datenattribute `data-*`. Hier lassen sich Informationen ablegen, für die bisher vor allem Klassennamen missbraucht wurden. Somit wird eine bessere Trennung zwischen der Darstellung und dem Markup erreicht.[49] Validierer in Code-Editoren ignorieren meist diese Attribute und verhindern somit „false positives“, also Falschmeldungen über Syntax- oder Validierungsfehler. Die Attributnamen werden von der JavaScript-Engine der Browser automatisch vom „Kebab-Case“[50] zum „lowerCamelCase“ [51] umgewandelt.

[44] Microsoft stellt seit dem 12. Januar 2016 keine Updates mehr für ältere Versionen zur Verfügung (Microsoft, 2016).
[45] Laut Statistik für das Jahr 2015, lag der Anteil an IE 8 noch bei 14,15%, vom IE 9 bei 7,37%, vom IE 10 bei 4,98% und IE 11 bei 25,09% (NETMARKETSHARE, 2016).
[46] Polyfill ist eine Analogie zu Moltofill, einer Fugenspachtel. Polyfill soll sozusagen Löcher in der Unterstützung von Funktionen „stopfen“.
[47] Das Markup `<!--[if lt IE 9]> … <![endif]-->` wird nur von Internet Explorer älter als Version 9 verarbeitet.
[48] (Visscher, 2016).
[49] Auch wenn die `data`-Attribute Teil des Markups sind.
[50] Kleingeschriebene Bestandteile getrennt durch Bindestrich, z. B. foo-bar.
[51] Binnenmajuskel mit kleinem Anfangsbuchstaben, z. B. fooBar.

Beispiel mit lowerCamelCase-Umwandlung:

HTML-Markup	JavaScript

`<button`
`  id="btn"`
`  data-custom-string="myString">`
`</button>`
	`var select =`
	`document.getElementById("btn");`
	`select.dataset.customString;`

Tabelle 3: Beispiel data-Attribut

3.5 CSS

CSS ist eine vom W3C beständig weiterentwickelte[52] Stylesheet-Sprache. Mittels CSS ist es möglich, Darstellung von Inhalt und Struktur zu trennen[53], was auch vom W3C empfohlen wird. Damit wird die Wartbarkeit von Dokumenten deutlich erleichtert. So ist als praktisches Beispiel ein Wechsel des Designs einer Webseite möglich, ohne die HTML-Dokumente neu bearbeiten zu müssen. Die letzte monolithische Version die zum Standard (Recommendation) erhoben wurde ist CSS 2.1.[54]

CSS 3 wird anders als die Vorgängerversionen in Modulen gepflegt. So sind z. B. die Selektoren Level 3 bereits zum Standard ernannt.[55] Viele heute bereits eingesetzte Funktionen von zukünftigen standardisierten CSS 3-Modulen werden bereits heute von den meisten Browsern unterstützt. Hier dienen als Referenz die Übersichten von Can I use[56], um einen Überblick zu erhalten inwieweit eine Funktion von unterschiedlichen Browsern unterstützt wird. Tendenziell kann man die Entwicklung von CSS so beschreiben, dass es zunehmend Funktionen umsetzen kann, für welche früher JavaScript notwendig war.

CSS besteht im Wesentlichen aus zwei Komponenten. Den Selektoren und den Eigenschaften (Properties). Mit den Selektoren werden Elemente ausgewählt, denen man dann die Eigenschaften zuweist. Mehrere Eigenschaften lassen sich zu einem funktionalen Block in geschweiften Klammern zusammenfassen. Die häufigsten verwendeten Selektoren dürften die nach Typ, ID und Klassen sein.

Einige Beispiele:

Selektor (Beispiel)	Beschreibung
`h1 { … }`	Selektiert alle Überschriftelemente des Typs `h1`.
`#heading { … }`	Selektiert das Element mit dem Attribut `id="heading"`[57]
`.btn { … }`	Selektiert alle Elemente mit dem Attribut `class="btn"`
`input[type=search] { … }`	Selektiert alle Input-Elemente mit dem Attribut `type="search"`

Tabelle 4: Beispiel CSS-Selektoren

Es gibt über die genannten Beispiele hinaus auch die Möglichkeit nach anderen Attributen oder Pseudo-Klassen zu selektieren. Da eine vollumfängliche Auflistung aller Möglichkeiten den Rahmen dieser Arbeit sprengen würde, empfiehlt sich der Blick in die Spezifikation.[58]

[52] Auch als „living standard“ bezeichnet.
[53] Es ist auch möglich CSS mit in die Struktur zu integrieren.
[54] (W3C, 2011).
[55] (W3C, 2011).
[56] (Can I use, 2016).
[57] Der Attribut-Wert darf im Dokument immer nur einmal vorkommen.
[58] (W3C, 2011).

Die zweite große Komponente von CSS sind die Eigenschaften, die selektierten Elementen zugewiesen werden können. Bei mehr als einer Eigenschaft, bestehend aus durch Doppelpunkt getrennter Eigenschaft und deren Wert, muss mit Semikolon getrennt werden.[59]

Beispiel:

```
#header {
  font-weight: bold;
  font-size: 20px
}
```

Listing 8: Einfaches CSS-Beispiel

Neben den Grundkomponenten kommt mit CSS 3 noch eine für responsives Webdesign wichtige Komponente hinzu, die Media Queries. Mittels Angabe des Medientyps und dessen Eigenschaften (z. B. die Auflösung) kann man Selektoren und deren Eigenschaften bestimmten Medien zuweisen. So lässt sich zum Beispiel die Breite eines `div`-Containers auf großen Bildschirmen beschränken, um einen besseren Lesekomfort zu erreichen:

```
@media (min-width: 768px) {
  #main {
    max-width: 1000px;
  }
}
```

Listing 9: Media Query mit Abfrage der Display-Breite

Media Queries werden von allen modernen Browsern unterstützt. Ausnahme ist der noch oft verwendete Internet Explorer 8, für den ein Polyfill existiert[60] der sich ebenfalls via bedingten Kommentar einbinden lässt, sofern JavaScript aktiviert ist.

Wie der Name von CSS schon andeutet, ist ein weiteres wichtiges Prinzip von CSS die Kaskadierung. Nicht nur bei der Verwendung von Media Queries kommen mehrfache Deklarierungen von Stilregeln häufig vor. Auch jeder Browser bringt eigene browserspezifische Stilregeln mit, die durch den Autor einer Webseite überschrieben (stärker gewichtet) werden. Auch empfiehlt sich bei Verwendung von Frameworks[61], zwecks Update-Fähigkeit, eigene Stilregeln in einer separaten CSS zu definieren.

[59] Beim der letzten Eigenschaft, darf das Semikolon weggelassen werden.
[60] (Response JS, 2016).
[61] Zum Beispiel Twitter Bootstrap (Twitter, 2015).

Die Kaskadierung erfolgt in 4 Stufen:

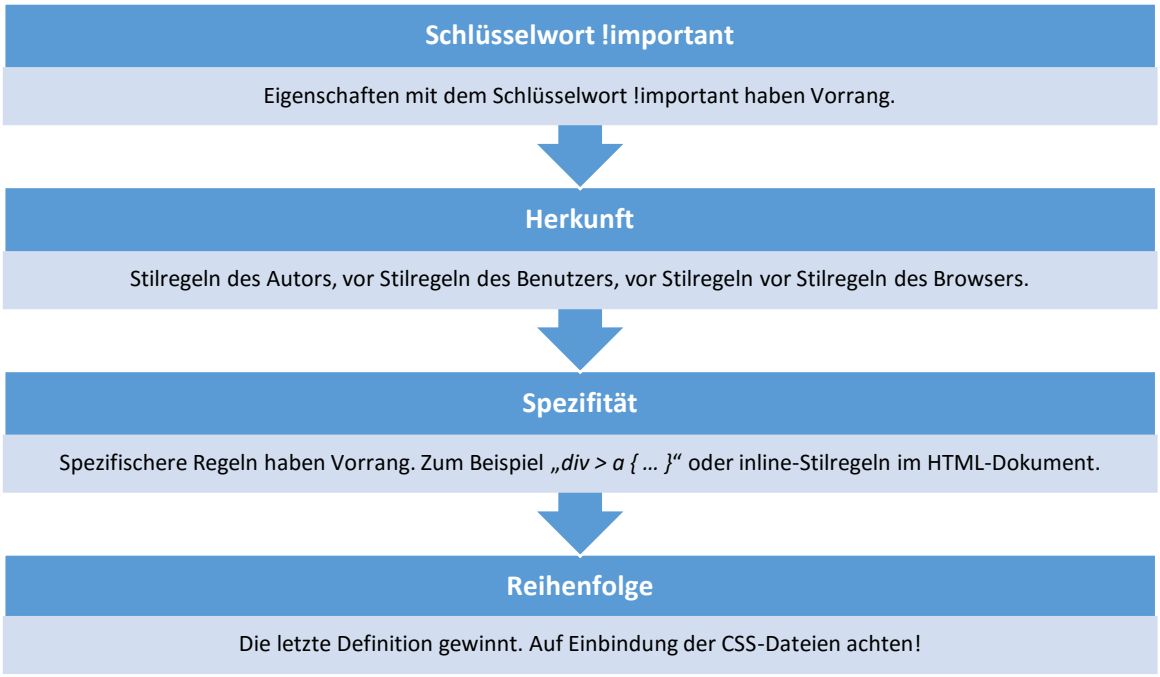


Abbildung 6: CSS-Kaskade

Die Kaskade wird von oben nach unten durchlaufen bis ein Fall ausschließlich für eine Definition zutrifft. Die darunterliegenden Stufen spielen dann keine Rolle mehr.

3.6 DOM

Document Object Model kurz DOM „[…] ist eine Spezifikation einer Schnittstelle für den Zugriff auf HTML- oder XML-Dokument“[62] und wurde vom W3C definiert[63]. DOM ist also kein Modell, sondern eine Schnittstelle, welche es Programmen und Scripts ermöglichen soll dynamisch auf den Inhalt, die Struktur oder den Stil eines Dokumentes zuzugreifen und ihn gegebenenfalls zu verändern. Es existieren verschiedene Versionen der Spezifikation, sogenannte DOM Level. Die letzte standardisierte Version ist DOM Level 4.[64]

Wichtigstes Element ist der DOM-Baum. Innerhalb dieses Baumes kann man z. B. mittels JavaScript oder XPath navigieren, d. h. sämtliche Elemente stehen in Bezug zueinander, ähnlich eines Stammbaumes in der Genealogie. Analog zu XML-Technologien gibt es immer genau ein Wurzelelement.

3.7 SVG

SVG ist eine auf XML basierende Markup-Sprache für zweidimensionale Vektorgrafiken und gilt als zusammengefasste Weiterentwicklung der Markup-Sprachen VML[65] und PGML[66]. Mittels SVG ist es möglich Grafiken darzustellen, die unabhängig von ihrer Skalierung immer die gleiche Qualität besitzen. Diese Eigenschaft prädestiniert SVG als Technologie für responsive Webdarstellung

[62] (Wikipedia, 2015).
[63] (W3C, 2005).
[64] (W3C, 2015).
[65] Vector Markup Language ist ein von Microsoft und Macromedia entworfener Standard.
[66] Precision Graphics Markup Language ist ein von Adobe, IBM, Netscape und Sun entworfener Standard.

gegenüber gerasterten Grafiken. Sie wird von allen modernen Browsern ohne Plug-ins unterstützt. Auch zahlreiche Bildbearbeitungs- und Grafikprogramme unterstützen einen SVG-Export.

Die erste empfohlene Spezifikation wurde im September 2001 veröffentlicht.[67] Die aktuelle Empfehlung vom W3C stammt vom 16. August 2011.[68] Speziell für Mobilgeräte angepasst existiert eine schlanke Spezifikation SVG Tiny 1.2.[69] Für die SVG-Versionen 1.0 und 1.1 existieren offizielle DTDs, die zur Validierung referenziert werden können.[70]

3.7.1 Koordinatensystem

Der Koordinatenursprung des SVG-Koordinatensystem liegt in der oberen linken Ecke des Zeichenbereichs. Das Koordinatensystem ist dimensionslos, d. h. es existieren initial keine Maßeinheiten. Während die X-Achse nach rechts zeigt, zeigt die Y-Achse nach unten. Mittels der Attribute `width` und `height`, wird das Koordinatensystem dimensioniert. Gibt man für das Wurzelelement keine Dimensionen an, wird die Größe auf 100% des verfügbaren Anzeigebereichs impliziert. Dimensionsangaben kann man sowohl relativ (Prozent oder (`r`)`em`) als auch absolut (Meter, Zoll, Pixel) angeben. Bei den Tiny-Versionen sind nur dimensionslose Längen zulässig.

Mittels der Attribute `viewBox` kann man einen Zeichenbereich in einen Anzeigebereich transformieren. Man muss sich also beim Erstellen der Grafiken noch keine Gedanken machen, ob sie auf den jeweiligen Bildschirm passt. Durch die Anpassung des `viewBox`-Attributes kann man die Grafiken dann automatisch für das jeweilige Anzeigegerät einpassen.

Beispiel einer viewBox-Transformation eines großen Rechteckes auf ein Anzeigegerät mit einer Auflösung von 1024 x 768 Pixel:

```
<svg width="1024" height="768" viewBox="0 0 2000 1000">
  <rect x="0" y="0" width="2000" height="1000" fill="black" />
</svg>
```

Listing 10: viewBox-Transformation

Mit dem Attribut `transform` sind weitere Transformationen möglich. Eine kurze Übersicht über mögliche Transformationen:

[67] (W3C, 2001).
[68] (W3C, 2011).
[69] (W3C, 2008).
[70] Beispielsweise die DTD für SVG 1.1 unter https://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd.

Operation	Transformations-Matrix	Markup
Verschiebung	$\begin{Bmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{Bmatrix}$	`translate(x, y)`
Skalierung	$\begin{Bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{Bmatrix}$	`scale(a, d)`
Rotation	$\begin{Bmatrix} \cos(w) & -\sin(w) & x \\ \sin(w) & \cos(w) & y \\ 0 & 0 & 1 \end{Bmatrix}$	`rotate(w, x, y)`
Scherung	$\begin{Bmatrix} 1 & \tan(u) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{Bmatrix}$ bzw. $\begin{Bmatrix} 1 & 0 & 0 \\ \tan(v) & 1 & 0 \\ 0 & 0 & 1 \end{Bmatrix}$	`skewX(u)` **bzw.** `skewY(v)`
Matrix	$\begin{Bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{Bmatrix}$	`matrix(a,b,c,d,e,f)`

Tabelle 5: SVG-Transformationen

3.7.2 Elemente

Das Wurzelelement eines SVG-Markups ist immer das Element `<svg>`. Mit dem Wurzelelement instanziiert man ein Koordinatensystem, das Basis für die Kindelemente ist. Fügt man eine SVG-Grafik direkt im HTML-Code ein, wird jeder moderne Browser dieses Element als SVG-Grafik behandeln und benötigt keine Einbindung des SVG-Namespaces und damit kein Namespace-Präfix.

SVG bietet eine Auswahl an geometrischen Formen, aus denen sich komplexere Objekte zusammensetzen lassen. Während sich ein Kreis oder Rechteck noch vergleichsweise bequem mit dem Code-Editor einsetzen lässt, erfordern komplexere geometrische Formen den Einsatz eines Frameworks oder Grafik-Editors. Beispiele für komplexe geometrische Formen sind z. B. die Polylinie oder der Path. Gerade letzteres Beispiel ist oft sehr komplex und kann wieder eine beliebige Anzahl von Unterpfaden enthalten und ermöglicht z. B. das Zeichnen von Bézier-Kurven[71].

Um eine Barrierefreiheit von SVG-Grafiken zu gewährleisten, sollte das Wurzelelement mindestens ein Kindelement `<title>` in dem eine Kurzbeschreibung enthalten sein sollte und optional ein Element `<desc>` mit genauerer Beschreibung enthalten. Ein Element `<metadata>` ist ebenfalls möglich, um Metadaten anzugeben. Metadaten können nach dem XML-Format RDF[72] oder gemäß Dublin Core[73] angegeben werden.

[71] Parametrisch modellierte Kurve.
[72] W3C Standard zur Beschreibung von Metadaten (W3C, 2014).
[73] Konvention zur Beschreibung von Dokumenten (Dublin Core Metadata Initiative, 2001).

Folgend eine kleine Auswahl von SVG-Elementen:

Beschreibung	Tag	Wichtige Attribute
Kreis	`<circle>`	Koordinate Mittelpunkt (cx, cy)
		Radius (r)
Rechteck	`<rect>`	Koordinate obere linke Ecke (x, y)
		Breite und Höhe (width, height)
Text	`<text>`	Startkoordinate obere linke Ecke (x, y)
Linie	`<line>`	Startkoordinate (x1, y1)
		Endkoordinate (x2, y2)
Ellipse	`<ellipse>`	Koordinate Mittelpunt (cx, cy)
		Radien für Weite und Höhe (rx, ry)
Polygon	`<polygon>`	Koordinaten der Eckpunkte (points)
Polygonzug	`<polyline>`	Koordinaten der Start-, End- und Zwischenpunkte (points)
Pfad	`<path>`	Pfad-Daten d-Attribut
Bild	`<image>`	Pfad, Breite, Höhne und Koordinaten (url, width, height, x, y)
Gruppe	`<g>`	
Gruppe (Pool)	`<defs>`	

Tabelle 6: Wichtige SVG-Elemente

3.7.3 Properties

Jedes Element benötigt Attribute, um dessen Größe, Position oder z. B. Farbe zu beschreiben. Die meisten Properties können als Inline-Style mittels `style`-Attribut gesetzt werden. Wie XML-Technologien allgemein, sollte hier aber eine Trennung von Markup und Darstellung getrennt und CSS zur Definition der Darstellung genutzt werden.

3.7.4 Filter

Neben den reinen geometrischen Formen bietet SVG eine Reihe von Effekten und Filtern, welche die Möglichkeiten der Gestaltung deutlich erweitern. Um einen Effekt auf ein Element bzw. Objekt anzuwenden, muss er erst einmal in einen `<defs>` Element definiert werden. Die darin enthaltenen Filter oder Effekte erhalten dann eine ID, auf die dann im anzuwendenden Element referenziert wird. Die Auswahl an Effekten und Filtern ist groß, daher soll hier nur ein exemplarisches Beispiel vorgestellt werden:

```
<svg height="200" width="200">
  <defs>
    <filter id="filter1">
      <feGaussianBlur in="SourceGraphic" stdDeviation="5" />
    </filter>
  </defs>
  <circle cx="100" cy="100" r="50" fill="red" filter="url(#filter1)" />
</svg>
```

Listing 11: SVG-Filter „Blur“ auf roten Kreis

Einige Filter sind schon von den Möglichkeiten aus CSS 3 bekannt, welche auf HTML-Elemente angewandt werden können. Typische Beispiele sind Unschärfe (engl.: blur), Schlagschatten oder Kontrast. SVG bietet im Gegensatz zu CSS 3 aber einige Filter mehr[74] und mit einem kleinen Trick

[74] CSS 3 hat 10 verschiedene Filter, SVG bietet 19 verschiedene Filter.

lassen sich sogar SVG-Filter in Kombination mit etwas CSS 3-Markup auf HTML-Elemente anwenden.[75] Eine kurze Übersicht über die SVG-Filter findet sich auf der Webseite von W3Schools.[76]

3.7.5 Animation

Anwendungen und Übergänge zu SVG-Filtern und Transformationen lassen sich auch animieren. Dazu stehen verschiedene Mittel zur Verfügung. Einerseits kann man auf CSS 3 Transitions[77] zurückgreifen oder man verwendet SMIL[78], eine vom W3C standardisierte, auf XML basierende, Auszeichnungssprache. Probleme gibt es bei der Unterstützung durch die Browser, bei denen gerade Microsofts Internet Explorer und Edge Schwierigkeiten bereiten. Eine Möglichkeit könnte es bei diesen Browsern sein das Element `<object>` zu verwenden, welches über das `data`-Attribut auf die SVG-Datei referenziert. In den `<object>`-Container kann dann Fallback-Code[79] eingefügt werden. Die Variante mit der besten Browserkompatibilität setzt auf JavaScript und beispielsweise einer Funktion `setInterval()`, welche eine SVG-Filter oder -Transformation ausführt.

3.8 JavaScript

Oft mit JS abgekürzt, ist JavaScript eine Skriptsprache für clientseitiges dynamisches HTML in Webbrowsern. Inzwischen gibt es auch serverbasiertes JavaScript, z. B. wie es mit dem NodeJS-Framework[80] möglich ist. Der aktuelle Standard wurde im Juni 2015 als ECMAScript 2015 veröffentlicht. Während JavaScript ursprünglich für einfache Effekte auf Webseiten verwendet wurde[81], wird sie heute für komplette Applikationen genutzt, die im Browser lauffähig sind.

JavaScript bietet die meisten Kontrollstrukturen, wie sie in anderen Programmiersprachen zu finden sind und ist objektorientiert. Variablen lassen sich lokal in Funktionen und global definieren. JavaScript unterstützt auch ein Exception Handling mit `try … catch … finally`. Für eine umfassende Einführung und Übersicht zu den Möglichkeiten von JavaScript wird das Buch ELOQUENT JAVASCRIPT[82] empfohlen.

Ein Vorteil und zugleich Nachteil von JavaScript ist das Sandbox-Prinzip. Da JavaScript-Anwendungen, mit Ausnahme von NodeJS, im Browser ausgeführt werden, sind überwiegend nur Zugriffe innerhalb des Browsers zulässig. Ein Zugriff z. B. auf das Dateisystem ist nur über ein `<input type="file">` möglich. Auch auf externe Ressourcen kann nicht zugegriffen werden, weshalb ein Datenaustausch zwischen Webservern nicht möglich ist.

In JavaScript ist die DOM-Schnittstelle implementiert, weshalb ein Zugriff auf HTML-Elemente und Attribute durch Selektoren möglich wird.

Eine weitere Besonderheit von JavaScript gegenüber serverbasierenden Script-Sprachen wie PHP ist das Event-Handling. JavaScript erkennt eine Reihe von Events, wie `click`, `resize` oder `mouseover`, welche für Benutzerschnittstellen notwendig sind. Es können auch eigene Events definiert werden.

---

[75] (Potschien, 2016).
[76] (W3Schools, 2016).
[77] Noch als Working Draft deklariert (W3C, 2013).
[78] Die Working Group wurde geschlossen, daher sind CSS 3 Transitions die wohl zukunftsfähigere Technologie (W3C, 2008).
[79] Alternativ auszuführender Code oder darzustellendes Markup.
[80] Siehe www.nodejs.org.
[81] Bis z. B. auf das `click`-Event lassen sich viele Funktionen inzwischen mit CSS 3 abbilden.
[82] (Haverbeke, 2015).

3.8.1 jQuery

jQuery ist das bekannteste und meist verwendete[83] JavaScript-Framework. Es bietet eine teilweise vereinfachte Syntax und zusätzliche Funktionen (z. B. eine `each`-Funktion zum Iterieren von Arrays). So sind z. B. die Selektoren an die CSS 3-Syntax angelehnt. jQuery bietet auch eine kleine Sammlung an Funktionen für Animationen und Effekte, sowie für AJAX. Für Benutzerschnittstellen existiert zudem ein Plug-in namens jQuery UI.[84]

Von jQuery (Core) existieren zwei Versionsstränge, welche unabhängig voneinander aktiv gepflegt werden. Während jQuery vor allem ältere Browser wie den Internet Explorer 8 (IE8) noch unterstützt, wurde bei jQuery 2.0 grundlegend umstrukturiert und aufgeräumt, auf Kosten des IE8-Supports. Die aktuellen Versionen beider Versionsstränge haben eine kompatible API.

Der Aufruf der jQuery-Funktionen und Selektoren erfolgt über die `$`-Funktion. Ein Vergleich zweier JavaScript Code-Schnipsel zwischen jQuery und JavaScript offenbart die Vereinfachung der Syntax bei jQuery:

JavaScript	jQuery

`var selection = document.getElementByClassName("btn");`
`for (i=0; i<selection.length; i++) {`
`  // Anweisung`
`}`
	`$(".btn").each(function(){`
	`  // Anweisung`
	`})`

Listing 12: Vergleich zwischen JavaScript und jQuery

3.8.2 JSON

JavaScript Object Notation ist ein Datenaustauschformat. JSON ist durch zwei konkurrierende Standards spezifiziert.[85] Wie XML können Daten bei JSON verschachtelt werden. So kann ein Objekt ein Array enthalten, welches wiederum Objekte enthält. Als Zeichenkodierung wird UTF-8[86] verwendet.

[83] Im Dezember 2015 auf 68% aller Webseiten (W3Techs, 2015).
[84] (jQuery, 2014).
[85] RFC 7159 und ECMA-404.
[86] UTF-16 und UTF-32 sind ebenfalls möglich.

Die Datentypen die JSON unterstützt[87] sind fast denen von JavaScript analog[88]:

Datentyp	Beschreibung
Nullwert	`null`
Boolescher Wert	Wert `true` oder `false`.
Integer	Folge von Ziffern von `0` bis `9`, Dezimal mit Punkt (`.`), Vorzeichen (−).
Float	Exponenten werden mit `e`/`E` und einer Ziffernfolge nach *Integer*[89] repräsentiert. JSON und JavaScript kennen nur den Typ *number.*
String	In doppelten (") Anführungszeichen[90] eingeschlossene Zeichenketten.
Array	In eckigen Klammern `[ ... ]` und durch Kommata getrennte Liste von Werten oder Objekten.
Objekt	In geschweiften Klammern `{ ... }` eingeschlossene Schlüssel-Wert-Paare. Schlüssel und Wert werden durch Doppelpunkt (`:`) getrennt. Ein Schlüssel muss pro Objekt einzigartig sein und vom Typ *String*. Der Wert darf allen hier genannten Datentypen entsprechen.

Tabelle 7: JSON-Datentypen

Die Umwandlung zwischen JavaScript-Objekten und JSON-String erfolgt mittels der beiden JavaScript-Funktionen `JSON.stringify()` und `JSON.parse()`. `JSON.stringify()` wandelt ein Objekt in einen String um (Serialsierung), während `JSON.parse()` ein JSON-String in ein JavaScript-Objekt umwandelt. Ältere Browser unterstützen eventuell nur die Umwandlung über die Funktion `eval(JSON)`. Der sehr einfache Austausch von JavaScript-Objekten ist eine der größten Vorteile von JSON. Als Nachteil wäre die vergleichsweise schlechte Lesbarkeit[91] zu erwähnen.

3.8.3 AJAX

AJAX[92], ein Apronym für Asynchronous JavaScript And[93] XML, ist ein Modell einer asynchronen Datenübertragung zwischen Browser und Server. Mittels AJAX ist es möglich mittels HTTP-Anfragen Inhalte einer Webseite zu verändern, ohne sie komplett neu zu laden. Prominente Beispiele sind die Google-Suchmaschine (Live-Suche) und Google Docs. AJAX wird seit langem von allen Browsern unterstützt[94] und steht damit einem breiten Einsatz offen.

Der Einsatz von AJAX ist denkbar einfach. Die größte Schwierigkeit sollte allenfalls das Error-Handling bereiten, welches sich durch `try-catch`-Anweisungen und der Auswertung der HTTP-Statuscodes[95] aber gut abfangen lässt. Mittels dem `XMLHttpRequest`-Objekts instanziiert man eine neue AJAX-Abfrage. Der Methode `open()` übergibt man als Argumente die Sendemethode (`GET` / `POST`), die URL und einen Booleschen Wert, welcher angibt, ob die Abfrage synchron (`false`) oder asynchron (`true`) erfolgt. Da bei einer synchronen Abfrage die JavaScript-Funktion solange angehalten wird, bis eine Antwort vom Server erfolgt, wird meist die asynchrone Abfrage verwendet.

---

[87] (JSON, 2016) und (ECMA International, 2013, pp. 1-5).
[88] Funktions-, RegExp- und Error-Objekte werden verworfen. Datums-Objekte werden nach dem ISO-8601-Format konvertiert. `NaN` und (−) `Infinity` werden zu `null` serialisiert.
[89] Positives Vorzeichen (+) darf hier nicht weggelassen werden!
[90] Nicht in einfachen (') Anführungszeichen!
[91] Zum Beispiel im Vergleich zu YAML. Meist wird von einer besseren Lesbarkeit von JSON gegenüber XML gesprochen.
[92] Hier wird die Majuskel-Schreibweise bevorzugt. Ajax wäre auch richtig.
[93] Bewusst groß geschrieben, da der Anfangsbuchstabe ins Apronym einfließt.
[94] Unterstützung ab Mozilla 1.0, Netscape 7, Firefox, Safari 1.2, Opera 7 und Internet Explorer 5 (im Beispiel beschriebene Methode).
[95] Eine Übersicht der Codes findet sich auf (HTTP Status Codes, 2016).

```
var http = new XMLHttpRequest();
http.open("GET", "document.html", true);
http.onreadystatechange = function () {
  if (http.readyState == 4 && http.status == 200) {
    document.getElementById("container").innerHTML =
    http.responseText;
  }
}
```

Listing 13: Einfaches AJAX-Beispiel

Die asynchrone Abfrage erfordert eine Callback-Funktion, die an die Eigenschaft `onreadystatechange` übergeben wird. Das „[…] Ereignis `readystatechange` wird immer dann ausgelöst […], wenn sich der Zustand des HTTP-XML-Objektes ändert.“[96] Die Eigenschaft `readyState` kann dabei 5 verschiedene Werte annehmen:

Wert	Beschreibung
0	Nicht initialisiert
1	Lädt
2	Fertig geladen
3	Wartet
4	Fertig

Tabelle 8: Übersicht AJAX readyStates

Sobald der Server die angeforderten Daten (`responseText`) zurückgeliefert hat, können sie wie im Beispiel *Listing 13: Einfaches AJAX-Beispiel* in den DOM in das Element `#container` eingefügt (`innerHTML`) werden. Verwendet man jQuery für AJAX-Anfragen, verändert sich die Syntax, was hier aber nicht näher erläutert werden soll, sondern im praktischen Teil der Arbeit näher erläutert wird.

3.9 D3.js

Auch wenn der Name vielleicht Assoziationen zu 3D wecken könnte, steht D3 für Data-Driven Documents.[97] D3.js ist eine JavaScript-Bibliothek, deren Schwerpunkt in der Visualisierung von Daten liegt und basiert auf den Web-Technologien HTML, SVG und CSS. Neben der einfachen Darstellung großer Datensätze ermöglicht D3.js auch Interaktivität und Animationen. Die JavaScript-Bibliothek wird als GitHub-Repository gepflegt und auch via CDN[98] angeboten.

3.9.1 Selektoren

Ähnlich wie jQuery, bietet auch D3.js vereinfachte Selektoren an, die sich an die CSS-Syntax anlehnen. Im folgenden Beispiel wird nicht nur die kürzere Syntax für die Selektoren deutlich, sondern auch die Iteration eines Arrays wird von D3.js übernommen:

[96] (Wenz, 2010, p. 360).
[97] (Data-Driven Documents, 2016).
[98] Das neueste Release wird mit der URL „//d3js.org/d3.v3.min.js“ verlinkt.

JavaScript	D3.js

`var items = documents.getElementsByClassName("item");`
`for (var i=0; i<items.length; i++) {`
`  var item = items[i];`
`  item.classList.add("hidden");`
`}`
	`d3.selectAll(".item")`
	`.classed("hidden", true);`

Listing 14: Vergleich JavaScript mit D3.js

3.9.2 Dynamische Eigenschaften

Eine sehr praktische Funktion von D3.js sind die dynamischen Eigenschaften. Einer den Eigenschaften übergebenen Funktion werden automatisch zwei Argumente übergeben, das aktuelle Objekt selber (`d` = aktueller Node) und dessen Index (`i`):

```
node.append("circle")
.attr("xlink:href", function(d, i) {
  return d.url;
})
.attr("r", function(d, i) {
  return i;
});
```

Listing 15: Dynamische Eigenschaften in D3.js

3.9.3 Animation

Die Animation von Übergängen ist mit D3.js sehr einfach und vor allem sparsam. Durch die `transition`-Funktion werden nur die geänderten Eigenschaften animiert.

Ein einfaches Beispiel bei dem der Radius eines Kreises innerhalb einer Sekunde animiert vergrößert wird:

```
d3.selectAll("circle").transition()
  .duration(1000)
  .attr("r", function(d) {
    return d*2
  });
```

Listing 16: D3.js Transition Beispiel

3.9.4 Force-Directed Graph

Neben den bekannten Standard-Diagramm-Typen wie Balken-, Kreis- oder Kurvendiagrammen, bietet D3.js einen nativ unterstützen Diagramm-Typ, der sich „Force-Directed Graph“ nennt. Dieser Diagramm-Typ basiert auf Knoten (Nodes) und Verbindungen (Links), die ähnlich eines Atom-Modells durch die Verbindungen und Kräfte zueinander positioniert werden. Die Positionierung und Bewegungen basieren auf diversen Algorithmen aus der Physik und Chemie[99]. Inwieweit sich die Nodes anziehen und abstoßen, kann mit folgenden Parametern beeinflusst werden:

[99] Unter anderen dem Verlet-Algorithmus und dem Barnes-Hut-Algorithmus.

Eigenschaft	Beschreibung
force.linkDistance(*distance***)**	Entfernung zwischen zwei verbundenen Nodes
force.linkStrength(*strength***)**	Stärke der Links zwischen zwei Nodes
force.friction(*friction***)**	Entschleunigungsfaktor
force.charge(*charge***)**	Anziehung (+) / Abstoßung (-) von Nodes (Ladung)
force.chargeDistance(*distance***)**	Wirkungsradius der Ladung
force.gravity(*gravity***)**	Anziehungskraft aller Nodes um den Mittelpunkt

Tabelle 9: D3js Parameter für Atommodell

Praktisch an diesen Diagrammtyp ist, dass man sich beim Standard-Layout nicht um die Berechnungen der Positionen der einzelnen Nodes und Links kümmern muss. Um ein Force-Directed Graph zu starten, muss man der `force`-Funktion die Nodes und optional auch die Links übergeben.

```
var width = 1280,
    height = 720;
    svg = d3.select("body").append("svg")
      .attr("width", width)
      .attr("height", height),
    force = d3.layout.force()
      .size(width, height);
d3.json("graph.json", function(error, json) {
  if (error) throw error;
  force
      .nodes(json.nodes)
      .links(json.links)
      .start();
  var link = svg.selectAll(".link")
        .data(json.links)
      .enter().append("line")
        .attr("class", "link"),
      node = svg.selectAll(".node")
        .data(json.nodes)
      .enter().append("circle")
        .attr("class", "node")
        .attr("cx", function(d) { return d.x; })
        .attr("cy", function(d) { return d.y; })
        .attr("r", 8)
  force.on("tick", function() {
    link.attr("x1", function(d) { return d.source.x; })
        .attr("y1", function(d) { return d.source.y; })
        .attr("x2", function(d) { return d.target.x; })
        .attr("y2", function(d) { return d.target.y; });
    node.attr("transform", function(d) { return "translate(" + d.x + "," +
d.y + ")"; });
  });
});
```

Listing 17: Beispiel für ein Force-Directed-Graph

3.9.5 Tick-Event/Funktion

Das `tick`-Event bzw. die `tick`-Funktion kann ausgelöst werden, um von einem zufälligen Initial-Layout in ein stabiles Endlayout Zwischenberechnungen durchzuführen. Diese Simulation steht u. a. für den Force-Directed Graph zur Verfügung und ermöglicht es, die Positionen von Nodes und Links dynamisch zu berechnen und weiche Bewegungen und Übergänge zu erzeugen. Die Berechnung stoppt, indem die Methode `force.stop` aufgerufen wird oder wenn der Wert der Eigenschaft `force.alpha` unter 0.005 fällt. Die Eigenschaft `force.alpha` ist eine Art Abkühlungs-Parameter und kann für eigene Berechnungen ausgelesen und mit verarbeitet oder selber gesetzt werden. Man

kann auch eigene `tick`-Funktionen erstellen, dabei bleiben die Atommodell-ähnlichen Eigenschaften der Nodes und Links erhalten.

3.9.6 Enter und Exit

Mit den Funktionen `enter` und `exit` steuert man in D3.js das Hinzufügen und Entfernen von Elementen aus dem DOM anhand von Daten. Betritt man eine Auswahl, werden automatisch die Elemente zum DOM hinzugefügt `enter().append()`, welche noch nicht anhand der Daten im DOM hinzugefügt wurden. Werden Einträge aus den Daten entfernt, werden auch die Elemente aus dem DOM entfernt `exit().remove()`:

```
var node = d3.selectAll("circle")
  .data(myNodes);
node.enter()
  .append("circle")
    .attr("xlink:href", function(d) {
      return d.url;
    })
    .attr("r", 20);
node.exit().remove();
```

Listing 18: Enter und Exit mit D3.js

„Betritt“ man nicht die Auswahl mit `enter`, werden nur die vorhandenen Elemente aktualisiert.

3.10 Bootstrap

Twitters Framework Bootstrap[100], im Netz oft auch mit TBS abgekürzt, ist ein Framework für responsive Webseiten. Es besteht aus vordefinierten CSS-Klassen und einigen JavaScript-Funktionen. Genau genommen ist Bootstrap kein Framework für responsives Webdesign, sondern adaptives Webdesign. Gerade der adaptive Ansatz bietet eine genauere Kontrolle über die Darstellung der Inhalte, auf Kosten einer geringfügig eingeschränkten Kompatibilität von Geräten. Die Zeitschrift t3n hat die beiden Design-Ansätze miteinander verglichen.[101]

Wesentlicher Bestandteil von Bootstrap ist das Grid-System, welches in der Standard-Einstellung auf 12 Spalten basiert. Mittels Grid-System kann man eine Brücke von den Gestaltungsrastern in Print-Medien zu Webmedien schlagen. Einzelnen `div`-Elementen weist man dann eine oder mehrere Klassen mit gewünschter Spaltenbreite zu (von 1-12). Am Ende muss man nur darauf achten, dass eine Reihe (umgebender `div`-Container der Klasse `row`) Elemente mit einer Summe von 12 Spaltenbreiten enthält. Je nach Endgerät bricht das Layout dann von einer Reihendarstellung in eine Spaltendarstellung um.

[100] (Twitter, 2015).
[101] (t3n, 2015).

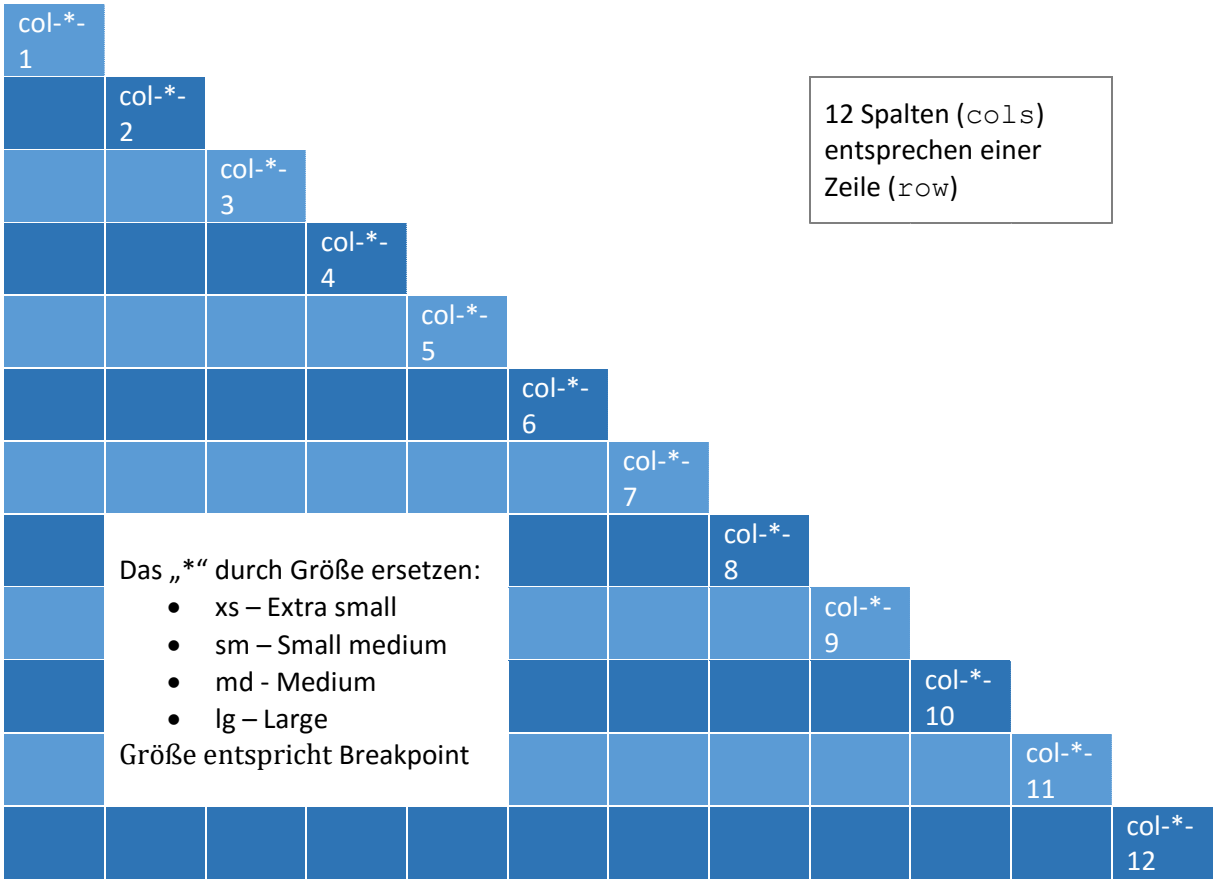


Abbildung 7: Grid-Klassen von Bootstrap

Weitere vordefinierte Elemente von Bootstrap sind beispielsweise der Glyphicons Font, eine Schriftart mit Symbolen, Buttons in allen möglichen Farben, Modals, Panels und vieles mehr. Eine Übersicht der Komponenten und Funktionen findet sich auf der Website von Bootstrap unter „Components“. Generell ist die lückenlose und mit reichlich Beispielen versehene Dokumentation des Frameworks einer der größten Vorteile. Eine Neuauflage des Frameworks ist bereits als Alpha-Version erschienen.[102]

Viele der Elemente sind in verschiedenen Farben verfügbar. Das Standard-Farbschema mit den Klassensuffixen von Bootstrap[103]:

Default	Primary	Info	Success	Warning	Danger

Abbildung 8: Standard-Farbschema von Bootstrap

3.11 Font Awesome

Ursprünglich als Teil von Twitter Bootstrap ist Font Awesome[104] eine Schrift, die Icons darstellt. Font Awesome bietet ein großes Spektrum an Icons, mit denen sich Benutzerschnittstellen sprachunabhängig (i18n) herstellen lassen. Man kann sich damit zum Teil die Lokalisierung (l10n) von Elementen des Benutzerinterfaces sparen und vermeidet mitunter Platzprobleme bei vielen Schaltflächen.[105] Der Umfang an verschiedenen Icons übertrifft die Auswahl bei den TBS Glyphicons

[102] (Twitter, 2016).
[103] Aufgrund der unterschiedlichen Farbsysteme (additiv ⇔ subtraktiv) sind die dargestellten Farben nur schematisch.
[104] (Font Awesome, 2016).
[105] Ein aussagekräftiges Symbol sagt manchmal mehr als tausend Worte.

deutlich und lässt für die meisten Anwendungszwecke kaum Wünsche offen. Die Einbindung des Fonts erfolgt als CSS, welches praktischerweise auch über das Bootstrap CDN angeboten wird.

```
<i class="fa fa-camera-retro"></i>
```

Listing 19: Font Awesome Beispiel mit Kamera-Icon

Das Beispiel *Listing 19: Font Awesome Beispiel mit Kamera-Icon* zeigt die Einbindung eines Font Awesome Icons. Die Größe der Fonts wird über eine extra zugewiesene Klasse definiert:

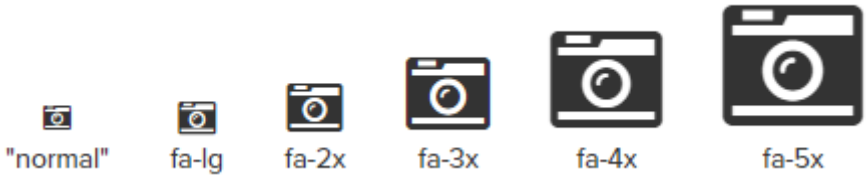


Abbildung 9: Größenklassen von Font Awesome

Neben den reinen Fonts und ihren Größen gibt es noch weitere Formatierungsklassen, die sich auf das Font-Element anwenden lassen. So lassen sich Icons mit Rahmen und Hintergrund versehen, Rotieren (z. B. `fa-rotate-90` oder auch animiert mit `fa-spin`), oder eine feste Weite zuweisen (`fa-fw`). Auf einer Webseite von Font Awesome[106] sind alle Möglichkeiten live zu begutachten.

[106] Unter den Menüpunkt „Examples“ auf (Font Awesome, 2016).

4 Styleguides und Programmierparadigmen

4.1 XSLT

Alle XSLT-Templates sind nach dem Standard XSLT 2.0 zu erstellen. Außerdem sollte auf verschachtelte Definition von Templates verzichtet werden. Es gilt das Salami-Slice-Prinzip[107], um dem Paradigma des funktionalen Programmierens gerecht zu werden.

4.2 JavaScript

Variablennamen und Funktionsnamen sind nach der lowerCamelCase-Variante zu benennen. Sind mehr als zwei Bedingungen pro Variable zu prüfen, ist Switch-Case zu verwenden. Alle Konsolenausgaben[108] sind im finalen Code zu entfernen. Es sind nur Funktionen zu verwenden, die folgende Browser ohne Workarounds[109] beherrschen:

- Microsoft Internet Explorer 9+
- Microsoft Edge
- Mozilla Firefox
- Google Chrome
- Apple Safari 5+

Sämtlicher JavaScript-Code ist in einer externen Datei einzubinden. Dies ermöglicht einfacheres Debugging, da dann die Angaben der Zeilennummer in der Konsole mit der Zeilennummer im Editor übereinstimmt. Diese Vorgehensweise hat bis zur Umstellung auf HTTP/2 zwar Nachteile bei der Geschwindigkeit des Seitenaufbaus[110], diese werden hier aber vernachlässigt.

Argumente sind auszuschreiben, um die Lesbarkeit des JavaScript-Codes zu erhalten.[111] Sollten Optimierungen bei der Speichergröße notwendig werden, kann auf einen separaten Minimizer-Prozess zurückgegriffen werden, wobei auch hier HTTP/2 Abhilfe schaffen sollte.[112]

JavaScript-Funktionen sind Variablen zuzuweisen. Damit sich die Funktionen unabhängig von der Position im JavaScript-Dokument aufrufen lassen sollen, sind die Variablen zu Beginn als leere Variablen vorzudefinieren. Um eine Kapselung der Anwendung zu erreichen, sollten globale Funktionen vermieden werden und einem Objekt als Methode angehangen werden.

Auch wenn viele Browser beim Weglassen von Semikolons sehr fehlertolerant sind, sollten keine Semikolons weggelassen werden. Zeichenketten (Strings) sind, soweit möglich, in doppelte, statt einfache Anführungszeichen zu setzen.

4.3 HTML

Mit Ausnahme externer Frameworks ist sämtlicher Code nach dem Google Styleguide für HTML und CSS[113] zu erstellen. Daten, die für Funktionen in JavaScript benötigt werden, sind in data-Attributen abzulegen. Klassen dienen ausschließlich für Selektoren in CSS und JavaScript. Klassennamen sind

---

[107] Templates werden global definiert und dann nur noch referenziert.
[108] Durch console.log() in JavaScript erzeugte Ausgaben.
[109] Programmweichen für browserspezifischen Code.
[110] Mit HTTP/2 können mehrere Anfragen an den Server gestellt werden (Multiplexing). Somit erübrigt sich das Zusammenfassen von Code in so wenig Dateien wie möglich.
[111] Zum Beispiel `function(event){...}` statt `function(e){...}`.
[112] HTTP/2 bringt auch neue Datenkompressionsmöglichkeiten mit.
[113] (Google, 2016).

nach dem Kebab-Case zu benennen. Die `click`-Events sollten nicht im HTML-Markup angegeben werden, sofern sich keine Laufzeit-Probleme ergeben.

4.4 CSS

Die Media Queries sind nach dem Mobile First-Konzept umzusetzen.[114] Das bedeutet, dass ohne Media Queries die Regeln definiert werden, wie die Darstellung auf den kleinsten Geräten zu erfolgen hat. Mittels Abfrage der Display-Auflösung wird die Darstellung für größere Bildschirme neu definiert.

Da dieses Projekt einige Frameworks als Grundlage nutzt, die bereits CSS-Regeln mit sich bringen, sollten diese als erstes eingebunden werden. Benutzerdefinierte Regeln, die Framework-spezifische Regeln überschreiben, werden in einer separaten Datei nach dem zugrundeliegenden Framework eingebunden. Somit bleiben die Framework-spezifische Styles unverändert und somit bedingt Upgrade-fähig.[115]

Die bei CSS Eigenschaften vorkommenden Zahlenwerte, die als Dezimalbruch unter Umständen mit einer führenden Null beginnen können, sind ohne führende Null zu schreiben. Das heißt der Wert beginnt mit einem Punkt.

[114] Entwurf und Umsetzung erfolgen erst für die kleinsten Geräte, bei größeren Geräten kann die Darstellung entsprechend erweitert werden. Mobile First gilt als Rückbesinnung auf die wesentlichen Inhalte.
[115] Zumindest lassen sich anhand von Migration-Guides Änderungen gezielter einpflegen.

5 Werkzeuge

5.1 Oxygen XML Editor

Oxygen XML Editor[116] ist einer der derzeit mächtigsten XML-Editoren. Er ist mittels Java programmiert und damit auf einem Großteil von Plattformen lauffähig. Oxygen XML bringt eine Reihe von DTDs für XML Formate von Haus aus mit und hat bereits XSLT- und XSL-FO-Prozessoren integriert. Die Oxygen-Lizenz ist kostenpflichtig. Obwohl Oxygen die Bearbeitung und Validierung von JavaScript und CSS unterstützt, wird aus persönlichem Geschmacksempfinden auf Visual Studio Code zurückgegriffen. Ähnliches gilt für den SVN-Client und den eigentlichen Transformationsprozess, bei dem auf eine Kommandozeilen-basierende Lösung (CLI) zurückgegriffen wird, da dies ein eingespielter Prozess im Unternehmens ist.

5.2 Cygwin

Cygwin[117] ist eine Art Mischung aus Kommandozeilen-Interface und Linux-Distribution, die als eigenständiges Programm unter Windows läuft. Mittels Cygwin und optional installierbaren Modulen lassen sich eine Vielzahl von Linux-Tools im CLI ausführen. Cygwin ermöglicht aber nicht das Ausführen von beliebigen Linux-Programmen, da hierfür wesentliche Bibliotheken fehlen (z. B. ein Windows-Manager/GUI). Die für das Projekt notwendigen, installierten Erweiterungen sind:

- SVN Client zur Versionsverwaltung.
- Makefile zum Erzeugen der Zieldateien und für den Transformationsprozess.
- ImageMagick[118] zur automatischen Umwandlung von Bilddateien.

5.3 Makefile

Mittels den kommandozeilenbasierenden Makefile-Tools wird der gesamte Erstellungsprozess aus den DocBook-Dateien zur Webseite gesteuert und aufgerufen. Im Wesentlichen werden die einzelnen sprachspezifischen DocBook-Dateien und die XSLT-Stylesheets dem Saxon-Prozessor übergeben und dieser aufgerufen. Dazu werden einige Variablen gebildet, wie die Sprachenkürzel oder Pfad-Präfixe. Außerdem ist ein Parameter „`Standalone`“ angelegt, der den Wert „`yes`“ oder „`no`“ annehmen kann und in den Transformationsprozess als Variable `externallinks` an das XSLT-Stylesheet übergeben wird. Dadurch werden z. B. für den Messestand-Betrieb die externen Links deaktiviert oder Reset-Timer aktiviert, der die interaktive Navigation nach einer definierten Zeit der Inaktivität auf die Standard-Einstellung zurücksetzt.

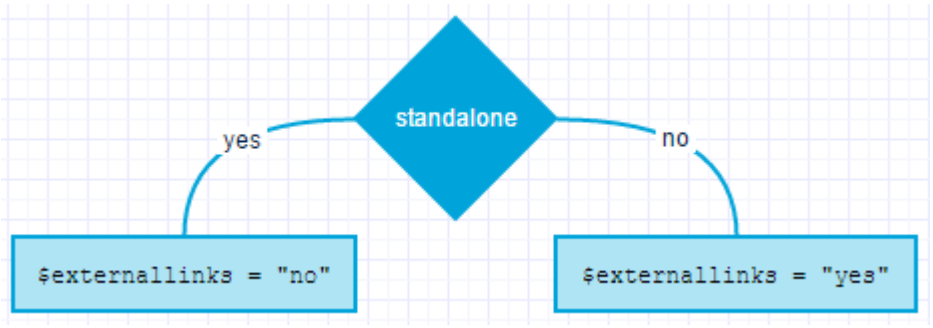


Abbildung 10: UML zum Parameter Standalone

Ebenfalls praktisch ist, in Kombination mit ImageMagick, die Stapelverarbeitung von Bildern, welche in vordefinierte Dimensionen eingepasst werden. Zudem bekommen die Bilder einen Rahmen im

[116] Zu finden unter www.oxygenxml.com.
[117] (Cygwin, 2015).
[118] (ImageMagick, 2016).

Corporate Design. Das alles lässt sich sehr schnell im Makefile an neue Bedürfnisse anpassen, ohne auf weitere u. a. kostspielige externe Programme oder Prozesse zuzugreifen.

Das Hochladen der Website auf den Apache-Webserver erfolgt in einem unabhängigen manuellen Deploy-Prozess, der nicht durch den Autor dieser Arbeit durchgeführt wird. Ein automatisierter Deploy-Prozess existiert nicht, da die Häufigkeit von Änderungen an der Website sehr gering ist.

5.4 Visual Studio Code

Visual Studio Code, oft auch mit VSC abgekürzt, ist ein Open Source Code Editor von Microsoft. Er bietet die klassischen Features wie Syntax Highlighting[119] oder Code Completion[120] für eine Vielzahl an Sprachen, wie CSS, HTML oder JavaScript. Um das Debugging zu erleichtern, kann man in einem Kommentar globale Variablen definieren und falsche Warnungen unterdrücken.

```
/* global d3, $ */
```

Listing 20: Kommentar mit globalen Variablen in Visual Studio Code

5.5 Google Chrome

Zur Webentwicklung ist ein Browser genauso unerlässlich wie ein Code-Editor. Mehr noch, meist ist eine Sammlung an Browsern notwendig, um Kompatibilitäten zu prüfen und unerwünschte Darstellungsfehler zu beseitigen. Überwiegend wurde auf Google Chrome zurückgegriffen, welcher mit zwei Plugins, die bei Google Extensions genannt werden, erweitert wurde.

5.5.1 Clear Cache

Diese Extension erfüllt einen einfachen Zweck: Sie löscht mit einem einzelnen Klick auf die Schaltfläche den Browsercache um das Laden der Ressourcen beim erneuten Laden der Seite zu bewirken. Im Extension lässt sich das erneute Laden der Seite gleichzeitig mit dem Löschen auslösen, somit erspart man sich einige Klicks.

5.5.2 Quick Javascript[121] Switcher

Mittels dieser Extension lässt sich JavaScript mit einem einzelnen Klick deaktivieren und die Seite neu rendern. Somit kann man das Verhalten der Webseite mit und ohne aktiviertem JavaScript sehr schnell überprüfen.

5.6 Andere Browser

Als weitere Browser für Kompatibilitätstests wurden Mozilla Firefox, Microsoft Edge und Safari verwendet. Somit sollen browserspezifische Fehldarstellungen vermieden werden.

[119] In JavaScript z. B. Hervorhebung von Variablen, Werten oder Argumenten anhand unterschiedlicher Zeichenfarben.
[120] Zum Beispiel automatisches Setzen von Schließtags bei noch offenen Elementen, sobald man mit der Zeichenkette </ beginnt.
[121] Sic!

6 Praktische Umsetzung

6.1 Visuelles Konzept

Im theoretischen Teil der Arbeit wurden bereits die Vorzüge der Navigation durch Querbeziehungen aufgrund des Roter-Faden-Konzepts dargelegt. Um den Roten Faden als Instrument sinnvoll umzusetzen, müssen zu jeder aktuell ausgewählten Seite die direkten Querbeziehungen visuell hervorgehoben werden. So entstand die Idee einer radialen Darstellung aller Knoten, wobei der aktuelle Knoten, welcher die aktuelle Webseite repräsentiert, in der Mitte des Navigationsbereichs platziert wird. Die Querbeziehungen werden radial um diesen Knoten als direkte Nachbarknoten auf der Umlaufbahn eines konzentrischen Kreises platziert. Auf einer weiteren Umlaufbahn eines weiteren konzentrischen Kreises befinden sich dann die verbliebenen Knoten, welche einen Bezug größer als ein Link (n>1) haben.[122] Mit jedem Umschalten der aktiven Seite, soll die Darstellung neu berechnet und die Knoten und Links transformiert werden.

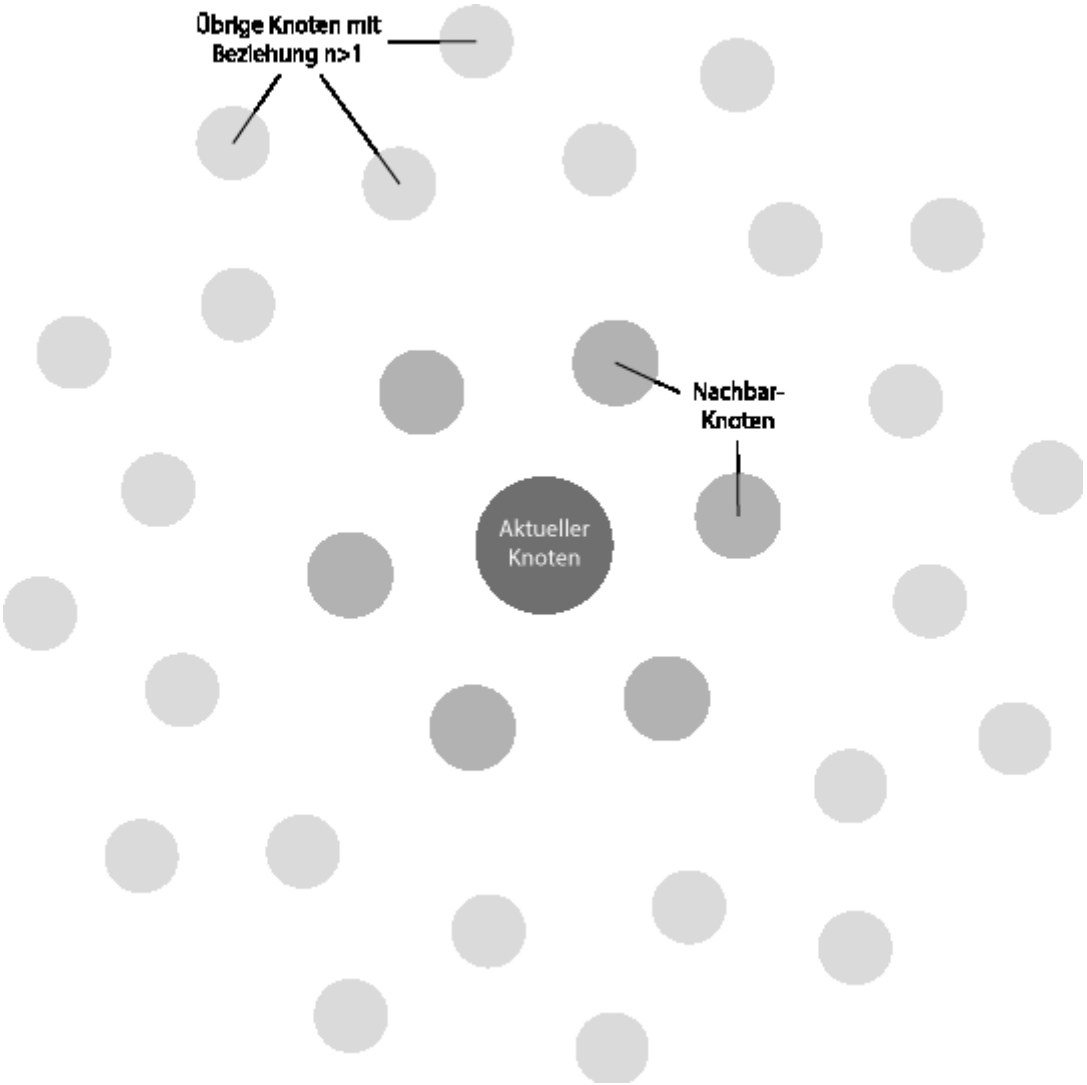


Abbildung 11: Konzept der radialen Darstellung

Am Messestand steht ein Display bereit, welches ein Seitenverhältnis im 16:9-Format aufweist[123] und mit einem Touchscreen ausgestattet ist. Die Webseite von le-tex weist ein Seitenverhältnis im

[122] In der Netzwerktechnik würde man von mehr als einen „Hop“ sprechen.
[123] Full HD-Auflösung mit 1920 x 1080 Pixel.

Hochformat auf, womit sich auf dem Messestand-Display und vielen PC-Bildschirmen[124] eine vertikal geteilte Ansicht anbietet. Für die Realisierung wird eine Side-by-side-Darstellung gewählt, welche den Kiosk in zwei jeweils 50% der verfügbaren Breite teilt. Auf der rechten Hälfte soll die Webseite dargestellt werden, auf der linken Hälfte dann die interaktive Navigation:

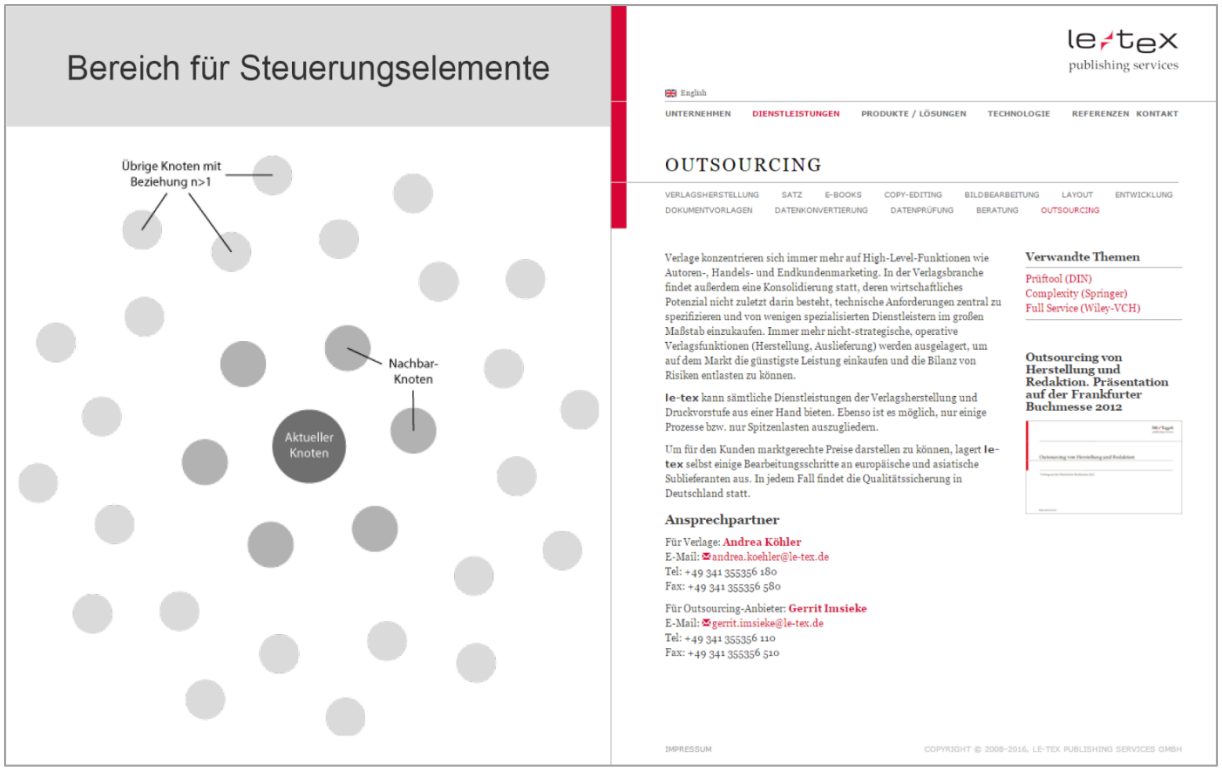


Abbildung 12: Konzept für die Side-By-Side-Darstellung

Der Bereich für die Steuerelemente enthält dann die Bedienelemente für die Schlagwort-Suche, die Typ-Filter und die Darstellung. Wie die Elemente am Ende angeordnet werden sollen muss dann in der Praxis eruiert werden, da man beim responsiven Webdesign nicht unnötig lange mit der Konzeptionsphase verbringen sollte.

Für die Darstellung der einzelnen Knoten wurde ein Design entworfen, welches die einzelnen Knoten nicht zu groß darstellt, aber auch noch ausreichend Platz für die Beschriftung bietet. Es wurde eine Kombination aus Kreis und horizontal gestrecktem Rechteck gewählt, welches die Knotenbeschriftung enthält. Der weiße Rahmen soll eine leichtere Unterscheidung von Knoten bei Überlagerung ermöglichen:

Abbildung 13: Layout-Konzept für die Knoten

[124] Viele PC- und Notebook-Bildschirme weisen ein Seitenverhältnis von ebenfalls 16:9, oder 16:10 auf.

Die Breite des Rechtecks für die Beschriftung soll sich dynamisch an die Breite der Beschriftung anpassen. Damit die unterschiedlichen Typen von Knoten zu unterscheiden sind, erhalten sie eine Füllfarbe nach dem folgenden Farbschema[125]:

Typ	CSS-Klasse	HEX-Farbcode	Farbe
Startseite	`.home`	`#EF8842`	
Kategorien	`.category`	`#E2032C`	
Allgemeines	`.generic`	`#81297B`	
Services / Dienstleistungen	`.services`	`#0053A1`	
Technologien	`.technology`	`#44AA44`	
Produkte	`.products`	`#F07F98`	
Referenzen	`.story`	`#777777`	

Tabelle 10: Knoten-Farbklassen

6.2 Definition der Anforderungen

6.2.1 Auto-Reset im Messebetrieb

Neben der Umsetzung anhand des visuellen Konzeptes sind weitere klare Anforderungen für die Umsetzung zu definieren. Da die Navigator-Anwendung vor allem für den Messestand entwickelt wurde, sind Funktionen erforderlich, die im Alltag des Messebetriebes notwendig sind. So muss sich die Darstellung bzw. Anwendung nach einer gewissen Zeit der Inaktivität wieder zurücksetzen, damit jeder neue Besucher des Messestandes gleiche Startvoraussetzungen hat. Das Zurücksetzen auf die Standard-Einstellungen umfasst folgende Anforderungen:

- Die Startseite der Website wird geladen und als aktiver Knoten angezeigt.
- Eventuell vorhandene Skalierungen (`scale`) werden zurückgesetzt.
- Eventuell vorhandene Verschiebungen (`translate`) werden zurückgesetzt.
- Aktivierte Filter und Stichwort-Suche werden zurückgesetzt.
- Umschalten auf die radiale Darstellung.

6.2.2 Responsivität

Da die Anwendung auf Geräten unterschiedlicher Auflösung und mit unterschiedlichen Eingabegeräten bedienbar sein soll, muss die Anwendung auch uneingeschränkt responsive sein und auf Änderungen des Viewports[126] reagieren:

- Darstellung passt sich der gegebenen Auflösung an.
- Änderung des Viewports führt zur Neuberechnung der Darstellung.
- Unterstützung von Maus-Steuerung und Multi-Touch.
  - Mausrad ermöglicht Steuerung der Skalierung.
  - Pinch-To-Zoom bewirkt Skalierung.
  - Hover-Effekt wird auf Touchscreens deaktiviert.
  - Alternativ Buttons zur Steuerung der Zoom-Stufe.

6.2.3 Ästhetik

Da die Anwendung überwiegend für den Messebetrieb gedacht ist, müssen auch die Faktoren Eyecatcher und Ästhetik gewürdigt werden, um ein Interesse bei vorbeigehenden Besuchern zu erreichen. Dazu soll nicht nur die statische Darstellung diesen Faktoren genügen, auch die Animation

[125] Die Farbproben sind unverbindlich aufgrund der unterschiedlichen Farbsysteme (Additiv ⇔ Subtraktiv).
[126] Größe des Browserinhaltes. Eine Änderung führt zum Auslösen des `window.resize`-Events.

der Benutzung soll entsprechend flüssig und stufenlos erfolgen. Letztendlich dient eine verfolgbare Bewegung auch der Benutzbarkeit, da der Nutzer das Geschehen besser einordnen kann.

6.2.4 No-JavaScript-Kompatiblität

Diese Anforderung bekommt eine große Gewichtung, besonders wenn die Anwendung als optionales Feature auf der Website zur Verfügung steht. Beim Messebetrieb hat man noch die Kontrolle, ob beim Endgerät JavaScript aktiviert ist. Aber im Betrieb als Website im Internet kann nicht sichergestellt werden, ob der Besucher JavaScript aktiviert hat. Somit muss die Website auch ohne Nutzung der interaktiven Navigation uneingeschränkt nutzbar bleiben, was ein Konzept für die verschiedenen Szenarien notwendig macht:

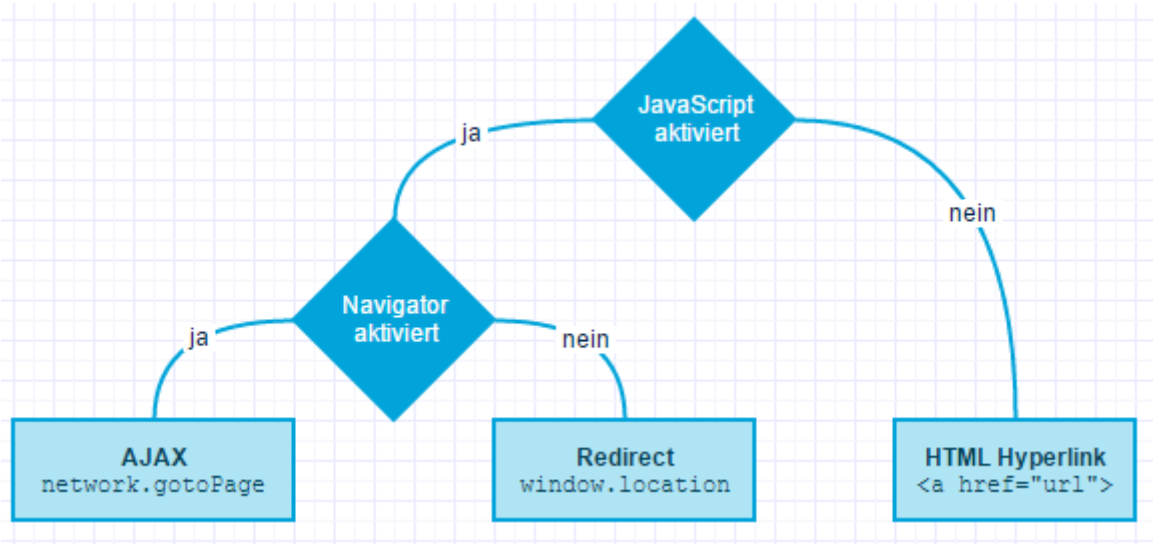


Abbildung 14: UML zur Fallentscheidung JavaScript

6.3 Ordner- und Dateistruktur

Als nächstes soll kurz die gewählte Ordner- und Dateistruktur erläutert werden. Alle Webseiten befinden sich in einem Unterordner, der nach dem sprachspezifischen Kürzel benannt ist. Im Image-Ordner im Hauptverzeichnis sind alle Bilder für die Webseiteninhalte gespeichert. Im Library-Verzeichnis sind alle strukturellen Dateien gespeichert, welche die Funktionen oder Erscheinung der Website beeinflussen. Dazu gehören Frameworks, Stylesheets, Template-Bilder (z. B. Logo der Firma) und die JavaScript-Funktionen. Navigator-spezifische JavaScript-Funktionen (`navigator-*.js`) sind von den Navigator-unspezifischen JavaScript-Funktionen (`scripts.js`) getrennt.

```
www.le-tex.de/
                de/
                      index.html
                      imprint.html
                      ...
                en/
                      index.html
                      imprint.html
                      ...
                img/
                      ...
                lib/
                      bootstrap/
                                  ...
                      d3/
                                  d3.min.js
                      images/
                                  ...
                      js/
                                  jquery.min.js
                                  scripts.js
                                  html5shiv.min.js
                                  respond.min.js
                      navigator/
                                  data/
                                        structure-en.json
                                        structure-de.json
                                  js/
                                        navigator-core.js
                                        navigator-run.js
                      stylesheets/
                                  ...
```

Abbildung 15: Ordner- und Dateistruktur

6.4 Herstellen der Querbeziehungen

Die Verbindungen werden aus den Relationen, die sich aus der Hierarchie der Website ergeben und aus den zu bildenden Querbeziehungen erzeugt. Um die Querbeziehung darzustellen, muss die vorhandene Seitenstruktur um eine Ebene erweitert werden, in der die Relationen zu anderen Seiten abseits der einfachen Eltern-Kind-Beziehung zugeordnet werden. Die *Abbildung 16: Erweiterte Sitemap mit Querbeziehungen* soll die Erweiterung mit der 4. Ebene verdeutlichen. Es sei anzumerken, dass nur der Pfad „Dienstleistungen/Verlagsherstellung“ ausführlicher dargestellt wird, was der Übersichtlichkeit geschuldet ist.

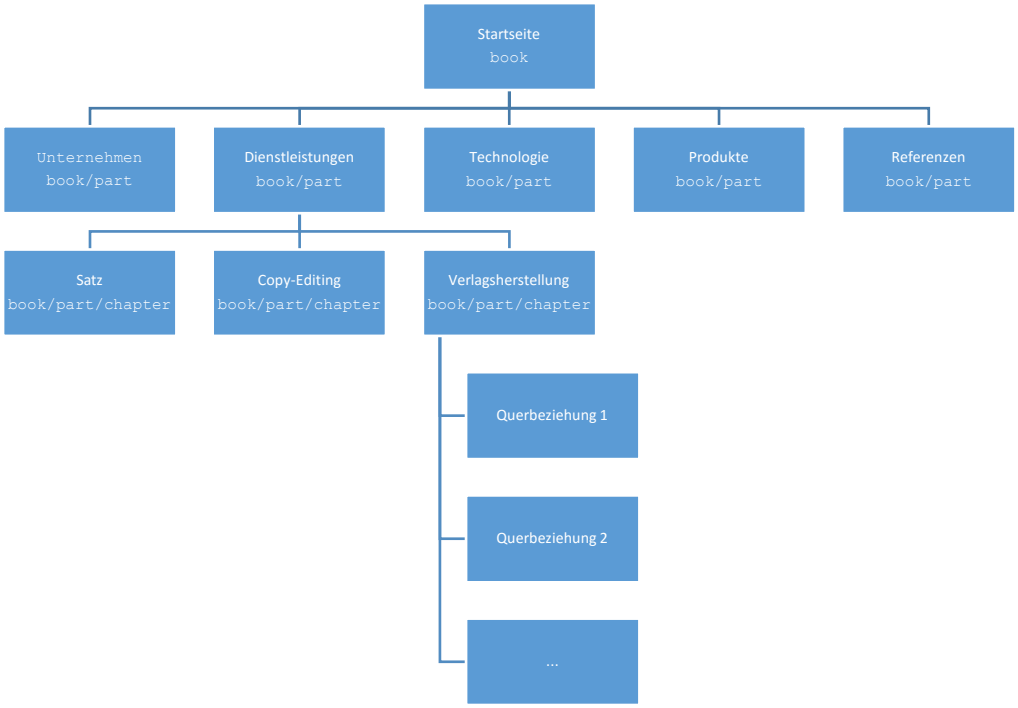


Abbildung 16: Erweiterte Sitemap mit Querbeziehungen

Zur 4. Ebene müssen nun Relationen gesucht werden, die in Verbindung zum aktuellen Knoten stehen. Dazu werden die in der DocBook-Quelle enthaltenen `<xref>`- und `<link>`-Elemente ausgewertet. Beide Elemente enthalten ein Attribut `linkend`, welches als Wert die `xml:id` des Zielknotens und damit die Referenz zur Zielseite beinhaltet.

Ansprechpartner

Die Seiten der einzelnen Ansprechpartner werden aus den Pfaden `book/info/authorgroup/author` bzw. `book/info/authorgroup/othercredit` generiert, in denen die Kontaktdaten, Beschreibungstexte und Bildreferenzen hinterlegt sind:

```
<sect1 xml:id="ebooks-contact">
  <title>Ansprechpartner</title>
  <para><xref linkend="Othercredit-MartinKraetke"/></para>
</sect1>
```

Listing 21: Beispiel für verlinkten Ansprechpartner

Eigene explizite Relation

Die DocBook-XML-Dateien enthalten bereits explizite Relationen, welche im Element `<info>` jeweils mit den Pfad `biblioreleation/link` gespeichert sind:

```
<info>
  <bibliorelation>
    <link linkend="epub"/>
  </bibliorelation>
  <bibliorelation>
    <link linkend="data-conversion"/>
  </bibliorelation>
</info>
```

Listing 22: Beispiel für explizite Relationen im DocBook

Andere explizite Relation

Die dritte Quelle für eine Querbeziehung sind andere Seiten bzw. Knoten, welche auf den aktuellen Knoten verweisen. Dazu muss überprüft werden, ob auf die aktuelle `xml:id` in einem anderen `bibliorelation/link`-Element verwiesen wird.

An beiden Beispielen für die expliziten Relationen wird deutlich, dass durch den Redakteur des DocBook-XML die Beziehung mittels des Elements `<bibliorelation>` hergestellt werden muss. Ohne mündigen und fachkundigen Redakteur sind Querbeziehungen in diesem Projekt nicht auf anderen Wegen herstellbar.

Zusammenfassend kann man sagen, dass die Querbeziehungen sich durch die in Relation stehenden Knoten in der 4. Ebene einer hierarchischen Struktur ergeben. Stellt man jetzt die Querbeziehungen eines Knotens dar, müssen nur der Eltern-Knoten und alle Kind-Knoten dargestellt werden. Folgend als Beispiel für den Knoten „Verlagsherstellung“ nach *Abbildung 16: Erweiterte Sitemap mit Querbeziehungen*:

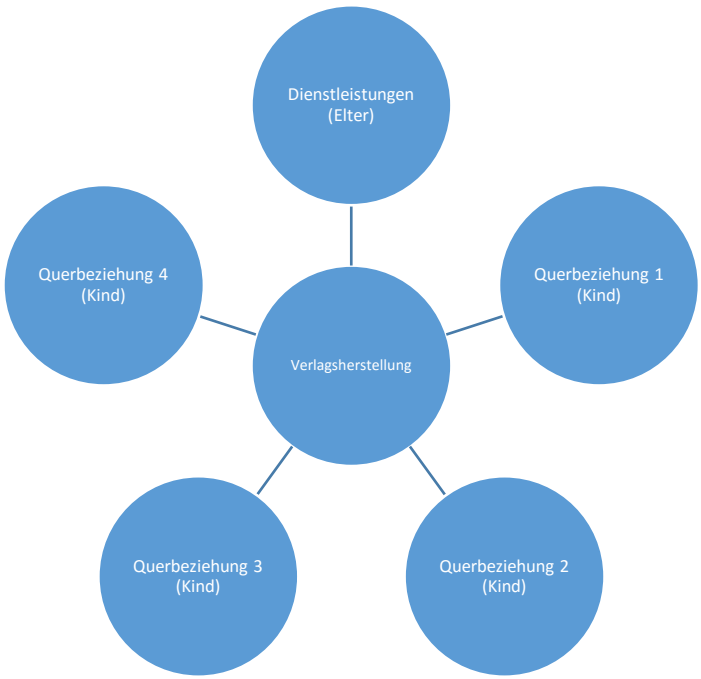


Abbildung 17: Alle Querbeziehungen eines Knotens

6.5 Erzeugen der JSON-Daten

Als Datenbasis für das Navigationsprogramm sollen zwei sprachspezifische JSON-Dateien dienen. Diese stellen einerseits die Querbeziehungen für D3.js dar und bieten auch zusätzliche Daten, die zur Navigation (Pfad wird unter dem Schlüssel `url` abgelegt) und Schlagwort-Suche (abgelegt unter dem Schlüssel `keywords`) notwendig sind. Für D3.js werden jeweils ein Array mit den Nodes, welche die

einzelnen Webseiten repräsentieren, als auch ein Array mit den Links benötigt. Für die Links muss nur auf die IDs der Quell- und Ziel-Nodes referenziert werden. Die einzelnen Seiten werden kategorisiert und die Kategorie wird im Schlüssel `type` gespeichert. Der gewählte Aufbau der JSON-Dateien[127]:

```
{
  "nodes":[
    {
      "id": "node-id als Referenz für die Links",
      "name": "node-title als Beschriftung für die Nodes",
      "type": "node-type für Filter und CSS-Farbklasse",
      "url": "node-html-url zum Laden der Seite",
      "keywords": "Liste mit Leerzeichen separierten Schlagwörtern"
    },
    ...
  ],
  "links":[
    {
      "source": "source-node-id",
      "target": "target-node-id"
    },
    ...
  ]
}
```

Listing 23: Struktur der JSON-Dateien

Für die Daten der Navigation wurde das JSON-Format gewählt, da es sich sehr einfach in ein JavaScript-Objekt parsen lässt. Für die Erzeugung der HTML-Dateien und des XML-Schemas als Sitemap[128] aus den DocBook-Quellen existiert bereits ein XSLT-Dokument, dass angepasst wird. Dazu muss als Output im XSLT ein weiteres Output-Format erstellt werden. Hier wird es `kiosk` genannt.

```
<xsl:output method="text" name="kiosk" />
```

Listing 24: Output-Format `kiosk` wird definiert

Für die Erzeugung der JSON-Dateien wird ein Template angelegt, welches auf dem Format-Typ `text` basiert und als Ziel (`result-document`) jeweils eine JSON-Datei ausgibt. Die Erzeugung der Links und Nodes ist in separaten Templates angegeben. In den Templates werden auch die geschweiften Klammern für das JSON-Objekt inkludiert.

Es wird zudem eine Variable `sitemap` definiert, die mittels einer Funktion `letex:generate-sitemap` eine Sitemap der DocBook-Struktur nach dem Schema aus *Abbildung 16: Erweiterte Sitemap mit Querbeziehungen* erstellt. Die Funktion erzeugt aus den DocBook-XML Elementen eine eigene XML-Struktur mit dem Wurzelelement `<sitemap>` und den Querbeziehungen in der 4. Ebene, die Grundlage für die weitere Verarbeitung sind.

Die Variable `lang` wird durch die zugrundeliegende sprachspezifische DocBook-Quelldatei in Form eines Sprachkürzels erzeugt.[129]

---

[127] Die drei Punkte sind nicht valide und sollen hier nur anzeigen, dass mehr als ein Eintrag im Array vorhanden sein wird. Beim letzten Objekt entfällt das Komma nach der geschweiften Klammer.
[128] Nach dem XML-Schema Version 0.9 von (sitemaps.org, 2008).
[129] Derzeit existiert nur eine deutsche (Kürzel: de) und eine englischsprachige Version (Kürzel: en) der Webseite.

```
<xsl:template name="kiosk-navigator">
  <xsl:variable
    name="sitemap"
    select="letex:generate-sitemap(/)"
    as="element(sitemap)"
  />
  <xsl:result-document
    href="lib/navigator/data/structure-{$lang}.json"
    format="kiosk">
    {
      <xsl:apply-templates select="$sitemap" mode="add-nodes" />
      <xsl:apply-templates select="$sitemap" mode="add-links" />
    }
  </xsl:result-document>
</xsl:template>
```

Listing 25: Template zur JSON-Erzeugung

6.5.1 Sitemap

In das Wurzelelement `<sitemap>` wird als einziges Child-Element der `index`-Knoten (`xml:id="index"`) aus dem DocBook-Preface gebildet. Als nächstes Child-Element dessen, dienen die einzelnen Parts (`book/part`) der DocBook-XMLs, welche die einzelnen Kategorien der Website bilden. Für sie wird in diesem Zusammenhang auch gleich der `type` auf dem Wert `category` gesetzt. Unter dieser Ebene befinden sich die einzelnen Seiten, die der Kategorie zugeordnet sind. Bis zu diesem Punkt spiegeln die Sitemap-XML-Daten die Hierarchie der Website-Struktur wieder. Die Querbeziehung werden jetzt als 4. Ebene zu den Seiten der 3. Ebene hergestellt.

```
<xsl:function name="letex:generate-sitemap" as="element(sitemap)">
  <xsl:param name="root-node" as="document-node(element(book))" />
  <sitemap xmlns="http://docbook.org/ns/docbook">
    <node
      origin="note_1" xml:id="index"
      label="{letex:label($root-node/book/preface[1])}"
      type="home" xmlns="http://docbook.org/ns/docbook">
      <xsl:for-each select="$root-node/book/part">
        <node origin="note_2" xml:id="{@xml:id}" label="{letex:label(.)}"
          relation="child" type="category"
          xmlns="http://docbook.org/ns/docbook">
          <xsl:for-each select="chapter">
            <node origin="note_3" xml:id="{@xml:id}"
              label="{letex:label(.)}" relation="child"
              type="{letex:node-type(.)}"
              xmlns="http://docbook.org/ns/docbook">
              <xsl:apply-templates select="letex:related(.)"
                mode="related-for-sitemap" />
            </node>
          </xsl:for-each>
        </node>
      </xsl:for-each>
    </node>
  </sitemap>
</xsl:function>
```

Listing 26: Funktion letex:generate-sitemap

Damit das Verarbeiten der Relationen der Ebene 4 durch die Funktion `letex:related` den Generierungsprozess der Website nicht unnötig ausbremst, wird ein Index für die Attribute `xml:id`

und `linkend` mit der Funktion `xsl:key`, zu Beginn des Templates erstellt.[130] Die Funktion `letex:related` greift dann auf diesen Index zurück. In diesem Zusammenhang soll auch kurz darauf hingewiesen werden, dass alle benutzerdefinierten Funktionen an den Namespace-Präfix `letex` gebunden sind. Alle Funktionen ohne dieses Präfix sind Bestandteil von XSLT.

```
<xsl:key name="by-id" match="*[@xml:id]" use="@xml:id" />
<xsl:key name="biblio-refers-to" match="*[@xml:id][info/bibliorelation]"
use="info/bibliorelation/link/@linkend" />
```

Listing 27: XSLT-Funktion xsl:key

Das Attribut `xml:id` ist in den sprachspezifischen DocBook-XMLs zu den einzelnen Elementen hinterlegt und bildet u. a. auch die Grundlage für die Benennung der einzelnen HTML-Dateien. Im Template werden weitere XSLT-Funktionen aufgerufen. Unter anderen eine Funktion[131], die einen Namen aus dem `<titel>`-, `<titleabbrev>`- oder `<personname>`-Element als String[132] zurückliefert. Hier wird deutlich, dass das DocBook-Format als Quellformat für Webseiten nicht ganz optimal ist und etwas zweckentfremdet wurde.

6.5.2 Nodes

Um die Einträge für die einzelnen Nodes zu erzeugen, werden zwei ineinander verschachtelte Templates benötigt. Die in der Variable `sitemap` und von der Funktion `letex:generate-sitemap` erzeugten XML-Daten werden anschließend durch ein Template verarbeitet.

```
<xsl:template match="sitemap" mode="add-nodes">
    "nodes":[
      <xsl:apply-templates mode="#current" />
    ],
</xsl:template>
```

Listing 28: Template zur Erzeugung des Node-Arrays

Für den einzelnen Node sind bereits nahezu alle Werte für die Schlüssel vorbereitet und können meist ohne Änderungen aus den Attributen übernommen werden. Die Schlüssel `id` und `url` werden um einen, eventuell noch vorhandenen, Präfix bereinigt.[133] Eine Schwierigkeit bestand darin, nach dem letzten Array-Eintrag kein Komma nach dem Objekt zu setzen. Die Prüfung, ob noch ein weiteres Element folgt, wurde durch einen bedingten Ausdruck in XPath gelöst.

[130] Vergleichbar etwa zu Primärschlüsseln in Datenbanken.
[131] Die Funktion `letex:label` befindet sich im Anhang unter *Listing 80: Funktion letex:*label.
[132] Datentyp `xs:string`.
[133] Durch die Funktion `letex:vertex-id`, siehe Anhang unter *Listing 81: Funktion letex:*vertex-id.

```
<xsl:template match="node" mode="add-nodes">
  <xsl:if test="not(preceding::node[@xml:id eq current()/@xml:id])">
    {
      "id": "<xsl:value-of select="letex:vertex-id(@xml:id)" />",
      "name": "<xsl:value-of select="@label" />",
      "type": "<xsl:value-of select="@type" />",
      "url": "../<xsl:value-of select="concat($lang, '/', letex:vertex-
id(@xml:id))" />.html",
      "keywords": "<xsl:value-of select="letex:generate-keywords(@xml:id,
@type)"/>"
    }<xsl:value-of select="if(not(not(following::node[not(@xml:id =
current()/preceding::node/@xml:id)]) and ../@xml:id != 'index')) then ','
else ''"/>
  </xsl:if>
  <xsl:apply-templates mode="#current" />
</xsl:template>
```

Listing 29: Template zur Erzeugung der Node-Objekte

6.5.3 Keywords

Zur Erzeugung der durch Leerzeichen separierten Stichwörter, wird in *Listing 29: Template zur Erzeugung* der Node- eine Funktion `letex:generate-keywords` aufgerufen. Die Keywords werden für die Stichwort-Suche benötigt und aus dem Inhaltstext gebildet. Der Text wird noch mit dem Titel ergänzt (`string-join`) und unnötiger Leerraum (`normalize-space`) und Satzzeichen[134] werden entfernt. Danach hilft `tokenize` mit dem regulären Ausdruck[135] „`\s+`“, die einzelnen Wörter zu separieren. Da die Suche nicht durch unnötige „kleine“ Wörter falsche Ergebnisse liefern soll, werden diese durch die Funktion `letex:remove-fillwords`[136] entfernt. Als letzter Schritt werden noch doppelte Einträge entfernt (`distinct-values`).

[134] Durch die Funktion `letex:jsonify`, siehe Anhang unter *Listing 85: Funktion letex:*jsonify.
[135] Oft als Regex oder RegExp abgekürzt.
[136] Die Funktion `letex:remove-fillwords` findet sich im Anhang unter *Listing 87: Funktion letex:*remove-fillwords.

```
<xsl:function name="letex:generate-keywords" as="xs:string">
  <xsl:param name="id" as="xs:string"/>
  <xsl:param name="type" as="xs:string"/>
  <xsl:value-of select="distinct-values(
                          letex:remove-fillwords(
                            tokenize(
                              letex:jsonify(
                                normalize-space(
                                  string-join(
                                    if($type eq 'category') then
                                      $root//*[@xml:id eq
$id]//text()[not(ancestor::title or ancestor::chapter)]
                                    else
                                      $root//*[@xml:id eq
$id]//text()[not(ancestor::title)]
                                    union
                                    $root//title[../@xml:id = $id]//text(),
                                    ''
                                  )
                                )
                              ),
                              '\s+'
                            )
                          )
                        )"/>
</xsl:function>
```

Listing 30: Funktion letex:generate-keywords

6.5.4 Links

Für die Einträge der einzelnen Links werden ebenfalls zwei ineinander verschachtelte Templates verwendet. Auch hier werden die XML-Daten aus der Funktion `letex:generate-sitemap` als Datenbasis verwendet. Wie bereits in *6.4 Herstellen der Querbeziehungen* erläutert, bestehen die Links aus den hierarchischen Verbindungen als auch aus den Querverbindungen, die als Beziehungen hierarchisch zwischen Ebene 3 und 4 in den XML-Daten der Sitemap gebildet wurden.

```
<xsl:template match="sitemap" mode="add-links">
    "links":[
      <xsl:apply-templates mode="#current" />
    ]
</xsl:template>
```

Listing 31: Template zur Erzeugung des Link-Arrays

```
<xsl:template match="node" mode="add-links">
  <xsl:if test="exists(parent::node) and not(preceding::node[@xml:id eq
current()/@xml:id][node[@xml:id eq current()/../@xml:id]])">
    {
      "source": "<xsl:value-of select="letex:vertex-id(../@xml:id)" />",
      "target": "<xsl:value-of select="letex:vertex-id(@xml:id)" />"
    }<xsl:value-of select="if(not(not(following::node) and ../@xml:id !=
'index')) then ',' else ''"/>
  </xsl:if>
  <xsl:apply-templates mode="#current" />
</xsl:template>
```

Listing 32: Template zur Erzeugung der Link-Objekte

6.6 JavaScript-Anwendung

Die gesamte JavaScript-Anwendung und der Code für deren Aufruf und Steuerung umfasst über 1000 Programmzeilen. Alle Programmzeilen zu kommentieren, würde den Rahmen der Arbeit überdehnen. Daher sollen einige exemplarische Problem- und Aufgabenstellungen in ihrem Lösungsansatz beschrieben werden.

6.6.1 Trennung Kernanwendung/Aufruf/Markup

Die JavaScript-Anwendung besteht aus 3 Teilen. Zum einen das Markup in den HTML-Dateien für die Klick-Events mittels `onclick`-Attribut, der Navigator-Kernanwendung `navigator-core.js` und den JavaScript-Funktionen in der `navigator-run.js` als Schnittstelle zwischen Darstellung (View) und der Kernanwendung (Controller). Mit den JSON-Dateien für die Datenbasis ist das Model-Element des MVC-Konzepts[137] bereits umgesetzt.

Funktion	Beschreibung
`$.fn.toggleNavigator()`	Ein- bzw. Ausschalten der interaktiven Navigation
`$.fn.startNavigator()`	Einschalten der interaktiven Navigation
`$.fn.stopNavigator()`	Ausschalten der interaktiven Navigation
`$.fn.toggleInternalLink()`	Zur internen Seite wechseln (AJAX bzw. Redirect)
`$.fn.switchLanguage()`	Umschalten der Sprache
`$.fn.restartDefaultTimer()`	Zurücksetzen des Reset-Timers

Tabelle 11: Zusätzliche jQuery Prototype-Funktionen

Das Markup zum Auslösen der Events ist in den HTML-Dateien zu sehen. Durch die `click`-Events werden Funktionen aufgerufen, welche als jQuery-Prototyp-Funktionen[138] (`$.fn.*`) angehängt sind. Dies ermöglicht auch den Aufruf dieser Funktion aus der Kernanwendung (`navigator-core.js`). Die im Markup hinterlegten Funktionsaufrufe betreffen ausschließlich den mit AJAX neuzuladenden Bestandteil der interaktiven Webseite und wurde gewählt, um Laufzeitprobleme mit dem Neusetzen der Event-Listener zu vermeiden. Man könnte das Problem sicherlich durch ein benutzerdefiniertes Event lösen, welches ausgelöst wird, sobald der Ladevorgang abgeschlossen wurde.

```
<div class="bookfold" onclick="$.fn.toggleNavigator(event)"></div>
```

Listing 33: Beispiel für onclick-Attribut zum Funktionsaufruf

[137] Das Muster Model View Controller wird in der Informatik oft angewandt, um eine Wiederverwendbarkeit und einfache Änderbarkeit eines Programms zu erleichtern.
[138] Der Aufruf `$.fn` ist äquivalent zum Aufruf `$.prototype`.

In der Datei `navigator-run.js` befinden sich zum einen alle Event-Listener (`click`, `resize`, usw.) der Navigator-Steuerung, als auch alle Funktionen, die zum Start- und Beenden der Navigator-Anwendung notwendig sind.

6.6.2 SVG-Darstellung

Bei ersten Versuchen hat sich herausgestellt, dass die Standard-Darstellung als Force-Directed Graph sich nur bedingt für Beziehungsdiagramme eignet, welche viele Querbeziehungen aufweisen und keine hierarchische Struktur hat. Die Querverbindungen scheinen den Algorithmus zur Berechnung der Positionen von Nodes dahingehend zu beeinflussen, je mehr Beziehungen ein Knoten besitzt. Es kommt also zu einer Art partieller Verdichtung („Klumpenbildung“). Man kann die Platzierung der Nodes zwar dahingehend beeinflussen, dass man die Eigenschaften des Force-Directed Graphs[139] so wählt, dass die Nodes ausreichend gut genug zu unterscheiden sind und sich nicht überlagern. Gleichzeitig wird verhindert, dass die Darstellung nicht eine unnötig große Dimensionierung einnimmt. Aber eine wirklich benutzerfreundliche Platzierung ist nur in der radialen Darstellung möglich. Dank der dynamischen Eigenschaften[140] kann man den aktiven Knoten mit mehr Raum und größerer Darstellung etwas hervorheben, was auch bei der radialen Darstellung genutzt wird.

Die Kernanwendung besteht aus zwei Funktionen, welche die Darstellung in einem zugewiesenen `div`-Container und mittels geparsten JSON-Daten erzeugt. Ein Funktions-Objekt `Network` enthält das gesamte dargestellt Netzwerk an Nodes und Links. Das zweite Funktions-Objekt `RadialPlacement` wird von der `Network`-Funktion aufgerufen, um die radiale Darstellung und Transformation, wie im Konzept vorgesehen, zu berechnen.

Um die Darstellung eines Nodes, wie in der Konzeption vorgesehen[141], zu erzeugen, werden in ein SVG-Gruppenelement `<g>` ein Kreis (`<circle>`), Rechtecke (`<rect>`) und Text (`<text>`) in Ebenen übereinandergelegt. Folgend ein Auszug aus der Funktion `updateNodes`[142]:

```
node.enter()
  .append("g")
    .attr("id", function(d) {
      return d.id;
    })
    .attr("class", function(d) {
      return "node " + d.type;
    })
  .append("circle")
    .attr("xlink:href", function(d) {
      return d.url;
    })
    .attr("r", function(d) {
      return nodeSize(d);
    });
node.exit().remove();
```

Listing 34: Darstellung eines Nodes mit D3.js (Auszug)

6.6.3 Handling interner Hyperlinks

Eine weitere Anforderung an die Anwendung ist, dass die Webseite bei deaktivierten JavaScript uneingeschränkt mit Verzicht auf die interaktive Navigation nutzbar bleibt. In *Abbildung 14: UML zur*

[139] Siehe *Tabelle 9: D3js Parameter für Atommodell*.
[140] Siehe *3.9.2 Dynamische Eigenschaften*.
[141] Siehe *Abbildung 13: Layout-Konzept für die Knoten*.
[142] Für die vollständige Funktion siehe Anhang unter *Listing 79: Funktion updateNodes*.

Fallentscheidung JavaScript wurden die drei Möglichkeiten bereits aufgeführt und sollen hier nun genauer erläutert werden.

1. Fall – JavaScript ist deaktiviert:

Da die internen Hyperlinks weiterhin auf den HTML-Seiten enthalten sind, sind bei deaktiviertem JavaScript keine Änderungen notwendig. Das `onclick`-Attribut wird vom Browser ignoriert und es wird zur entsprechenden internen Seite geleitet.

2. Fall – JavaScript ist aktiviert, aber der Navigator wurde nicht aktiviert:

Für den Fall, dass JavaScript aktiviert ist wird die Funktion ausgeführt, die im `onclick`-Attribut angegeben ist. Da verhindert werden soll, dass neben der Ausführung der Funktion der Browser auch dem Hyperlink folgt, muss außerdem der Wert `false` zurückgegeben werden. Das folgende Listing zeigt einen Auszug mit einem Template aus der XSLT-Datei, bei dem die internen Links angepasst wurden:

```
<a href="{$id}.html">
  <xsl:attribute name="class" select="'internal-link'" />
  <xsl:attribute name="data-link" select="$id" />
  <xsl:attribute name="onclick" select="'$.fn.toggleInternalLink(event);
return false;'" />
  <xsl:apply-templates/>
</a>
```

Listing 35: Angepasster interner Link im XSLT

Aus dem UML-Schema geht hervor, dass bei aktivierten JavaScript zwei Fälle zu unterscheiden sind, je nachdem ob der Navigator aktiv ist. Ist der Navigator nicht aktiv (`settings.active === false`), soll mittels JavaScript und der `window.location`-Funktion ein Redirect auf die interne Seite erfolgen. In der Funktion `$.fn.toggleInternalLink` sind noch zwei Fälle zu unterscheiden, die das `click`-Event ausgelöst hat. Es kann durch den Link `<a href="url">` selber erfolgt sein oder durch ein eingeschlossenes Element, wie etwa ein Bild. Dadurch muss jeweils unterschiedlich auf die Eigenschaften des Events zugegriffen werden. Während unter dem Schlüssel `target` bei einfachen Links die Schlüssel für das `href`- und `data`-Attribut liegen, liegen sie für den Fall von eingeschlossenen Elementen bei `currentTarget`.

```
$.fn.toggleInternalLink = function(event) {
  var pageID, href, anchor;

  if (event.currentTarget) {
    pageID = event.currentTarget.dataset.link;
    href = event.currentTarget.href;
  }
  else {
    pageID = event.target.dataset.link;
    href = event.target.href;
  }
  if (pageID.indexOf("#") !== -1) {
    anchor = pageID.substring((pageID.indexOf("#")));
    pageID = pageID.substring(0, (pageID.indexOf("#")));
  }
  else {
    anchor = "";
  }
  if (settings.active === true) {
    $.fn.restartDefaultTimer();
    return myNetwork.gotoPage(pageID, anchor);
  }
  else {
    window.location = href;
  }
};
```

Listing 36: Funktion $.fn.toggleInternalLink

3. Fall – JavaScript ist aktiviert, der Navigator ist auch aktiv:

Der dritte Fall, der zu behandeln ist, tritt ein, wenn JavaScript und Navigator aktiv sind. In diesem Fall wird der Anker-Link aus dem Wert des `data-link`-Attributes mittels `substring`-Funktion aus dem Seitenlink isoliert (Zeichenkette nach dem #-Zeichen). Auch die Seiten-ID wird aus dem Attribut extrahiert. Beide Werte werden beim Aufruf der `gotoPage`-Funktion (siehe *Listing 37: Methode network.gotoPage*) der Kernanwendung als Argumente übergeben. In der Funktion wird das Umschalten des aktiven Knotens aufgerufen[143] und der Browserverlauf erweitert[144].

```
network.gotoPage = function(pageID, anchor) {
  network.setPushState({ "id": pageID, "language": language, "anchor":
anchor });
  network.toggleActiveNode(pageID, anchor);
};
```

Listing 37: Methode network.gotoPage

In der Methode `network.toggleActiveNode` sind zwei wesentliche Funktionen enthalten: Die Aktualisierung der Darstellung im Navigator und das Laden des neuen Seiteninhaltes mit der jQuery-AJAX-Funktion in der Methode `network.loadPage`:

[143] Die Visualisierung wird aktualisiert. Siehe dazu im Anhang *Listing 67: Methode network.toggleActiveNode*.
[144] Die Manipulation des Browserverlaufs wird unter *6.6.4 Manipulation Browserverlauf* erläutert.

```
network.loadPage = function(urlToLoad) {
  return $.ajax({
    url: urlToLoad,
    dataType: "xml",
    success: function(response) {
      var innerContainer = $(response).find("#page-content").children();
      contentSelector.html(innerContainer);
    }
  });
};
```

Listing 38: Seiteninhalt mit AJAX laden

Bei der Methode wird mit der jQuery-Funktion `find`, der neue Seiteninhalt im DOM des Zieldokumentes gesucht (`find("#page-content").children`) und anschließend ersetzt dieser den bestehenden Seiteninhalt. Anschließend muss nur der Inhalt des `<title>`-Elements im `<head>` des Dokumentes aktualisiert werden.[145]

Zuletzt sollen auch Anker-Links benutzbar bleiben. Dieses Problem wurde durch Berechnung der Position des Ankers durch die Eigenschaft `offset().top` gelöst, in der die Position bis zum oberen Webseitenrand angegeben ist. Anschließend wird animiert zu der Position gescrollt. Die Methode `network.gotoAnchor` wird durch eine `setTimeout`-Funktion aufgerufen, um die Aufmerksamkeit des Benutzers gezielter zu steuern.[146]

```
network.gotoAnchor = function(anchor) {
  var position;
  position = $(anchor).offset().top;
  $("html body").animate({scrollTop: position}, 600);
};
```

Listing 39: Umsetzung von Anker-Links mit der Methode network.gotoAnchor

6.6.4 Manipulation Browserverlaufs

Um der Erwartungskonformität[147] der Browser-Schaltflächen zum Navigieren durch den Browser-Verlauf gerecht zu werden, muss beim 3. Fall, also bei aktiviertem Navigator, der Browserverlauf manuell gepflegt werden. Eine Änderung von Teilinhalten einer Webseite mit AJAX führt nicht zum Hinzufügen von Einträgen im Browserverlauf. JavaScript enthält aber die Funktion `history.pushState`, welche Einträge manuell in den Browserverlauf hinzufügen lässt und signalisiert das Klicken der Schaltflächen mittels `popstate`-Event.

Die Funktion history.pushState kann mit 3 Argumenten aufgerufen werden. Das erste Argument ist ein Objekt, welches dem Event-Objekt unter `history.state` angehängt wird, wenn das `popstate`-Event ausgelöst wird. Bei diesem Projekt wurde es verwendet, um die Seiten-ID und den Anker leichter zugänglich zu übertragen. Das zweite Argument kann ein Titel zum Status sein, welcher aber nicht von allen Browsern ausgewertet wird.[148] Das letzte Argument ist eine URL, zu der der Eintrag verweisen soll:

[145] Mittels der Methode in *Listing 68: Methode network.setPageTitle*.
[146] Als subjektiv optimales Intervall wurde eine halbe Sekunde, also 500 Millisekunden (ms) gewählt.
[147] Eine Schaltfläche muss sich so verhalten, wie vom Benutzer aufgrund von Erfahrungen erwartet wird (nach ISO 9241).
[148] Laut (MDN - Mozilla Developer Network, 2015) ignoriert z. B. Firefox den Title.

```
history.pushState(statusObjekt, "Title", "URL");
```

Listing 40: history.pushState

Wie in *Listing 37: Methode network.gotoPage* zu sehen, wird beim Umschalten der aktiven Seite auch die Methode `network.setPushState` aufgerufen. Diese Methode fügt dem Browserverlauf das oben beschriebe Objekt als erstes Argument, einen leeren String als zweites Argument und die URL als drittes Argument hinzu. Auch die Anker-Links werden hinzugefügt, um diese beim Navigieren mit den Browser-Schaltflächen verarbeiten zu können.

```
network.setPushState = function(pushObject) {
  if (pushObject.anchor) {
    history.pushState(pushObject, "", pushObject.id + ".html" +
pushObject.anchor);
  }
  else {
    history.pushState(pushObject, "", pushObject.id + ".html");
  }
};
```

Listing 41: Methode network.setPushState

Um das Umschalten des aktiven Knotens durch die Vor- und Zurück-Schaltflächen des Browsers zu erreichen, muss man nur noch einen Event-Listener auf das `popstate`-Event setzen und die Methode zum Umschalten des aktiven Knotens aufrufen:

```
window.onpopstate = function(event) {
  if (event.state.id) {
    return myNetwork.toggleActiveNode(event.state.id, event.state.anchor);
  }
};
```

Listing 42: popstate-Event-Listener

6.6.5 Responsivität

Damit die Darstellung des SVG-Containers sich an die Größe des Viewports anpasst, muss er beim Start des Navigators als auch bei jedem `resize`-Event neu dimensioniert werden. Dazu wird ein Event-Listener auf das `window.resize`-Event gesetzt:

```
$(window).on("resize", function() {
  if (settings.active === true) {
    return myNetwork.toggleSize();
  }
});
```

Listing 43: resize-Event-Listener

Wird das `resize`-Event ausgelöst, wird eine Neuberechnung der SVG-Darstellung mittels der Funktion `network.toggleSize` in der Kernanwendung ausgeführt. Dabei werden die Dimension des Force-Directed Graphs[149] (`force.size`), die Größe der Nodes (`network.toggleNodeRadius`), die Größe des Containers (`#page-container`) als auch die Zoom-Stufen (`network.resetZoom`) angepasst:

[149] Auch bei radialer Darstellung.

```
network.toggleSize = function() {
  var newWidth, newHeight;
  newWidth = navigationSelector.clientWidth;
  newHeight = window.innerHeight;
  force.stop();
  force.size([newWidth, newHeight]);
  setWidth(newWidth);
  setHeight(newHeight);
  network.toggleNodeRadius();
  network.resetZoom();
  zoom.size([width, height]);
  d3.select("#page-container").attr("height", height);
  return update();
};
```

Listing 44: Update der Darstellung bei Änderung der Viewport-Größe

Die unter *6.2.2 Responsivität* auf Seite 40 gestellte Anforderung, die `hover`-Funktion bei Touchscreens zu deaktivieren, konnte nicht sicher gelöst werden. Zwar kann man mit der JavaScript-Bibliothek Modernizr[150], Features vom Browser abfragen, aber die Funktion `Modernizr.touch` liefert nicht durchweg sicher zurück, ob das Endgerät mit einem Touchscreen ausgestattet ist oder ob der Browser nur die Touchscreen-API, das heißt Touch-Events, unterstützt.

```
if (Modernizr.touch) {
    alert('Ist Touchscreen');
} else {
    alert('Kein Touchscreen');
}
```

Listing 45: Modernizr Touch-Event-Erkennung

So gibt es Fehlerkennungen in beide Szenarien. Es können Touch-Geräte erkannt werden, welche keinen Touchscreen besitzen, oder eben werden keine Touch-Geräte erkannt, obwohl sie mit einem Touchscreen ausgerüstet sind.[151] Als Ergebnis der Versuche wurde die Implementierung gestrichen und nur im Messebetrieb wird die `hover`-Funktion explizit deaktiviert.

6.6.6 Nachbar-Nodes isolieren

Die aktuellen Nachbar-Knoten eines aktiven oder per `hover`-Funktion berührten Knotens, müssen aus der Gesamtzahl von Knoten isoliert werden. Die radiale Darstellung stellt die direkten Nachbar-Knoten auf einer näheren konzentrischen Kreisbahn dar, als die übrigen Knoten. Um die Nachbar-Knoten und Verbindungen zu isolieren, die bei einem selektierten Knoten in Bezug stehen, muss man zuerst die Links suchen, in denen die ID des ausgewählten Knotens referenziert ist:

[150] (Modernizr, 2016).
[151] Vgl. dazu die Diskussion zum Issue #548 des Github-Repositorys von Modernizr unter (GitHub - Modernizr, 2012).

```
filterConnectedLinks = function(nodeID, links) {
  var filteredLinks = [];
  links.forEach(function(d, i) {
    if (links[i].source.id === nodeID || links[i].target.id === nodeID) {
      filteredLinks.push(links[i]);
    }
  });
  return filteredLinks;
};
```

Listing 46: Isolieren der Links eines Nodes

Anschließend kann man anhand der gefilterten Links die angeschlossenen Knoten (mit Ausnahme des aktiven Knoten) ebenfalls filtern:

```
filterConnectedNodes = function(nodeID, links) {
  var filteredNodes = [];
  links.forEach(function(d, i) {
    if (links[i].source.id !== nodeID) {
      filteredNodes.push(links[i].source.id);
    }
    else if (links[i].target.id !== nodeID) {
      filteredNodes.push(links[i].target.id);
    }
  });
  return filteredNodes;
};
```

Listing 47: Isolieren der Nachbar-Nodes eines Nodes

6.6.7 Radiale Darstellung

Um die radiale Darstellung umzusetzen, benötigt man eine Funktion, welche die x- und y-Koordinaten aller Knoten berechnet. Dazu wurde der globalen Variable `RadialPlacement` eine Funktion zugewiesen. Die Berechnung der Positionen auf den beiden konzentrischen Kreisbahnen erfolgt mit Winkelfunktionen des JavaScript `Math`-Objektes. Das `Math`-Objekt enthält z. B. die Eigenschaft `Math.pi` mit der die mathematischen Konstante π als Wert. Die Methoden `Math.cos` und `Math.sin` ermöglichen dann die Berechnung von Winkeln mittels der Cosinus- und Sinus-Funktion:

```
radialLocation = function(center, angle, radius) {
  var x, y;
  x = center.x + radius * Math.cos(angle * Math.PI / 180);
  y = center.y + radius * Math.sin(angle * Math.PI / 180);
  return {
    "x": x,
    "y": y
  };
};
```

Listing 48: Funktion radialLocation

Für die inneren und äußeren Kreisbahnen müssen die entsprechenden Knoten erst in Nachbarn und Nicht-Nachbarn getrennt werden.[152] Die Nachbarn werden dann der inneren Kreisbahn[153] und die Nicht-Nachbarn[154] der äußeren Kreisbahn übergeben.

Um die Bewegungen der radialen Nachbarschafts-Darstellungen zu berechnen, wird eine eigene Handlerfunktion für die Koordinaten an das `tick`-Event übergeben.[155] Die Funktion `neighbourTick` macht sich für die Bewegungen die Eigenschaft `alpha` von D3.js zunutze:

```
neighbourTick = function(event) {
  node.each(moveToNeighbourLayout(event.alpha));
  node.attr("transform", function(d) {
      return "translate(" + d.x + "," + d.y + ")";
  });
  if (event.alpha < .03) {
    force.stop();
    return updateLinks();
  }
};
```

Listing 49: Funktion neighbourTick

Die Berechnung der Bewegung bis zur Endposition erfolgt für den aktiven Knoten und die übrigen Knoten getrennt, durch die Funktion `moveToNeighbourLayout`. Der Faktor `k` beeinflusst die Schnelligkeit der Bewegung zur Endposition:

```
moveToNeighbourLayout = function(alpha) {
  var k;
  k = alpha * .25;
  return function(d) {
    var centerNode;
    var offset = document.getElementById("vis-control").offsetHeight;
    if (d.id === activeNodeID) {
      centerNode = {"x": width/2, "y": (height/2) + offset};
      d.x += (centerNode.x - d.x) * k;
      d.y += (centerNode.y - d.y) * k;
    }
    else {
      centerNode = groupCenters(d.id);
      d.x += (centerNode.x - d.x) * k;
      d.y += (centerNode.y - d.y) * k;
    }
  };
};
```

Listing 50: Funktion moveToNeighbourLayout

6.6.8 Skalierung und Verschiebung

Da der Umfang des Navigator-Inhaltes trotz großem Bildschirm kaum komplett dargestellt werden kann, wurde als Anforderung die Möglichkeit gestellt, die Skalierung ändern zu können. D3.js bringt

[152] Siehe dazu im Anhang die Funktionen in *Listing 70: Funktion filterNeighbours* und *Listing 71: Funktion filterNotNeighbours*.
[153] Siehe Anhang *Listing 72: Funktion setInnerKeys*.
[154] Siehe Anhang *Listing 73: Funktion setOuterKeys*.
[155] Siehe dazu im Anhang *Listing 69: Funktion zum Umschalten der Layouts*.

bereits entsprechende Methoden mit, welche die Maus- und Touch-Events unterstützten.[156] Dabei werden nicht nur Änderungen an der Skalierung durch das Mausrad bzw. Pinch-To-Zoom[157] erfasst, sondern auch Verschiebungen.[158] Um die Methode `d3.behavior.zoom` aufzurufen, wird sie einer Variable zugewiesen:

```
var zoom;
zoom = d3.behavior.zoom();
```

Listing 51: Instanziieren von d3.behavior.zoom

In der Methode `network` muss nun ein Event-Listener auf das D3-Event `zoom` und auf die Selektion des SVG-Containers gesetzt werden. Gleichzeitig kann man Skalierungsgrenzen mit der Eigenschaft `scaleExtent` setzen:

```
zoom.scaleExtent([.1, 5]).on("zoom", network.toggleZoom).size([width,
height]);
container.call(zoom);
```

Listing 52: Event-Listener für das zoom-Event

Das genaue Verhalten beim `zoom`-Event wird durch die eigene Methode `network.toggleZoom` berechnet. Auf die Werte für `scale` und `translate` lässt sich über das Event-Objekt zugreifen.

```
network.toggleZoom = function() {
  var trans, scale;
  trans = d3.event.translate;
  scale = d3.event.scale;
  zoom.translate(trans);
  d3.select("svg").select("g")
    .attr("transform", "translate(" + trans + ")scale(" + scale + ")");
};
```

Listing 53: Methode network.toggleZoom

Um beim `resize`-Event die Skalierung anzupassen, wird wie in *Listing 44: Update der Darstellung bei Änderung der Viewport-Größe* bereits zu sehen war, mit der Eigenschaft `size`, die Größe des Zoom-Verhaltens neu gesetzt und eventuelle Skalierungen mit der Methode `network.resetZoom` zurückgesetzt.[159]

Für die Steuerung der Skalierung über die Schaltflächen der Navigator-Steuerung sind einige eigene Funktionen zur Transformation notwendig, da hier kein `zoom`-Event-Objekt mangels `zoom`-Events existiert. Die Methoden `network.zoomIn`[160] und `network.zoomOut`[161] werden dann über das `click`-Event der Buttons aufgerufen und führen eine manuelle Berechnung durch.[162] Die Skalierung

[156] Eine detaillierte Referenz findet sich unter (GitHub - d3, 2015).
[157] Durch Spreizen oder Zusammenschieben zweier auf dem Touch-Bildschirm aufliegender Finger, bewirkt das Vergrößern bzw. Verkleinern eines Objektes.
[158] Als Referenz-Mittelpunkt wird dabei die Position des Mauszeigers, bzw. der Mittelpunkt der beiden Finger verwendet.
[159] Die Methode ist im Anhang unter *Listing 74: Methode network.resetZoom* zu finden.
[160] Siehe Anhang *Listing 75: Methode network.zoomIn*.
[161] Siehe Anhang *Listing 76: Methode network.zoomOut*.
[162] Dafür sind die im Anhang zu findenden Methoden `network.scaleTranslate` (siehe *Listing 77: Methode network.scaleTranslate*) und `network.calculateTranslate` (siehe *Listing 78: Methode network.calculateTranslate*) notwendig.

ist dabei auf den zentralen Mittelpunkt beschränkt und nicht auf das sich aktuell in der Mitte befindende Objekt.

6.6.9 Zeitgesteuerte Autodefault-Funktion für den Messestand-Betrieb

Beim Messebetrieb sollen die Einstellungen und Darstellung nach einem bestimmten Intervall der Inaktivität auf die Initialwerte zurückgesetzt werden. Das Intervall ist dabei ausreichend lang zu wählen, dass nicht beim Lesen einer Seite versehentlich das Zurücksetzen ausgelöst wird. Andererseits sollte es nicht zu lang gewählt werden, da ein neuer Besucher eine Standard-Einstellung vorfinden soll.

Um nicht alle Einstellungen für jedes Event neu definieren zu müssen, wird ein Funktionsausdruck definiert, welcher alle Standard-Einstellungen enthält und zugewiesen werden kann. Die Funktion stellt eine Schnittstelle zwischen Darstellung und Kernanwendung dar, da nicht nur die SVG-Visualisierung zurückgesetzt werden muss, sondern auch alle Schaltflächen.

```
defaultSettings = function() {
  $("#layouts .active").removeClass("active");
  $("#layouts [data-view='neighbours']").addClass("active");
  $("#filters .active").removeClass("active");
  $("#filters [data-filter='all']").addClass("active");
  myNetwork.resetToDefaults();
  $("#search").val("");
  myNetwork.updateSearch("");
  myNetwork.resetZoom();
};
```

Listing 54: Funktion defaultSettings für Navigator-Defaults

Wird der Navigator im Messebetrieb gestartet, muss erstmalig eine `setTimeout`-Funktion auf den Funktionsausdruck `autoDefault` zugewiesen werden. Dazu wurde die Funktion in *Listing 66: Funktion $.fn.startNavigator zum Starten des Navigators* erweitert:

```
if (settings.kioskMode === true) {
  autoDefault = setTimeout(function() {
      defaultSettings();
  }, settings.resetTimer);
}
```

Listing 55: Initiales Setzen des Default-Timers

Zum Zurücksetzen einer `setTimeout`-Funktion muss diese erst mit der Funktion `clearTimeout` gelöscht und anschließend neu zugewiesen werden:

```
$.fn.restartDefaultTimer = function() {
  if (settings.kioskMode === true) {
    clearTimeout(autoDefault);
    autoDefault = setTimeout(function() {
      defaultSettings();
    }, settings.resetTimer);
  }
};
```

Listing 56: Funktion $.fn.restartDefaultTimer zum Zurücksetzen des Default-Timers

Damit bei jeder Aktion auf den Bildschirm der Timer zurückgesetzt wird, muss auf alle Events die Handlerfunktion `$.fn.restartDefaultTimer` zum Zurücksetzen des Timers zugewiesen werden.

```
$(window).on("click", function() {
  $.fn.restartDefaultTimer();
});
$(window).on("mousemove", function() {
  $.fn.restartDefaultTimer();
});
$(window).on("touchmove", function() {
  $.fn.restartDefaultTimer();
});
```

Listing 57: Events zum Zurücksetzen des Default-Timers

6.6.10 Schlagwort-Suche

Die Schlagwort-Suche, mit der nach Webseiten gesucht werden kann, welche die entsprechenden Schlagwörter enthalten, besteht aus zwei Komponenten. Der Methode `network.updateSearch` in der Kernanwendung und den JavaScript-Aufruf mit Event-Listener auf das Suchen-Formularfeld. Das Event `keyup` wird ausgelöst sobald sich der Inhalt im Formularfeld ändert. Ist eine Eingabe erfolgt, wird auch die Klasse `active` zur Formulargruppe hinzugefügt, welche das Einblenden des Löschen-Buttons steuert.

```
$("#search").keyup(function() {
  var searchTerm;
  searchTerm = $(this).val();
  if (searchTerm.length > 0) {
    $(this).parent().addClass("active");
  }
  else {
    $(this).parent().removeClass("active");
  }
  return myNetwork.updateSearch(searchTerm);
});
```

Listing 58: Event-Listener für das keyup-Event

Die Suche selbst erfolgt mittels eines regulären Ausdrucks, der case insensitive[163] mit der Eingabe aus dem Formular auf die `keywords`-Eigenschaft der Nodes angewendet wird. Um unerwünschte Effekte bei aktivierter Suche und Hovern der Nachbarn zu vermeiden, wird die `hover`-Funktion in diesem Fall solange deaktiviert:

[163] Groß- und Kleinschreibung wird nicht beachtet.

```
network.updateSearch = function(searchTerm) {
  if (searchTerm.length > 0) {
    setShowNeighbours(false);
  }
  else {
    setShowNeighbours(true);
  }
  var searchRegEx;
  searchRegEx = new RegExp(searchTerm.toLowerCase());
  return node.each(function(d) {
    var element, match;
    element = d3.select(this);
    match = d.keywords.toLowerCase().search(searchRegEx);
    if (searchTerm.length > 0 && match >= 0) {
      element.style("fill-opacity", 1)
        .select("circle").transition()
          .style("stroke-width", 5)
          .attr("r", nodeRadius*1.5);
      return d.searched = true;
    }
    else if (searchTerm.length === 0) {
      element.style("fill-opacity", 1)
        .select("circle")
          .style("stroke-width", 1)
          .attr("r", nodeSize(d));
       return d.searched = false;
    }
    else {
      element.style("fill-opacity", .3)
        .select("circle")
          .style("stroke-width", 1)
          .attr("r", nodeSize(d));
       return d.searched = false;
    }
  });
};
```

Listing 59: Methode network.updateSearch

Die Funktion bietet noch etwas Optimierungspotential bezüglich der Trennung von Darstellung und Funktion. Wobei sich der Radius nicht über CSS beeinflussen lässt.

Auf den eigentlich passenden Input-Typ `search` wurde verzichtet, da er nicht in allen Browsern gleich unterstützt wird und somit nicht in jedem Browser das Löschen-Kreuz erscheint. Es musste auf eigenes HTML-Markup für einen Löschen-Button zurückgegriffen werden. Ohne diesen Button hätte sonst die Benutzerfreundlichkeit gelitten.[164]

Abbildung 18: Screenshot der Schlagwort-Suche mit Reset-Button

[164] Der Nutzer muss andernfalls manuell alle Zeichen entfernen, um die Suche dann zu deaktivieren.

Zum Entfernen des Suchfilters, wird auf den Löschen-Button ein Event-Listener gesetzt:

```
$("#clear-search").click(function() {
  $("#search").val('');
  $(this).parent().parent().removeClass("active");
  return myNetwork.updateSearch('');
});
```

Listing 60: Event-Listener zum Zurücksetzen der Schlagwort-Suche

6.7 Modifikation der HTML-Dateien

Zur Steuerung der Navigator-Anwendung sind einige Modifikationen an der Website notwendig. Zunächst muss das Template für die Einbindung der JavaScript-Dateien angepasst werden.[165] Die Variable `resourcepath` wurde angelegt, um schneller Änderung an der Verzeichnisstruktur vornehmen zu können. Sie verweist derzeit auf den Pfad `www.le-tex.de/lib`. Die CSS-Definitionen für die Navigator-Elemente werden mit in die vorhandenen Stylesheets eingepflegt. Im Rahmen einer besseren Modularisierung der Navigator-Anwendung könnte man sie auch in einer separaten Stylesheet-Datei auslagern.

```
<xsl:template name="scriptsbody">
  <script src="{$resourcepath}/js/jquery.min.js" />
  <script src="{$resourcepath}/bootstrap/js/bootstrap.min.js" />
  <script src="{$resourcepath}/d3/d3.min.js" />
  <script src="{$resourcepath}/navigator/js/navigator-core.js" />
  <script src="{$resourcepath}/navigator/js/navigator-run.js" />
  <script src="{$resourcepath}/js/scripts.js" />
</xsl:template>
```

Listing 61: Template zur Einbindung der Scripts

Dem <`html`>-Element wird bei der Generierung der Website für den Messestand-Betrieb eine Klasse `kiosk-mode` zugewiesen, welche u. a. den `hover`-Effekt deaktiviert, um unerwünschte Effekte bei Touchscreens[166] zu vermeiden.

```
<xsl:if test="$externallinks = 'no'">
  <xsl:attribute name="class" select="'kiosk-mode'"/>
</xsl:if>
```

Listing 62: Klasse „kiosk-mode“ für Messestand-Betrieb

Auch wird in diesem Fall die Möglichkeit entfernt, den Navigator abzuschalten, indem die `onclick`-Attribute im Markup zum Umschalten des Navigators nicht hinzugefügt werden.

[165] Im Rahmen eines Praktikums und der Umgestaltung der Website auf ein responsives Layout, wurden zentrale Templates angelegt um Modifikationen zu vereinfachen.
[166] Manche Touchscreens lösen den Hover-Effekt durch unklares Tippen aus.

```
<xsl:choose>
  <xsl:when test="$externallinks = 'yes' ">
    <div class="bookfold" onclick="$.fn.toggleNavigator(event)"><xsl:text>
</xsl:text></div>
  </xsl:when>
  <xsl:otherwise>
    <div class="bookfold"><xsl:text> </xsl:text></div>
  </xsl:otherwise>
</xsl:choose>
```

Listing 63: Onclick-Event zum Umschalten des Navigators

Als nächsten Schritt muss, wie im Konzept vorgesehen, die Webseite zweigeteilt in eine Side-By-Side-Darstellung bzw. Overlay-Darstellung überführt werden. Die Kontrolle, wann der Navigator eingeblendet wird, erfolgt durch Setzen von Klassen und CSS-Definitionen mit Media Queries.

Wie im *Listing 64: Template zur Aufteilung der Webseite für Navigator und Inhaltsteil* zu sehen ist, wurde das HTML5-Element `<aside>` gewählt, um den Navigator einzuschließen. Sollte die Navigation eine Überarbeitung unter dem Aspekt der Barrierefreiheit erfahren haben, wäre dann eine Umstellung auf ein `<nav>`-Element sinnvoller. Dann wäre sie für eingeschränkte Besucher als Navigationsinstrument zu identifizieren. Auf die `role`-Attribute wird hier verzichtet, da die Tags bereits die `role`-Attribute implizieren.

```
<div id="page-frame" class="container">
  <aside id="page-navigation">
    <div id="vis">
    </div>
    <div id="vis-control">
      <xsl:call-template name="kiosk-controls" />
    </div>
    <div id="navigator-close" onclick="$.fn.toggleNavigator(event)"
role="button">
      <xsl:choose>
        <xsl:when test="$lang = 'de'">
          <p><span class="glyphicon glyphicon-remove-circle" aria-
hidden="true"></span> Interaktive Navigation schließen</p>
        </xsl:when>
        <xsl:when test="$lang = 'en'">
          <p><span class="glyphicon glyphicon-remove-circle" aria-
hidden="true"></span> Close interactive navigator </p>
        </xsl:when>
      </xsl:choose>
    </div>
  </aside>
  <section id="page-content">
    <!-- Page Content -->
  </section>
</div>
```

Listing 64: Template zur Aufteilung der Webseite für Navigator und Inhaltsteil

Das Template für die Steuerungselemente der Navigation mit den Filtern, der Stichwort-Suche und der Darstellungssteuerungselemente befindet sich im Anhang unter *Listing 88: Template für die Navigator-Steuerung*. Das Template besteht aus zwei sprachspezifischen Varianten, welche Bestandteil jeder einzelnen Webseite sind. Man könnte als Lösung auch eine dynamische Beschriftung umsetzen. Da es derzeit nur zwei sprachenspezifische Versionen gibt, wurde aber darauf verzichtet. Für die Steuerungselemente wurden überwiegend Elemente aus dem

Bootstrap-Framework verwendet. Daten die zur Steuerung der Navigator-Anwendung notwendig sind, wurden in den `data`-Attributen abgelegt.

6.8 CSS-Styles

Zur Umsetzung des visuellen Konzeptes und des in *Tabelle 10: Knoten-Farbklassen* definierten Farbschemas sind einige Anpassungen der Datei `default.css` notwendig. Das SVG-Gruppierungselement, welches einen Knoten repräsentiert und bündelt, trägt neben der eindeutigen ID auch eine Klasse mit der Bezeichnung des Typs der Nodes. Anhand dieser lassen sich die Farben einfach mit der Eigenschaft `fill` im CSS definieren und eine Trennung zwischen Markup und Styling erreichen.

```
.node.home circle,
.node.home .front-rect {
  fill: #ef8842;
}
.node.category circle,
.node.category .front-rect {
  fill: #e2032c;
}
.node.generic circle,
.node.generic .front-rect {
  fill: #81297B;
}
.node.services circle,
.node.services .front-rect {
  fill: #0053a1;
}
.node.technology circle,
.node.technology .front-rect {
  fill: #44aa44;
}
.node.story circle,
.node.story .front-rect {
  fill: #777;
}
.node.products circle,
.node.products .front-rect {
  fill: #f07f98;
}
```

Listing 65: Definition der Node-Farben im CSS

Die CSS-Klassen entsprechen auch den Klassen für den Typ-Filter. Bei der Vorstellung des Projektstandes bei der firmeninternen Entwicklerrunde, wurde der Vorschlag unterbreitet, die Buttons mit einer Art Legende für die Filter bzw. Knoten zu versehen. Dieser Vorschlag wurde angenommen und umgesetzt, somit tragen die Buttons für die Filter auch eine kleine Farbreferenz, um die Knoten besser zuordnen zu können.[167] Eine kleine Schwierigkeit bestand in der Beschriftung dieser Buttons. Für diese fanden sich keine eindeutigen Symbole im Font Awesome, bzw. die Symbole des Startseiten-Menüs der Unternehmens-Website schienen ebenfalls nicht eindeutig

[167] Die CSS-Definitionen lassen sich im Anhang unter *Listing 89: Farbreferenz für die Filter-Buttons* begutachten.

genug zu sein. Die Text-Bezeichnungen aber waren für einige Bildschirmgrößen zu lang[168] und somit fand sich eine Lösung mit der CSS-Eigenschaft `text-overflow` und dem sicherlich wenig bekannten Wert `ellipsis`:

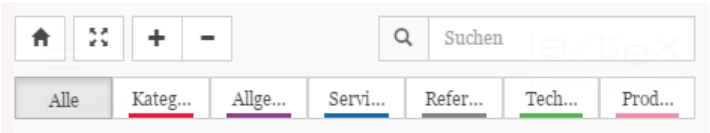
Visuelles Ergebnis	CSS-Code
	`#filters label { overflow: hidden; text-overflow: ellipsis; }`

Tabelle 12: text-overflow: ellipsis

Dank des Wertes `ellipsis` wird der Text automatisch eingekürzt und das Ausgelassene durch drei Auslassungspunkte ersetzt. In *Tabelle 12: text-overflow: ellipsis* ist auch das Ergebnis der Farblegende für die Filter zu erkennen.

[168] Man kann zwar die Buttons in eine Spaltendarstellung umbrechen lassen, aber dann würde das Navigator-Steuerungs-Menü einen Großteil des vorhandenen Viewports benötigen. Die Filter-Option wird besonders für kleine Bildschirme als wichtig erachtet und sollte daher auch nicht weggelassen werden. Hier handelt es sich also um ein responsives Dilemma ;-).

7 Auswertung

So wie viele Projekte, offenbart auch dieses Projekt mit etwas zeitlichem Abstand, einen ungetrübten Blick auf Aspekte, welche sich optimieren lassen. Spätestens wenn die Navigator-Anwendung bei einem weiteren Projekt zum Einsatz kommen soll, ist eine weitere Modularisierung obligatorisch. Dann müssen z. B. die `click`-Events vom HTML-Markup getrennt werden. In diesem Zusammenhang könnte die gesamte Anwendung auf GitHub gehostet und als externes Repository im Build-Prozess eingebunden werden.

Wenn an dieser Stelle von Modularisierung und weiteren Einsatzszenarien gesprochen wird, so muss sofort DITA[169] Map genannt werden. Mittels DITA Map lassen sich Querbeziehungen durch das `<topicref>`-Element herstellen. Diese Beziehungen können dann nicht nur dazu verwendet werden, um Table of Contents (TOC) aus den DITA Topics zu generieren. Sie können auch beim Erzeugen einer Website aus einem DITA-Dokument für die interaktive Navigation genutzt werden.

Ein großes Feld für Optimierungen ist die Barrierefreiheit der Anwendung. Um diese zu verbessern, ist eine Auseinandersetzung mit den Accessibility Features von SVG[170] und den WIA-ARIA-Standard[171] unumgänglich. Ideal wäre dann ein Test mit Probanden, welche im Umgang mit Screenreadern[172] vertraut sind. Insgesamt ist der meist stiefmütterlich behandelte Bereich der Barrierefreiheit ein Problem, welches mit der Alterung der Gesellschaft einen zunehmenden Stellenwert bekommen wird, nicht nur im Bereich der Webentwicklung.

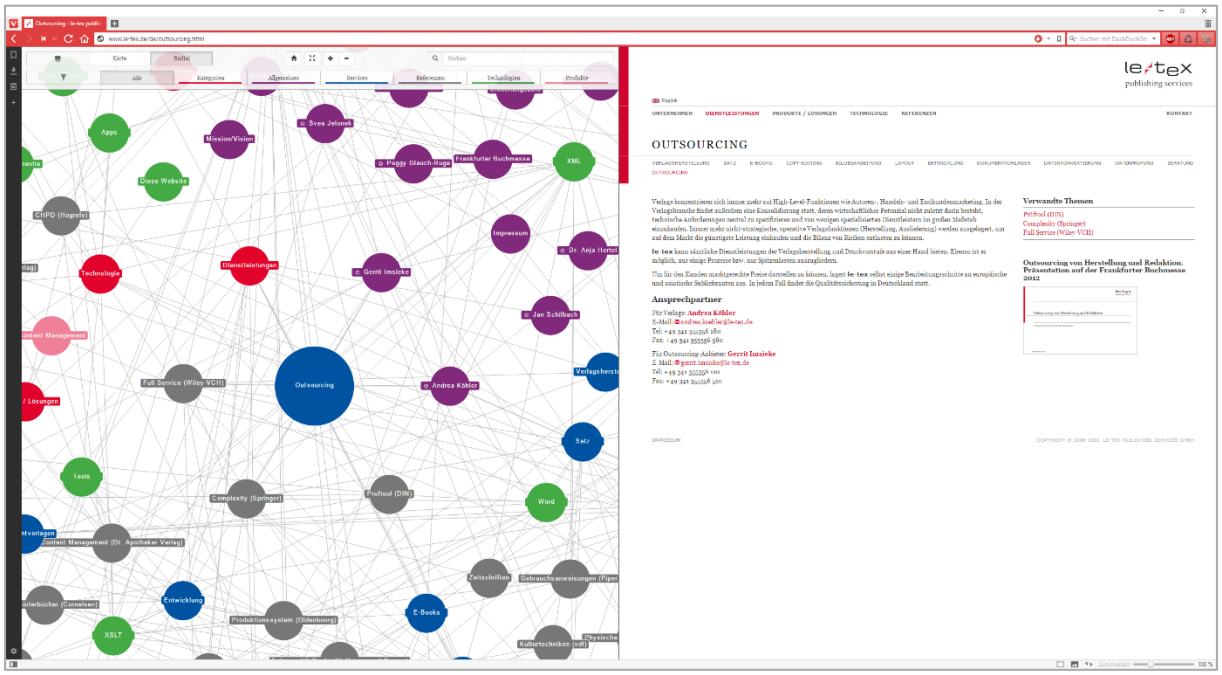


Abbildung 19: Screenshot der finalen Umsetzung

Was die Problematik der Lokalisierung anbelangt, ist diese bisher nur statisch umgesetzt. Zwar werden die SVG-Elemente und Daten für die Stichwortsuche bereits durch sprachspezifische JSON-Dateien vorgehalten, aber die Steuerungselemente sind bisher statisch in den sprachspezifischen Webseiten integriert. Hier wäre eine Lösung mit sprachspezifischen HTML-Dateien, welche die Elemente für die Steuerung der Navigator-Anwendung enthalten, sinnvoll. Diese ließe sich dann auch

[169] Ähnlich wie DocBook ein Dokumentenformat, welcher sogleich OASIS-Standard ist (OASIS, 2015).
[170] (W3C, 2000).
[171] (W3C, 2014).
[172] Zum Beispiel durch Vorlesen mittels Sprachsynthese oder durch Berühren eines Braille-Lesegerätes.

bei aktivierter Anwendung problemlos mit AJAX austauschen. Alternativ könnte man auch auf Templates von Web Components zurückgreifen.

Für mobile Endgeräte werden derzeit noch keine minimierten Versionen der Stylesheets und JavaScript-Dateien angeboten. Ein via Cygwin-Modul oder XProc-Step integrierter minimizer-Prozess, der die Dateien von unnötigem Whitespace befreit und Variablennamen verkürzt, könnte für mobile Geräte die Ladezeiten verkürzen. Insgesamt ist die Verwendung der interaktiven Navigation auf mobilen Geräten zwar möglich, aber aufgrund der Anzahl an Knoten ist eine praxistaugliche Benutzung unter Umständen nicht komfortabel genug. Eventuell ist eine rein textbasierende Lösung sinnvoller.

Kleinere Optimierungspotentiale stecken noch im Umschalten der Anwendung. Die bisher eher versteckte Möglichkeit könnte durch eine offensichtliche Schaltfläche ersetzt werden. Die Zoom-Funktion mittels Schaltflächen könnte auch um eine aktuelle Auswahl als Mittelpunkt skalieren, statt um den Mittelpunkt des gesamten SVG-Objektes.

Unabhängig von den Potentialen, die noch in der Umsetzung und Wiederverwertung der interaktiven Navigation stecken, wurden die gesetzten Ziele mit Hilfe dieser Arbeit erreicht. Die Entscheidung D3.js als Framework für die Umsetzung zu wählen, hat sich durch eine schnelle Umsetzbarkeit und hohe grafische Qualität bewährt. Jetzt muss sich die Anwendung nur noch als optionales Feature abseits der Frankfurter Buchmesse bewähren, was die Auswertung der Webserver-Statistiken zeigen werden.

Abkürzungsverzeichnis

AJAX ..... *Asynchronous JavaScript and XML*
API ..... *Application Programming Interface*
BSD ..... *Berkeley Software Distribution*
CDN ..... *Content Delivery Network*
CLI ..... *Command Line Interface*
CMS ..... *Content Management System*
CSS ..... *Cascading Style Sheets*
DITA ..... *Darwin Information Typing Architecture*
DOM ..... *Document Object Model*
DSSSL ..... *Document Style Semantics and Specification Language*
DTD ..... *Document Type Definition*
EOL ..... *End Of Life*
GUI ..... *Graphical User Interface*
HTML ..... *HyperText Markup Language, HyperText Markup Language*
HTTP ..... *Hypertext Transfer Protocol*
i18n ..... *Internationalization*
IE8 ..... *Internet Explorer Version 8*
JSON ..... *JavaScript Object Notation*
l10n ..... *Lokalization*
MVC ..... *Model View Controller*
PGML ..... *Precision Graphics Markup Language*
QName ..... *Qualified Name*
RDF ..... *Resource Description Framework*
Regex ..... *Regular Expression*
RegExp ..... *Regular Expression*
RELAX NG ..... *Regular Language Description for XML New Generation*
SEO ..... *Search Engine Optimization*
SGML ..... *Standard Generalized Markup Language*
SMIL ..... *Synchronized Multimedia Integration Language*
SVG ..... *Scalable Vector Graphics*
svn ..... *Subversion*
TBS ..... *Twitter Bootstrap*
TOC ..... *Table Of Contents*
UML ..... Unified Modeling Language
URI ..... *Uniform Resource Identifier*
UTF ..... *Unicode Transformation Format*
VML ..... *Vector Markup Language*
VSC ..... *Visual Studio Code*
W3C ..... *World Wide Web Consortium*
XHTML ..... *Extensible Hypertext Markup Language*
XML ..... *Extensible Markup Language*
XPath ..... *XML Path Language*
XSD ..... *XML Schema Definition*
XSL ..... *Extensible Stylesheet Language*
XSL-FO ..... *Extensible Stylesheet Language - Formatting Objects*
XSLT ..... *Extensible Stylesheet Language Transformation*

Literaturverzeichnis

Can I use, 2016. *Can I use... Support tables for HTML5, CSS3, etc.* [Online]
Available at: http://caniuse.com/
[Zugriff am 20 01 2016].

Cygwin, 2015. *Cygwin.* [Online]
Available at: http://www.cygwin.com
[Zugriff am 06 02 2016].

Data-Driven Documents, 2016. *D3.js - Data-Driven Documents.* [Online]
Available at: https://d3js.org/
[Zugriff am 15 02 2016].

Dublin Core Metadata Initiative, 2001. *DCMI Specifications.* [Online]
Available at: http://dublincore.org/specifications/
[Zugriff am 03 02 2016].

ECMA International, 2013. *ECMA-404.* [Online]
Available at: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf
[Zugriff am 29 01 2016].

Font Awesome, 2016. *Font Awesome, the iconic font and CSS toolkit.* [Online]
Available at: http://fontawesome.io/
[Zugriff am 22 01 2016].

GitHub - d3, 2015. *Zoom Behavior · mbostock/d3 Wiki · GitHub.* [Online]
Available at: https://github.com/mbostock/d3/wiki/Zoom-Behavior
[Zugriff am 23 02 2016].

GitHub - Modernizr, 2012. *Modernizr.touch detects touch events not touch devices · Issue #548 · Modernizr/Modernizr.* [Online]
Available at: https://github.com/Modernizr/Modernizr/issues/548
[Zugriff am 22 02 2016].

Google, 2016. *Google HTML/CSS Style Guide.* [Online]
Available at: https://google.github.io/styleguide/htmlcssguide.xml
[Zugriff am 02 01 2016].

Haverbeke, M., 2015. *ELOQUENT JAVASCRIPT. A Modern Introduction to Programming.* Second Edition Hrsg. San Francisco: no starch press.

HTTP Status Codes, 2016. *HTTP Status Codes — httpstatuses.com.* [Online]
Available at: https://httpstatuses.com/
[Zugriff am 15 02 2016].

ImageMagick, 2016. *ImageMagick: Convert, Edit, Or Compose Bitmap Images.* [Online]
Available at: http://www.imagemagick.org
[Zugriff am 06 02 2016].

iX, 2015. *Web-Browser: Microsoft integriert Googles VP9-Codec in Edge.* [Online]
Available at: http://www.heise.de/ix/meldung/Web-Browser-Microsoft-integriert-Googles-VP9-

Codec-in-Edge-2808386.html
[Zugriff am 28 01 2016].

jQuery, 2014. *jQuery UI Core | jQuery.* [Online]
Available at: http://plugins.jquery.com/ui.core/
[Zugriff am 29 01 2016].

JSON, 2016. *JSON.* [Online]
Available at: http://www.json.org/
[Zugriff am 29 01 2016].

Langer, M., 2014. *STIFTDESIGN/Leipzig | Grafikdesign, Illustration, Kommunikationdesign.* [Online]
Available at: http://www.stiftdesign.de
[Zugriff am 22 01 2016].

le-tex publishing services GmbH, 2015. *XML made in Germany - le-tex publishing services GmbH.* [Online]
Available at: http://www.le-tex.de/de/xmig.html
[Zugriff am 10 02 2016].

le-tex publishing services GmbH, 2016. *le-tex - le-tex publishing services GmbH.* [Online]
Available at: http://www.le-tex.de
[Zugriff am 28 01 2016].

Lohmann, S., Ziegler, J. & Tetzlaff, L., 2009. *Comparison of Tag Cloud Layouts: Task-Related Performance and Visual Exploration.* [Online]
Available at: https://www.uni-due.de/~s400268/Lohmann09-interact.pdf
[Zugriff am 04 02 2016].

MDN - Mozilla Developer Network, 2015. *Manipulieren des Browser-Verlaufes.* [Online]
Available at:
https://developer.mozilla.org/de/docs/Web/Guide/DOM/Manipulating_the_browser_history#Adding_and_modifying_history_entries
[Zugriff am 22 02 2016].

Microsoft, 2016. *Microsoft Support Lifecycle.* [Online]
Available at: https://support.microsoft.com/de-de/lifecycle/search?sort=PN&alpha=internet%20explorer
[Zugriff am 28 01 2016].

Modernizr, 2016. *Modernizr: the feature detection library for HTML5/CSS3.* [Online]
Available at: https://modernizr.com/
[Zugriff am 22 02 2016].

NETMARKETSHARE, 2016. *Browser market share.* [Online]
Available at: http://www.netmarketshare.com/browser-market-share.aspx?qprid=2&qpcustomd=0&qpsp=2016&qpnp=1&qptimeframe=Y
[Zugriff am 28 01 2016].

OASIS, 2015. *OASIS Darwin Information Typing Architecture (DITA) TC | OASIS.* [Online]
Available at: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dita
[Zugriff am 04 03 2016].

Oelke, D. & Gurevych, I., 2014. *A Study on Human-Generated Tag Structures to Inform Tag Cloud Layout.* [Online]
Available at: https://www.ukp.tu-darmstadt.de/fileadmin/user_upload/Group_UKP/publikationen/2014/StructuredTagClouds_CAMERA-READY.pdf
[Zugriff am 04 02 2016].

P.Hogan, B., 2011. *HTML5 & CSS3.* 1. Auflage Hrsg. Köln: O'Reilly.

Potschien, D., 2016. *SVG-Filter: So wendest du sie auf HTML-Elemente an.* [Online]
Available at: http://www.drweb.de/magazin/svg-filter-auf-html-elemente-anwenden-68424/
[Zugriff am 05 02 2016].

Response JS, 2016. *Response JS: mobile-first responsive design in HTML5..* [Online]
Available at: http://responsejs.com/
[Zugriff am 28 01 2016].

Schraitle, T., 2004. *DocBook-XML.* Kösel, Kempten: Suse Press.

sitemaps.org, 2008. *sitemaps.org - Home.* [Online]
Available at: http://www.sitemaps.org/
[Zugriff am 08 02 2016].

t3n, 2015. *Adaptive oder Responsive? Über die Vor- und Nachteile der beiden Design-Ansätze | t3n.* [Online]
Available at: http://t3n.de/news/adaptive-responsive-design-637267/
[Zugriff am 20 01 2016].

Twitter, 2015. *Bootstrap · The world's most popular mobile-first and responsive front-end framework..* [Online]
Available at: http://getbootstrap.com/
[Zugriff am 20 01 2016].

Twitter, 2016. *Bootstrap · The world's most popular mobile-first and responsive front-end framework..* [Online]
Available at: http://v4-alpha.getbootstrap.com/
[Zugriff am 29 01 2016].

video2brain, 2016. *Schemasprachen im Vergleich.* [Online]
Available at: https://www.video2brain.com/de/tutorial/schemasprachen-im-vergleich
[Zugriff am 24 02 2016].

Visscher, S., 2016. *GitHub - aFarkas/html5shiv.* [Online]
Available at: https://github.com/aFarkas/html5shiv
[Zugriff am 28 01 2016].

W3C, 1998. *Extensible Markup Language (XML) 1.0.* [Online]
Available at: https://www.w3.org/TR/1998/REC-xml-19980210
[Zugriff am 25 01 2016].

W3C, 2000. *Accessibility Features of SVG.* [Online]
Available at: www.w3.org/TR/SVG-access/
[Zugriff am 18 02 2016].

W3C, 2001. *Scalable Vector Graphics (SVG) 1.0 Specification.* [Online]
Available at: https://www.w3.org/TR/2001/REC-SVG-20010904/
[Zugriff am 03 02 2016].

W3C, 2005. *W3C Document Object Model.* [Online]
Available at: https://www.w3.org/DOM/
[Zugriff am 21 01 2016].

W3C, 2007. *XSL Transformations (XSLT) Version 2.0.* [Online]
Available at: https://www.w3.org/TR/xslt20/
[Zugriff am 26 02 2016].

W3C, 2008. *Extensible Markup Language (XML) 1.0 (Fifth Edition).* [Online]
Available at: http://www.w3.org/TR/2008/REC-xml-20081126/
[Zugriff am 25 01 2016].

W3C, 2008. *Scalable Vector Graphics (SVG) Tiny 1.2 Specification.* [Online]
Available at: https://www.w3.org/TR/SVGTiny12/
[Zugriff am 03 02 2016].

W3C, 2008. *W3C Synchronized Multimedia.* [Online]
Available at: http://www.w3.org/AudioVideo/
[Zugriff am 05 02 2016].

W3C, 2010. *XML Linking Language (XLink) Version 1.1.* [Online]
Available at: https://www.w3.org/TR/xlink11/
[Zugriff am 05 02 2016].

W3C, 2011. *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification.* [Online]
Available at: https://www.w3.org/TR/CSS2/
[Zugriff am 20 01 2016].

W3C, 2011. *Scalable Vector Graphics (SVG) 1.1 (Second Edition).* [Online]
Available at: https://www.w3.org/TR/SVG11/
[Zugriff am 01 02 2016].

W3C, 2011. *Selectors Level 3.* [Online]
Available at: https://www.w3.org/TR/selectors/
[Zugriff am 20 01 2016].

W3C, 2013. *CSS Transitions.* [Online]
Available at: http://www.w3.org/TR/css3-transitions/
[Zugriff am 05 02 2016].

W3C, 2014. *Accessible Rich Internet Applications (WAI-ARIA) 1.0.* [Online]
Available at: https://www.w3.org/TR/wai-aria/
[Zugriff am 18 02 2016].

W3C, 2014. *HTML5.* [Online]
Available at: http://www.w3.org/TR/html5/
[Zugriff am 28 01 2016].

W3C, 2014. *RDF - Semantic Web Standards.* [Online]
Available at: https://www.w3.org/RDF/
[Zugriff am 03 02 2016].

W3C, 2014. *XML Path Language (XPath) 3.0.* [Online]
Available at: https://www.w3.org/TR/xpath-30/
[Zugriff am 06 02 2016].

W3C, 2015. *W3C DOM4.* [Online]
Available at: https://www.w3.org/TR/domcore/
[Zugriff am 21 01 2016].

W3C, 2015. *XSL Transformations (XSLT) Version 3.0.* [Online]
Available at: https://www.w3.org/TR/xslt-30/
[Zugriff am 06 02 2016].

W3Schools, 2016. *SVG Filters.* [Online]
Available at: http://www.w3schools.com/svg/svg_filters_intro.asp
[Zugriff am 05 02 2016].

W3Techs, 2015. *Usage Statistics and Market Share of JavaScript Libraries for Websites, December 2015.* [Online]
Available at: http://w3techs.com/technologies/overview/javascript_library/all
[Zugriff am 20 12 2015].

Wenz, C., 2010. *JavaScript - Das umfassende Handbuch.* 10. Auflage Hrsg. Bonn: Galileo Press.

Wikipedia, 2015. *Document Object Model - Wikipedia.* [Online]
Available at: https://de.wikipedia.org/wiki/Document_Object_Model
[Zugriff am 21 01 2016].

XML made in Germany, 2015. *Fachvorträge rund um das crossmediale Publizieren.* [Online]
Available at: http://www.le-tex.de/img/fbm_flyer.pdf
[Zugriff am 10 02 2016].

Abbildungsverzeichnis

Tabellenverzeichnis

Markup- und Codeverzeichnis

Anhang

JavaScript-Funktionen

```
$.fn.startNavigator = function() {
  settings.active = true;
  settings.kioskMode = $("html").hasClass("kiosk-mode");

  if (settings.kioskMode === true) {
    settings.touchMode = true;
  }
  else {
    settings.touchMode = false;
  }
  myNetwork = Network();
  loadURL();
  d3.json("../lib/navigator/data/structure-" + settings.language + ".json",
function(json) {
    return myNetwork(settings, json);
  });

  if (settings.kioskMode === true) {
    autoDefault = setTimeout(function() {
        defaultSettings();
    }, settings.resetTimer);
  }

  window.onpopstate = function(event) {
    if (event.state.id) {
      return myNetwork.toggleActiveNode(event.state.id,
event.state.anchor);
    }
  };
  $(window).on("resize", function() {
    if (settings.active === true) {
      return myNetwork.toggleSize();
    }
  });
};
```

Listing 66: Funktion $.fn.startNavigator zum Starten des Navigators

```
network.toggleActiveNode = function(newActiveNode, anchor) {
  force.stop();
  setActiveNode(newActiveNode);
  d3.selectAll("circle").attr("r", nodeSize);
  network.setPageTitle(nodesMap.get(newActiveNode).name)
  network.loadPage((nodesMap.get(newActiveNode).url));
  if (anchor) {
    setTimeout(function() {
      network.gotoAnchor(anchor);
    }, 500);
  }
  return update();
};
```

Listing 67: Methode network.toggleActiveNode

```
network.setPageTitle = function(titleString) {
  var titleSelector;
  titleSelector = $('html head title');
  if (titleString) {
    titleSelector.text(titleString + " - le-tex publishing services");
  }
  else {
    titleSelector.text("le-tex publishing services");
  }
};
```

Listing 68: Methode network.setPageTitle

```
setLayout = function(newLayout) {
  layout = newLayout;
  switch (layout) {
    case "force":
      return force
        .on("tick", forceTick)
        .linkDistance(10)
        .linkStrength(.05)
        .gravity(.05)
        .charge(function(d) {
          return nodeCharge(d);
        });
      break;
    case "radial":
      return force
        .on("tick", radialTick)
        .charge(function(d) {
          return nodeCharge(d);
        });
      break;
    case "neighbours":
      return force
        .on("tick", neighbourTick)
        .charge(function(d) {
          return nodeCharge(d);
        });
      break;
  }
};
```

Listing 69: Funktion zum Umschalten der Layouts

```
filterNeighbours = function(nodeID, nodes, links) {
  var connectedLinks, neighbours;
  connectedLinks = filterConnectedLinks(nodeID, links);
  neighbours = filterConnectedNodes(nodeID, connectedLinks);
  return neighbours;
};
```

Listing 70: Funktion filterNeighbours

```
filterNotNeighbours = function(neighbours, nodes) {
  var notNeighbours = [];
  nodes.forEach(function(d) {
    if (neighbours.indexOf(d.id) === -1 ) {
      notNeighbours.push(d.id);
    }
  });
  return notNeighbours;
};
```

Listing 71: Funktion filterNotNeighbours

```
setInnerKeys = function(innerKeys) {
  increment = 360 / innerKeys.length;
  innerKeys.forEach(function(d) {
    return place(d);
  });
};
```

Listing 72: Funktion setInnerKeys

```
setOuterKeys = function(outerKeys) {
  radius = radius * 3;
  increment = 360 / outerKeys.length;
  outerKeys.forEach(function(d) {
    return place(d);
  });
};
```

Listing 73: Funktion setOuterKeys

```
network.resetZoom = function() {
  zoom.translate([0,0]).scale(1);
  d3.select("svg").select("g").transition()
    .attr("transform", "translate(0,0)scale(1)");
};
```

Listing 74: Methode network.resetZoom

```
network.zoomIn = function() {
  var currentScale, currentTranslate, newScale, newTranslate;
  currentScale = Math.round(zoom.scale()*10)/10;
  currentTranslate = zoom.translate();
  if (currentScale < 1) {
    newScale = currentScale + 0.1;
  }
  else if (currentScale >= 1 && currentScale < 5) {
    newScale = currentScale + 1;
  }
  else {
    newScale = 5;
  }
  newTranslate = network.calculateTranslate(currentScale, currentTranslate,
newScale);
  network.scaleTranslate(newScale, newTranslate);
};
```

Listing 75: Methode network.zoomIn

```
network.zoomOut = function() {
  var currentScale, currentTranslate, newScale, newTranslate;
  currentScale = Math.round(zoom.scale()*10)/10;
  currentTranslate = zoom.translate();
  if (currentScale <= 1 && currentScale > 0.2) {
    newScale = currentScale - 0.2;
  }
  else if (currentScale >= 2) {
    newScale = currentScale - 1;
  }
  else {
    newScale = 0.2;
  }
  newTranslate = network.calculateTranslate(currentScale, currentTranslate,
newScale);
  network.scaleTranslate(newScale, newTranslate);
};
```

Listing 76: Methode network.zoomOut

```
network.scaleTranslate = function(newScale, newTranslate) {
  zoom.translate(newTranslate).scale(newScale);
  d3.select("svg").select("g").transition()
    .attr("transform", "translate(" + newTranslate + ")scale(" + newScale +
")");
};
```

Listing 77: Methode network.scaleTranslate

```
network.calculateTranslate = function(currentScale, currentTranslate,
newScale) {
  if (currentTranslate[0] === 0 && currentTranslate[1] === 0) {
    return [
      (-width/2)*(newScale-1),
      (-height/2)*(newScale-1)
    ];
  }
  else if (currentScale === 1) {
    return [
      (currentTranslate[0]*(newScale+1)),
      (currentTranslate[1]*(newScale+1))
    ];
  }
  else {
    var baseTranslate = [
      (-currentTranslate[0]/(currentScale-1)),
      (-currentTranslate[1]/(currentScale-1))
    ];
    return [
      (-baseTranslate[0]*(newScale-1)),
      (-baseTranslate[1]*(newScale-1))
    ];
  }
};
```

Listing 78: Methode network.calculateTranslate

```
updateNodes = function() {
  node = nodesG.selectAll("g.node")
    .data(curNodesData, function(d) {
      return d.id;
    });
  node.enter()
    .append("g")
      .attr("id", function(d) {
        return d.id;
      })
      .attr("class", function(d) {
        return "node " + d.type;
      })
    .append("circle")
      .attr("xlink:href", function(d) {
        return d.url;
      })
      .attr("r", function(d) {
        return nodeSize(d);
      });
  node.select("text").remove();
  node
    .append("text")
      .attr("text-anchor", "middle")
      .attr("dy", 3)
      .text(function(d) {
        return d.name;
      })
      .each(function(d) {
        var width = this.getBBox().width;
        if (width <= 70) {
          d.width = 90;
        }
        else {
          d.width = width + 10;
        }
      });
  node.select(".front-rect").remove();
  node
    .insert("rect", ["text"])
      .attr("class", "front-rect")
      .attr("width", function(d) {
        return d.width;
      })
      .attr("height", 18)
      .attr("x", function(d) {
        return -d.width/2;
      })
      .attr("y", -10)
      .attr("rx", 3);
  node.select(".back-rect").remove();
  node
    .insert("rect", ["circle"])
      .attr("width", function(d) {
        return d.width + strokeWidth*2;
      })
      .attr("class", "back-rect")
      .attr("height", 20)
```

```
      .attr("x", function(d) {
        return -(d.width/2 + strokeWidth);
      })
      .attr("y", -11)
      .attr("rx", 3);
  node.on("mouseover", showNeighbourRelation).on("mouseout",
hideNeighbourRelation);
  node.on("click", function(d) {
      network.gotoPage(d.id);
      if (touchMode === true){
        $.fn.restartDefaultTimer();
      }
  });
  return node.exit().remove();
};
```

Listing 79: Funktion updateNodes

```
<xsl:function name="letex:label" as="xs:string">
  <xsl:param name="content-item" as="element(*)" />
  <xsl:choose>
    <xsl:when test="matches($content-item/titleabbrev, '\S')">
      <xsl:value-of select="$content-item/titleabbrev" />
    </xsl:when>
    <xsl:when test="$content-item/title">
      <xsl:value-of select="$content-item/title" />
    </xsl:when>
    <xsl:when test="$content-item/personname">
      <xsl:value-of select="string-join(('&#x263a;', $content-
item/personname/node()[matches(.,'\S')]), ' ')" />
    </xsl:when>
    <xsl:otherwise>
      <xsl:sequence select="' '"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:function>
```

Listing 80: Funktion letex:label

```
<xsl:function name="letex:vertex-id" as="xs:string">
  <xsl:param name="id" as="xs:string" />
  <xsl:sequence select="replace($id, '^(Author|Othercredit)-', '')" />
</xsl:function>
```

Listing 81: Funktion letex:vertex-id

```
<xsl:template match="related" mode="related-for-sitemap">
  <xsl:for-each select="item">
    <xsl:variable name="content-item" select="key('by-id', @xml:id, $root)"
as="element(*)" />
    <node origin="note_4" xml:id="{@xml:id}" label="{letex:label($content-
item)}" relation="cs" type="{letex:node-type($content-item)}"
xmlns="http://docbook.org/ns/docbook"/>
  </xsl:for-each>
</xsl:template>
```

Listing 82: Template related

```
<xsl:function name="letex:related" as="element(related)?">
  <xsl:param name="to" as="element(*)" />
  <xsl:variable name="own-bibliorelations" select="key(
                                                    'by-id',
                                                    key('by-id',
$to/@xml:id, root($to))/info/bibliorelation/link/@linkend
                                                      union
                                                      key('by-id',
$to/@xml:id, root($to))/*[not(self::chapter)]//*[self::xref or
self::link[ancestor::*/@xml:id = 'phonebook']]/@linkend[matches(.,
'^(Author|Othercredit)-')],
                                                    root($to)
                                                  )" as="element(*)*" />
  <xsl:variable name="other-bibliorelations" select="key('biblio-refers-
to', $to/@xml:id, root($to))" />
  <xsl:variable name="all-bibliorelations" select="$own-bibliorelations
union $other-bibliorelations" as="element(*)*" />
  <xsl:if test="count($all-bibliorelations) gt 0">
    <related to="{$to/@xml:id}" xmlns="http://docbook.org/ns/docbook">
      <xsl:for-each select="$all-bibliorelations">
        <item xml:id="{@xml:id}" title="{letex:label(.)}" type="{local-
name()}"/>
      </xsl:for-each>
    </related>
  </xsl:if>
</xsl:function>
```

Listing 83: Funktion letex:related

```
<xsl:function name="letex:node-type" as="xs:string">
  <xsl:param name="content-item" as="element(*)" />
  <xsl:choose>
    <xsl:when test="$content-item/self::part">
      <xsl:value-of select="'category'"/>
    </xsl:when>
    <xsl:when test="$content-item/parent::part/@xml:id eq 'technology'">
      <xsl:value-of select="'technology'"/>
    </xsl:when>
    <xsl:when test="$content-item/parent::part/@xml:id eq 'products'">
      <xsl:value-of select="'products'"/>
    </xsl:when>
    <xsl:when test="$content-item/parent::part/@xml:id eq 'services'">
      <xsl:value-of select="'services'"/>
    </xsl:when>
    <xsl:when test="$content-item/parent::part/@xml:id eq 'case-studies'">
      <xsl:value-of select="'story'"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="'generic'"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:function>
```

Listing 84: Funktion letex:node-type

```
<xsl:function name="letex:jsonify" as="xs:string">
  <xsl:param name="text-to-jsonify"/>
  <xsl:value-of select="letex:replaces( $text-to-jsonify, (
    '\\','\\\\',
    '/', '\\/',
    '&quot;|&#8222;|&#8220;', '\\&quot;',
    '&#xA;','\\n',
    '&#xD;','\\r',
    '&#x9;','\\t',
    '\n','\\n',
    '\r','\\r',
    '\t','\\t',
    ',\s*$','',
    '\s+$','',
    '^\s+','',
    ':$',''
    ) )"/>
</xsl:function>
```

Listing 85: Funktion letex:jsonify

```
<xsl:function name="letex:replaces">
  <xsl:param name="replace-text" as="xs:string+"/>
  <xsl:param name="search-and-replacement" as="item()+"/>
  <xsl:variable name="replaced" select="replace( $replace-text,
    $search-and-replacement[1],
    $search-and-replacement[2] )"/>
  <xsl:choose>
    <xsl:when test="count( $search-and-replacement ) lt 3">
      <xsl:sequence select="$replaced"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:sequence select="letex:replaces( $replaced, $search-and-
replacement[ position() gt 2 ])"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:function>
```

Listing 86: Funktion letex:replaces

```
<xsl:function name="letex:remove-fillwords" as="xs:string*">
  <xsl:param name="with-fillwords" as="xs:string*"/>
  <xsl:sequence select="for $i in $with-fillwords return lower-case(
   replace(
       $i, '[.,;:!?\(\)]', ''
     )
   )
   [not(
     matches(
       ., '^(\d+|[ivx]+|
         und|oder|auch|aber|and|or|to|too|ist|sind|seid|is|are|but|also|
         wer|wie|was|who|what|how|
         der|das|die|des|this|of|the|
         mit|zum|zu|zur|auf|in|im|den|denen|deren|von|with|on|of|
         wir|ihr|sie|ich|du|er|sie|es|ihren|we|you|he|she|
         als|wie|wenn|like|if|else|bei|bis|aus|near|next|until|from|as|at|
         nur|nicht|nichts|not|only|ein|eine|einen|einem|one|a|an|
         für|von|vor|indem|dass|daß|durch|damit|for|that|this|them)$', 'x')
     )
    ]"/>
</xsl:function>
```

Listing 87: Funktion letex:remove-fillwords

```
<xsl:template name="kiosk-controls">
  <xsl:choose>
    <xsl:when test="$lang = 'de'">
      <div class="row">
        <div class="col-sm-4 hidden-xs">
          <div id="layouts" class="btn-group btn-group-justified" data-
toggle="buttons">
            <span class="input-group-addon">
              <span class="glyphicon glyphicon-blackboard" aria-
hidden="true"></span>
            </span>
            <label class="btn btn-sm btn-default btn-view-force" data-
view="force">
              <input type="radio" name="options" autocomplete="off"
checked="" /> Karte
            </label>
            <label class="btn btn-sm btn-default btn-view-neighbours
active" data-view="neighbours">
              <input type="radio" name="options" autocomplete="off" />
Radial
            </label>
          </div>
        </div>
        <div id="base-controls" class="col-xs-6 col-sm-4 btn-space-
between">
          <button id="defaults" class="btn btn-sm btn-default">
            <span class="glyphicon glyphicon-home" aria-
hidden="true"></span>
          </button>
          <button id="relocate" class="btn btn-sm btn-default">
            <span class="glyphicon glyphicon-fullscreen" aria-
hidden="true"></span>
          </button>
          <div class="btn-group" data-toggle="buttons">
            <button id="zoom-in" class="btn btn-sm btn-default">
              <span class="glyphicon glyphicon-plus" aria-
hidden="true"></span>
            </button>
            <button id="zoom-out" class="btn btn-sm btn-default">
              <span class="glyphicon glyphicon-minus" aria-
hidden="true"></span>
            </button>
          </div>
        </div>
        <div class="col-xs-6 col-sm-4">
          <div class="input-group input-group-sm input-group-search">
            <span class="input-group-addon">
              <span class="glyphicon glyphicon-search" aria-
hidden="true"></span>
            </span>
            <input type="search" class="form-control" id="search"
placeholder="Suchen" />
            <span class="input-group-btn">
              <button id="clear-search" class="btn btn-sm btn-default">
                <span class="glyphicon glyphicon-remove" aria-
hidden="true"></span>
```

```
            </button>
          </span>
        </div>
      </div>
    </div>
    <div id="filters" class="btn-group btn-group-justified" data-
toggle="buttons">
      <span class="input-group-addon hidden-xs">
        <span class="glyphicon glyphicon-filter" aria-hidden="true" />
      </span>
      <label class="btn btn-sm btn-default btn-filter-all active" data-
filter="all">
        <input type="radio" name="options" autocomplete="off" checked=""
/> Alle
      </label>
      <label class="btn btn-sm btn-default btn-filter-category" data-
filter="category">
        <input type="radio" name="options" autocomplete="off"
/>Kategorien
        <div class="filter-color"></div>
      </label>
      <label class="btn btn-sm btn-default btn-filter-generic" data-
filter="generic">
        <input type="radio" name="options" autocomplete="off"
/>Allgemeines
        <div class="filter-color"></div>
      </label>
      <label class="btn btn-sm btn-default btn-filter-services" data-
filter="services">
        <input type="radio" name="options" autocomplete="off" />Services
        <div class="filter-color"></div>
      </label>
      <label class="btn btn-sm btn-default btn-filter-story" data-
filter="story">
        <input type="radio" name="options" autocomplete="off"
/>Referenzen
        <div class="filter-color"></div>
      </label>
      <label class="btn btn-sm btn-default btn-filter-technology" data-
filter="technology">
        <input type="radio" name="options" autocomplete="off"
/>Technologien
        <div class="filter-color"></div>
      </label>
      <label class="btn btn-sm btn-default btn-filter-products" data-
filter="products">
        <input type="radio" name="options" autocomplete="off" />Produkte
        <div class="filter-color"></div>
      </label>
    </div>
  </xsl:when>
  <xsl:when test="$lang = 'en'">
    <div class="row">
      <div class="col-sm-4 hidden-xs">
        <div id="layouts" class="btn-group btn-group-justified" data-
toggle="buttons">
          <span class="input-group-addon">
            <span class="glyphicon glyphicon-blackboard" aria-
```

```
hidden="true"></span>
            </span>
            <label class="btn btn-sm btn-default btn-view-force" data-
view="force">
              <input type="radio" name="options" autocomplete="off"
checked="" /> Map
            </label>
            <label class="btn btn-sm btn-default btn-view-neighbours
active" data-view="neighbours">
              <input type="radio" name="options" autocomplete="off" />
Radial
            </label>
          </div>
        </div>
        <div id="base-controls" class="col-xs-6 col-sm-4 btn-space-
between">
          <button id="defaults" class="btn btn-sm btn-default">
            <span class="glyphicon glyphicon-home" aria-
hidden="true"></span>
          </button>
          <button id="relocate" class="btn btn-sm btn-default">
            <span class="glyphicon glyphicon-fullscreen" aria-
hidden="true"></span>
          </button>
          <div class="btn-group" data-toggle="buttons">
            <button id="zoom-in" class="btn btn-sm btn-default">
              <span class="glyphicon glyphicon-plus" aria-
hidden="true"></span>
            </button>
            <button id="zoom-out" class="btn btn-sm btn-default">
              <span class="glyphicon glyphicon-minus" aria-
hidden="true"></span>
            </button>
          </div>
        </div>
        <div class="col-xs-6 col-sm-4">
          <div class="input-group input-group-sm input-group-search">
            <span class="input-group-addon">
              <span class="glyphicon glyphicon-search" aria-
hidden="true"></span>
            </span>
            <input type="text" class="form-control" id="search"
placeholder="Search" />
            <span class="input-group-btn">
              <button id="clear-search" class="btn btn-sm btn-default">
                <span class="glyphicon glyphicon-remove" aria-
hidden="true"></span>
              </button>
            </span>
          </div>
        </div>
      </div>
      <div id="filters" class="btn-group btn-group-justified" data-
toggle="buttons">
        <span class="input-group-addon hidden-xs">
          <span class="glyphicon glyphicon-filter" aria-hidden="true" />
        </span>
        <label class="btn btn-sm btn-default btn-filter-all active" data-
```

```
filter="all">
          <input type="radio" name="options" autocomplete="off" checked=""
/> All
        </label>
        <label class="btn btn-sm btn-default btn-filter-category" data-
filter="category">
          <input type="radio" name="options" autocomplete="off"
/>Categories
          <div class="filter-color"></div>
        </label>
        <label class="btn btn-sm btn-default btn-filter-generic" data-
filter="generic">
          <input type="radio" name="options" autocomplete="off" />Generic
          <div class="filter-color"></div>
        </label>
        <label class="btn btn-sm btn-default btn-filter-services" data-
filter="services">
          <input type="radio" name="options" autocomplete="off" />Services
          <div class="filter-color"></div>
        </label>
        <label class="btn btn-sm btn-default btn-filter-story" data-
filter="story">
          <input type="radio" name="options" autocomplete="off" />Stories
          <div class="filter-color"></div>
        </label>
        <label class="btn btn-sm btn-default btn-filter-technology" data-
filter="technology">
          <input type="radio" name="options" autocomplete="off"
/>Technologies
          <div class="filter-color"></div>
        </label>
        <label class="btn btn-sm btn-default btn-filter-products" data-
filter="products">
          <input type="radio" name="options" autocomplete="off" />Products
          <div class="filter-color"></div>
        </label>
      </div>
    </xsl:when>
  </xsl:choose>
</xsl:template>
```

Listing 88: Template für die Navigator-Steuerung

CSS-Definitionen

```
.btn-filter-category .filter-color {
  background-color: #e2032c;
}
.btn-filter-generic .filter-color {
  background-color: #81297B;
}
.btn-filter-services .filter-color {
  background-color: #0053a1;
}
.btn-filter-technology .filter-color {
  background-color: #44aa44;
}
.btn-filter-story .filter-color {
  background-color: #777;
}
.btn-filter-products .filter-color {
  background-color: #f07f98;
}
```

Listing 89: Farbreferenz für die Filter-Buttons

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt und durch meine Unterschrift, dass die vorliegende Bachelorarbeit von mir selbstständig, ohne fremde Hilfe angefertigt worden ist.

Inhalte und Passagen, die aus fremden Quellen stammen und direkt oder indirekt übernommen worden sind, wurden als solche kenntlich gemacht. Ferner versichere ich, dass ich keine andere, außer der im Literaturverzeichnis angegebenen Literatur verwendet habe. Diese Versicherung bezieht sich sowohl auf Textinhalte sowie alle enthaltenden Abbildungen, Skizzen und Tabellen.

Die Arbeit wurde bisher keiner Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Leipzig, 04.03.2016, Alexander Jache