



Jade Hochschule
Management, Information & Technologie
Wirtschaftsinformatik

Bachelorarbeit

über das Thema

**Prototypische Implementierung einer SAP UI5 Applikation
im SAP Umfeld und Analyse eines effizienten Einsatz von
UI-Objekten**

eingereicht von: Nils Lutz

bei: Prof. Dr. Hergen Pargmann
Prof. Dr. Harald Schallner

I Kurzfassung

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Abstract

Das ganze auf Englisch.

II Inhaltsverzeichnis

I	Kurzfassung	I
II	Inhaltsverzeichnis	II
III	Abbildungsverzeichnis	IV
IV	Tabellenverzeichnis	V
V	Listing-Verzeichnis	V
VI	Abkürzungsverzeichnis	VI
1	Einleitung	1
2	Technologien	2
2.1	HTML5	2
2.2	CSS3	9
2.3	JavaScript	12
2.4	ABAP	14
2.5	SAP UI5 Framework	15
2.5.1	Definition	15
2.5.2	Architektur	15
2.5.3	OData Protokoll	15
3	Software Ergonomie	16
3.1	Definition	16
3.2	DIN EN ISO 9241	16
3.3	Analyse Methoden	17
3.4	SAP Technologien in Bezug auf Software Ergonomie	17
3.4.1	Business Server Pages	17
3.4.2	Web Dynpro for ABAP	17
3.4.3	SAP Fiori / SAP UI5 / SAP Screen Personas	17
4	Fallbeispiel SAP UI5	18
4.1	Beschreibung	18
4.2	Hilfsmittel	18
4.2.1	Entwicklungsumgebung	18
4.2.2	UI Design und Prototyping	18
4.3	Implementierung	18
4.3.1	View	18
4.3.2	Model und Controller	20
4.3.3	Backend	21
5	Analyse	22
5.1	Heatmap	22
5.2	UI-Objekte	22

5.3 PLATZHALTER	22
6 Schluss	23
7 Quellenverzeichnis	24
Anhang	I
A GUI	I

III Abbildungsverzeichnis

Abb. 1	HTML5 Spezifikationen Übersicht	3
Abb. 2	CSS-Boxmodell	10
Abb. 3	DOM Beispielbaum	13
Abb. 4	Model-View-Controller-Architekturmuster	15

IV Tabellenverzeichnis

Tab. 1	HTML5 Browserkompatibilität	4
--------	---------------------------------------	---

V Listing-Verzeichnis

Lst. 1	(X)HTML4.01 <i>doctype</i> -Element	5
Lst. 2	HTML5 <i>doctype</i> -Element	5
Lst. 3	HTML5 Basis Dokument	5
Lst. 4	HTML5 <i>meta</i> -Element	6
Lst. 5	HTML5 <i>header</i> - und <i>footer</i> -Element	6
Lst. 6	HTML5 Struktur Elemente	7
Lst. 7	HTML5 Internet Explorer Fallback	8
Lst. 8	HTML5 <i>input</i> -Element	9
Lst. 9	CSS3 Syntax Beispiel	10
Lst. 10	CSS3 medienspezifisches Stylesheet	10
Lst. 11	CSS3 eigenschaftsspezifisches Stylesheet	11
Lst. 12	Stylesheet Einbindung über <i>link</i> -Element	11
Lst. 13	Stylesheet Einbindung über <i>style</i> -Element	11
Lst. 14	Stylesheet Einbindung in <i>html</i> -Element	12
Lst. 15	JavaScript Einbindung als separate Datei im <i>head</i> -Element	12
Lst. 16	JavaScript Einbindung in Skript Element im <i>head</i> - und <i>body</i> -Element	12
Lst. 17	DOM5 Beispiel Definition	12
Lst. 18	Root View der Applikation	18
Lst. 19	Component.js - Datenmodell an die Root View binden	20

VI Abkürzungsverzeichnis

JSP	Java Server Pages
BSP	Business Server Pages
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
WWW	World Wide Web
W3C	World Wide Web Consortium
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language
SGML	Standard Generalized Markup Language
DOM	Dokument-Objekt-Modell
WHATWG	Web Hypertext Application Technology Working Group
API	Application-Programming-Interface

1 Einleitung

Motivation // wieso weshalb warum wo

// Beschreibung abatAG

// Entstehung des Projekts

Problemstellung // aktuelle situationsbeschreibung

// was soll besser laufen

Zielsetzung // Das Produkt - Template Programmierung für SAP Frontends mit SAP UI5

Struktur // der weg über die software ergonomie und ihre wichtigkeit, gezeigt über die Marktanalyse, hin zur praktischen Umsetzung durch Grundlagen und Beschreibung des Lösungsweges

2 Technologien

Zum besseren Verständnis der gesamten Thematik werden in den folgenden Kapiteln Technologien erläutert. Die Grundlagen und besonderen Merkmale der einzelnen Technologien helfen dabei die spätere Analyse nachvollziehen zu können. Zu den Kernsprachen, mit denen im Browser visuelle Informationen angezeigt und verändert werden können, zählen unter anderem die Auszeichnungssprache Hypertext Markup Language (HTML), die Gestaltungssprache Cascading Style Sheets (CSS) und die Skriptsprache JavaScript (JS). Aufbauend auf den drei genannten Sprachen setzen sich in der Regel Frameworks. Frameworks sind in sich konsistente Bibliotheken die gewisse Sprachkonstrukte, welche häufig benötigt werden in der Entwicklung, zur Verfügung stellen. Mit dem Einsatz eines Frameworks verfolgt man das Ziel oft geschriebenen Programmcode in eine Art „Bausatz-Konstruktion-Set“ auszulagern. So lässt sich ein einmal durchgeführter Entwicklungsprozess beliebig oft und mit weit weniger Aufwand bewerkstelligen, als wenn man jedes Mal den Programmcode von neuem entwickeln müsste.

2.1 HTML5

Historie HTML5 ist die aktuell empfohlene Spezifikation des World Wide Web Consortium (W3C) und sie stellt eine der Kernsprachen des World Wide Web dar. Angefangen hat es am 13. März 1989, als Tim Berners-Lee am CERN in Genf das World Wide Web (WWW) ins Leben gerufen und damit zusammen HTML festgelegt hat. So entstand ab 1990 eine Spezifikation seitens des W3C zur Festlegung und Vereinheitlichung der Kommunikation über das Internet. Im November 1995 erklärte das W3C HTML 2.0 zum offiziellen Sprachstandard. Grundlegende Unterschiede zwischen Version 1.0 und 2.0 existieren nicht. Version 3.0 der HTML Spezifikation ist gänzlich am Browser Markt vorbei definiert worden. Aus diesem Grund wurde HTML 3.2 ab Januar 1997 zum Nachfolger von Version 2.0 gemacht. Die folgende Entwicklung der Spezifikation brachte 1999 die überarbeitete Version 4.01 hervor. Im selben Zug wurde CSS, als Gestaltungssprache für HTML, immer mehr fokussiert. So begann die Fragmentierung der HTML Spezifikation und es existierten drei Versionen zur selben Zeit. Nämlich HTML 4.01 *strict*, die dem eigentlich definierten HTML am nächsten kam. HTML 4.01 *transitional*, nach welcher auch einige übliche physische Textauszeichnungen vorgesehen waren. „Physische Textauszeichnungen haben Bedeutungen wie „fett“ oder „kursiv“, stellen also direkte Angaben zur gewünschten Schriftformatierung dar. Bei physischen Elementen sollte der Web-Browser eine Möglichkeit finden, den so ausgezeichneten Text entsprechend darzustellen.“[Sela]. Sie wurde als Übergangslösung entwickelt. Die dritte Variante ist HTML 4.01 *frameset*. Der einzige Unterschied zur

transitional Variante ist, dass sich im Rumpf eines HTML Dokuments ein Element verändert. Neben HTML wurde ab Januar 2000 auch eine Extensible Hypertext Markup Language (XHTML) genannte Spezifikation entwickelt, die HTML mit dem Extensible Markup Language (XML) Standard vereinen sollte. XHTML ist allerdings nicht als eigenständige Sprache zu verstehen, sondern als eine Serialisierungsform für HTML unter Verwendung von XML. Mit HTML5 wurde die Spezifikation nicht mehr durch die Standard Generalized Markup Language (SGML) - eine Metasprache zur Definition von Auszeichnungssprachen - sondern durch ein Dokument-Objekt-Modell (DOM) beschrieben. Die in dieser Version neu eingeführten Elemente sollten es erlauben HTML Dokumente semantisch vernünftiger zu strukturieren.(vgl. [CG12] S.20ff) Im Oktober 2014 wurde HTML5 dann vom W3C zum De-facto Standard des WWW erklärt. Heute existiert neben der Spezifikation des W3C auch noch ein sogenannter „lebender Standard“ der Web Hypertext Application Technology Working Group (WHATWG). Die WHATWG ist ein Zusammenschluss von Unternehmen wie zum Beispiel Mozilla Foundation, Opera Software und Apple. Der allgemeine Sprachgebrauch von HTML ist dadurch nicht an die W3C Spezifikation gebunden. Er erstreckt sich über den „lebenden Standard“ derWHATWG hinaus und beinhaltet zahlreiche Schnittstellen zu anderen Technologien. Abbildung 1 verdeutlicht diese Situation.

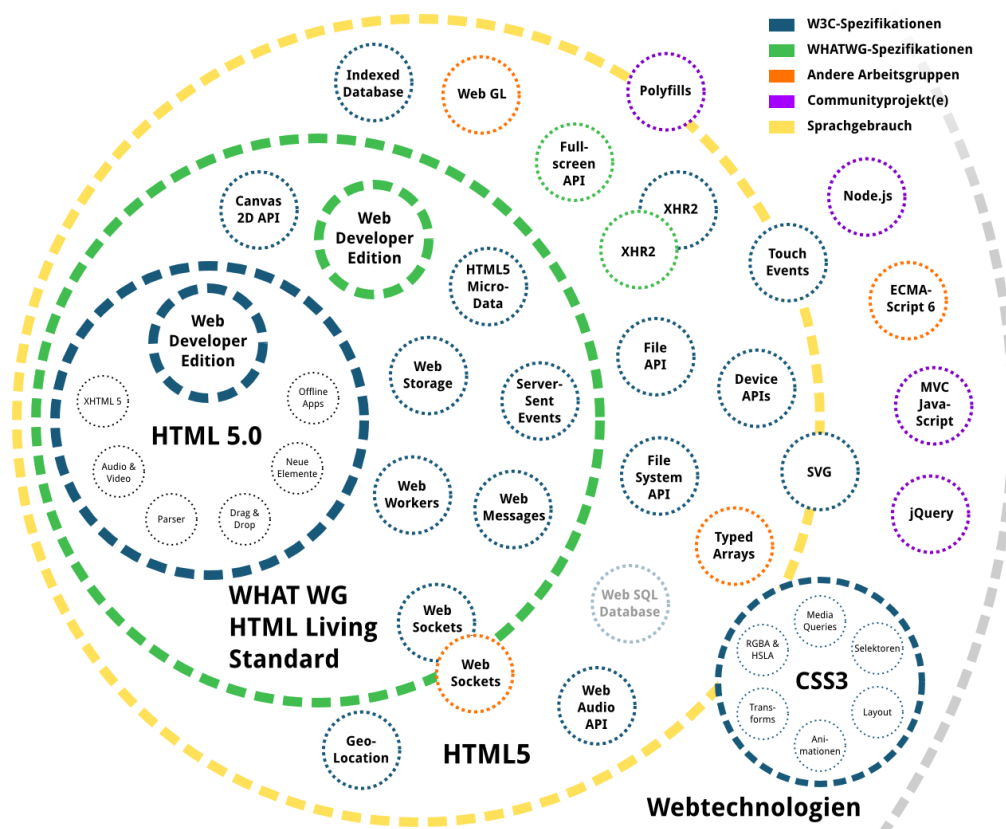


Abbildung 1: HTML5 Spezifikationen Übersicht

Ziele HTML5 wurde mit besonderem Augenmerk auf die Kompatibilität entwickelt. Vorhandene Spezifikationen wie HTML4.01, XHTML1.0 und DOM2 sollten unter einem Dach gebündelt werden. Hierdurch wird der vorangegangenen Fragmentierung entgegen gewirkt. Schon vorhandene Inhalte müssen weitestgehend unterstützt werden auch wenn sie nicht zur HTML5 Spezifikation gehören. Beispielsweise werden fehlerhaft verschachtelte Elemente trotzdem akzeptiert. *Graceful degradation* ist als ein weiteres Ziel für HTML5 definiert worden und bedeutet soviel wie „Schrittweise Abstufung“. Es stellt sicher, dass ein HTML Dokument auch dann verarbeitet wird sollte der verwendete Browser ein bestimmtes benutztes Element nicht unterstützen. Weiter galt für die Spezifikation, dass schon vorhandene Techniken, die weitläufig verbreitet sind, nicht neu entwickelt werden sollten. Stattdessen sollten sie übernommen werden. Dies beruht auf dem Umstand, dass die Browser Hersteller jeweils ihre eigenen Techniken bevorzugen und weiter entwickeln und dadurch auch für ihre Verbreitung sorgen. Evolution statt Revolution stand über den Zielen von HTML5. (X)HTML wurde weiter entwickelt und nicht von Grund auf neu definiert. So ist in Tabelle 1 die zum aktuellen Zeitpunkt verfügbare Unterstützung von HTML5 in den gängigsten Browsern abzulesen.

Hersteller	Desktop/Mobile	Version
Mozilla	Firefox	4.0
	Firefox Mobile	16
Google	Chrome	10
	Chrome Mobile	25
	Android	4.0
Apple	Safari	5.1
	Safari iOS	5.1
Microsoft	Internet Explorer	10
	Windows Phone	8
Opera Software	Opera	11.64
Blackberry	Browser	10

Tabelle 1: HTML5 Browserkompatibilität

Aufbau Ein jedes HTML Dokument beginnt mit dem sogenannten *Doctype*. Dieser legt fest mit welcher Syntax das Dokument aufgebaut ist und wie das Dokument vom Browser verarbeitet werden soll. Verschiedene Varianten wie *strict*, *transitional* und *frameset* sind in HTML5 nicht vorgesehen. In den Vorgängerversionen musste die Variante jedoch mit angegeben werden um eine eindeutige Interpretation des Dokuments zu gewährleisten. Listing 1 zeigt die beiden *Doctype* von HTML 4.01 und XHTML 1.0. Durch die Abwärtskompatibilität von HTML5 sind auch diese *Doctype* heute noch

gültig und das Dokument wird korrekt vom Parser interpretiert werden.

```

1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
2      "http://www.w3.org/TR/html4/strict.dtd">
3  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
4      "http://www.w3.org/TR/html4/loose.dtd">
5  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
6      "http://www.w3.org/TR/html4/frameset.dtd">
7  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
8      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

```

Listing 1: (X)HTML4.01 *doctype*-Element

Listing 2 hingegen zeigt das *Doctype* von HTML5. Es wurde enorm gekürzt im Vergleich zu dem *Doctype* von HTML 4.01 und XHTML 1.0. Groß- und Kleinschreibung ist nicht von Bedeutung innerhalb des *Doctype*.

```

1  <!DOCTYPE html>

```

Listing 2: HTML5 *doctype*-Element

Nach dem *Doctype* folgt der restliche Dokument Aufbau in HTML Syntax. Diese teilt sich auf in Elemente und Attribute, die diesen Elementen zugeordnet und mit Werten versehen werden können. Es existieren für die meisten Elemente Start- und Endmarkierungen. Für einige Elemente sind die Start- bzw. Endmarkierungen optional und für einige wiederum verpflichtend in der Dokumentstruktur zu setzen. In Listing 3 sieht man ein valides HTML5 Dokument mit den Mindestanforderungen. (vgl. [Krö11] S.58)

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4  <title>Beispiel Seite</title>
5  </head>
6  <body>
7  <h1>Beispiel Seite</h1>
8  <p>Dies ist ein <a href="demo.html">einfaches</a> Beispiel.</p>
9  <!-- Dies ist ein Kommentar -->
10 </body>
11 </html>

```

Listing 3: HTML5 Basis Dokument

Wichtige neue Sprachelemente In HTML5 wurden die Mikrodaten mit aufgenommen. Mit Mikrodaten bietet sich eine weitere Möglichkeit ein HTML Dokument semantisch zu spezifizieren. Metadaten wie z.B. der verwendete Zeichensatz lassen sich so festlegen. Browser und Webseiten können über die Mikrodaten-Application-Programming-Interface (API) die gesetzten Werte auslesen und weiter verarbeiten. Auch Suchmaschinen können auf die Metadaten zugreifen, verwenden sie jedoch heutzutage weitestgehend nicht mehr. Aus diesem Grund tragen die Metadaten zwar zur semantischen Struktur des HTML bei, können aber aus Sicht der Suchmaschinenoptimierung getrost vernachlässigt werden.(vgl. [Selb]) Weiter kann man bei Mikrodaten davon sprechen, „[...] dass sie auf Name/Werte-Paaren basieren. Jedes Mikordatenvokabular definiert eine Menge benannter Eigenschaften.“(vgl. [Sch11] S.174) Listing 4 zeigt Beispielhaft das *head*-Element eines HTML Dokuments mit vier, darin eingeschlossenen, *meta*-Elementen. Unter anderem wird mit dem ersten *meta*-Element der Zeichensatz näher definiert. Das *meta*-Element mit Namen *viewport*, dient dazu die Skalierung auf Mobilgeräten zu unterdrücken, damit die Seite sich an den *viewport* anpasst.

```

1 <head>
2   <meta charset="utf-8">
3   <meta name="viewport" content="width=device-width;" />
4   <meta name="keywords" content="Lorem ipsum">
5   <meta name="author" content="dolor sem it">
6 </head>

```

Listing 4: HTML5 *meta*-Element

Zwei weitere neue Elemente in HTML5 sind das *header*- und *footer*-Element. Zu Zeiten vor HTML5 wurden jene Bereiche einer Website durch *div*-Elemente mit dem *id*- oder *class*-Attribut entsprechend gekennzeichnet. Das ist jetzt nicht mehr nötig durch die beiden neuen Elemente. Üblich ist es im *header*-Element einer Website Komponenten wie das Logo, das Menü und den Titel unterzubringen. Im *footer*-Element dagegen werden Kontakt, Impressum und das Copyright aufgeführt. In Listing 5 ist die Platzierung des *header*- und *footer*-Elements innerhalb eines *body*-Elements einer Webseite zu sehen.

```

1 <body>
2   <header>
3     
4     <h1>Ueberschrift </h1>
5   </header>
6   <footer>
7     <a href="kontakt.html">Kontakt </a>

```

```
8    </footer>
9  </body>
```

Listing 5: HTML5 *header*- und *footer*-Element

Um ein HTML Dokument nach heutigen Maßstäben korrekt zu strukturieren wurden einige Elemente der Spezifikation hinzugefügt. So lässt sich die Navigation nun mit dem *nav*-Element umschließen wie in Listing 6 ab Zeile 3 zu sehen. „Das *section*-Element repräsentiert einen allgemeinen Abschnitt in einem Dokument oder einer Anwendung. Ein Abschnitt ist in diesem Kontext eine thematische Gruppierung von Inhalten, die üblicherweise unter einer Überschrift stehen. Beispiele für Abschnitte wären Kapitel, die verschiedene Tabs in einem Dialog mit Tabs oder die nummerierten Abschnitte einer wissenschaftlichen Arbeit.[...] Das *article*-Element repräsentiert eine abgeschlossene Einheit in einem Dokument, einer Anwendung oder einer Site, die unabhängig verbreitet oder wiederverwendet werden kann, z.B. in RSS-Feeds. Es könnte beispielsweise ein Forenbeitrag, ein Zeitschriften- oder Zeitungsartikel, ein Blog-Eintrag, ein Benutzerkommentar, ein interaktives Widget oder Gadget oder ein Element mit unabhängigem Inhalt enthalten. Das *aside*-Element repräsentiert einen Abschnitt einer Seite, der Inhalte enthält, die sich zwar auf den das *aside*-Element umgebenden eigentlichen Inhalt der Seite beziehen, aber als von ihm unabhängig betrachtet werden können. In Druckwerken werden derartige Abschnitte häufig als Seitenleisten dargestellt. Das Element kann für typografische Effekte wie herausgehobene Zitate oder Seitenleisten, für Werbung, für Gruppen von *nav*-Elementen und andere Inhalte verwendet werden, die als vom eigentlichen Inhalt der Seite getrennt betrachtet werden können.“[Sch11] Das neue *main*-Element ist zur Auszeichnung des Seitenhauptinhalts vorgesehen. Mit dieser Auszeichnung lässt sich z.B. mit Screenreadern direkt zum Hauptinhalt springen. Alle genannten Elemente sind in Listing 6 in ihrer vorgesehenen Reihenfolge abgebildet.

```
1  <body>
2    <header>
3      <nav>
4        <ul>
5          <li><a href="#link_1.html">Wiki</a></li>
6          ...
7        </ul>
8      </nav>
9    </header>
10   <main>
11     <article>
12       <h1>Ueberschrift</h1>
13       <p>Dies ist eine Beispiel HTML5-Seite</p>
```

```

14  </article>
15  <aside>
16    <section>
17      <h2>Kontakt</h2>
18      <ul>
19        <li><a href="link_1.html">Wiki</a></li>
20        ...
21      </ul>
22    </section>
23  </aside>
24  </main>
25  <footer>
26  </footer>
27 </body>

```

Listing 6: HTML5 Struktur Elemente

„Damit auch ältere Internet Explorer der Versionen 6-8 die neuen HTML5-Elemente darstellen können, kann ein kurzes JavaScript eingebunden werden. Am einfachsten ist es, dieses nicht auf dem eigenen Server vor zuhalten, sondern direkt von Google abzurufen. Dies hat überdies den Vorteil, dass es oft schon im Browser-Cache der Nutzer vorhanden ist. Der Aufruf erfolgt in einem *Conditional Comment*, der nur vom Internet Explorer kleiner als Version 9 (lt IE 9) verstanden wird. Alle andere Browser ignorieren dies als reinen Kommentar.“[Selc] In Listing 7 ist der beschriebene Rückfallmechanismus verdeutlicht.

```

1  <head>
2    <meta charset="utf-8">
3    <!--[if lt IE 9]>
4      <script src="//html5shim.googlecode.com/svn/trunk/html5.js"></
        script>
5    <![endif]-->
6    <title>HTML5-Seite mit Grundstruktur</title>
7  </head>

```

Listing 7: HTML5 Internet Explorer Fallback

Eingabefelder sind in den meisten Applikationen unabdingbar. Aus diesem Grund wurden in HTML5 eine Vielzahl an *input*-Elementen hinzugefügt. So müssen keine komplizierten Workarounds mit JavaScript oder anderen Skriptsprachen mehr verwendet werden. „Das typische HTML5-Formular unterscheidet sich nicht fundamental von seinen in HTML 4.01 oder XHTML 1 geschriebenen Gegenstücken. Alle alten Formularelemente sind noch da und verhalten sich weitgehend wie bisher. Die Neuerungen bestehen

aus einigen neuen Funktionen, Attributen und APIs und aus einer ganzen Reihe neuer möglicher Werte für das *type*-Attribut des *input*-Elements.“[Krö11] In Listing 8 sind beispielhaft einige neue Werte des *type*-Attributs ausgeführt. Eingabefelder des Typs *tel*, *email* oder *url* sind vom Verständnis her simple Texteingabefelder. Als wichtige Besonderheit lässt sich bei *email*- und *url*-Feldern aber ihre eingebaute Validation nennen. Verwendet man die Validierungs-API werden nur korrekte URLs bzw. E-Mails zugelassen. Für ein *tel*-Feld gilt dies nicht. Außerdem wird auf Smartphones und Tablet-Geräten die angezeigte Bildschirmtastatur entsprechend des erwarteten Eingabetyps für eine optimale Eingabe angepasst dargestellt.(vgl. [Krö11] S.178) So wird bei einer erwarteten E-Mail Adresse das @-Symbol direkt auf der Bildschirmtastatur als eigene Taste mit angezeigt, was normalerweise nicht der Fall ist. Einige der neuen Elementausprägungen besitzen noch zusätzliche Attribute die die Eingabemöglichkeit weiter einschränken und präzisieren können. Zu sehen in Zeile 5 von Listing 8 im *time*-Feld, bei dem die erwartete Zeit von 9:00Uhr bis 17:00Uhr nur einstellbar ist. *input*-Elemente können über das *required*-Attribut außerdem als Pflichtfeld markiert werden.

```

1 <body>
2   <input type="tel">
3   <input type="email">
4   <input type="url">
5   <input type="time"   min="09:00" max="17:00" >
6   <input type="date"   required="required">
7 </body>

```

Listing 8: HTML5 *input*-Element

Für multimediale Inhalte auf Webseiten wurden der Spezifikation passende Elemente hinzugefügt. Das *canvas*-Element „[...] stellt eine Fläche zur Verfügung, auf die mittels JavaScript dynamische Bitmap-Grafiken gezeichnet werden können. So lassen sich Animationen erstellen, Diagramme zeichnen, eigene Interface-Elemente kreieren und Videos manipulieren“[Krö11] Für Audio und Video Inhalte wurden die gleichnamigen Elemente geschaffen.

2.2 CSS3

// Allgemeiner Aufbau - Gestaltungssprache, Kern Element des WWW, Darstellung und Inhalt getrennt, Unterschiedliche Optik je nach Ausgabe Gerät
 // Syntax - Selektoren, Eigenschaften, Werte, Pseudoklassen
 // Listing 9

```

1  Selektor [, Selektor2 , ...]
2  {
3      Eigenschaft1: Wert1;
4      ...
5      EigenschaftN: WertN[;]
6  }
7  /* Kommentar – In eckigen Klammern stehen optionale Angaben */

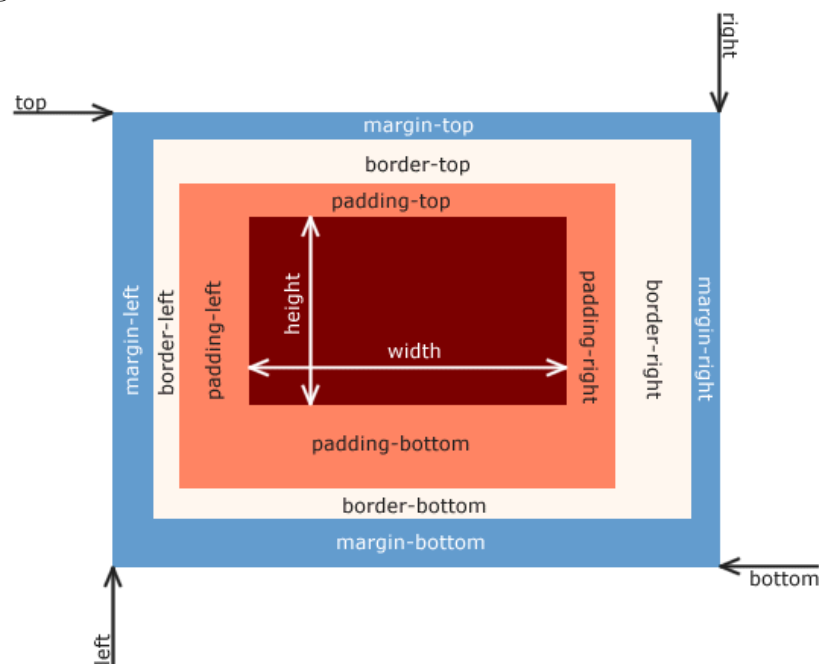
```

Listing 9: CSS3 Syntax Beispiel

```

// CSS-Box-Modell - margin, border, padding
// Abbildung 2

```

Abbildung 2: CSS-Boxmodell¹

```

// Medienspezifische Stylesheets (@media print, screen, ...)
// Listing 10

```

```

1  @media print {
2      body {
3          color: black;
4          background-color: white;
5      }
6      .navigation {
7          display: none;

```

¹Quelle: <http://commons.wikimedia.org/wiki/File:Boxmodell-detail.png>

```

8     }
9 }

```

Listing 10: CSS3 medienspezifisches Stylesheet

```

// Eigenschaftsspezifische Stylesheets (@media screen and (max-width:1024px))
// Listing 11

```

```

1 #inhalt {
2     width: 800px;
3 }
4
5 @media screen and (max-width: 1024px) {
6     #inhalt {
7         width: 600px;
8     }
9
10    aside {
11        display: none;
12    }
13 }

```

Listing 11: CSS3 eigenschaftsspezifisches Stylesheet

```

// Verzahnung mit HTML5 - link tag, style tag, html tag, @import innerhalb Stylesheet
// Listing 12 - Einbindung über Link Tag

```

```

1 <link rel="stylesheet" type="text/css" href="beispiel.css" />

```

Listing 12: Stylesheet Einbindung über *link*-Element

```

// Listing 13 - Einbindung über Style Tag

```

```

1 <head>
2     <title>Dokument mit Formatierungen</title>
3     <style type="text/css">
4         body { color: purple; background-color: #d8da3d; }
5     </style>
6 </head>

```

Listing 13: Stylesheet Einbindung über *style*-Element

```

// Listing 14 - Einbindung in HTML Tag

```

```
1 <span style="font-size: small;">Text</span>
```

Listing 14: Stylesheet Einbindung in *html*-Element

2.3 JavaScript

// Grundlagen - Geschichte, Sicherheit, aktueller Stand

// Listing 15 - Einbindung als separate Datei im Head

```
1 <script src="script.js" type="text/javascript"></script>
```

Listing 15: JavaScript Einbindung als separate Datei im *head*-Element

// Listing 16 - Einbindung in Skript Tag im Head und Body

```
1 <script type="text/javascript"></script>
```

Listing 16: JavaScript Einbindung in Skript Element im *head*- und *body*-Element

// Sprachelemente - Kommentare, Funktionen, Objekte

// Variablen - Dynamische Typisierung(Loose Typing), Case-Sensitive, ungarische Nomenklatur, spezielle Werte(*undefined*, *null*, *true*, *false*, *NaN*)

// Operatoren - *+*, *-*, ***, */*, zusätzlich *+* als Zeichenverkettung, In- und Dekrement, Zuweisung, Vergleich, *typeof*, Logisch

// Kontrollstrukturen - *if*, *switch*, *for*, *while* Anweisungen inklusive ihrer Varianten

// Document Object Model - Schnittstelle zum HTML Aufbau, W3C Spezifikation unterschiedlich implementiert, Knoten Beziehungen, Verarbeitung des DOM, Generierung von HTML durch Serialisierung, Listing 17 beschreiben und zur Baumstruktur hinleiten

// Listing 17

```
1 <table>
2   <thead>
3     <tr>
4       <th>Vorname</th>
```

```

5      <th>Name</th>
6    </tr>
7  </thead>
8  <tbody>
9    <tr>
10     <td>Donald</td>
11     <td>Duck</td>
12   </tr>
13 </tbody>
14 </table>

```

Listing 17: DOM5 Beispiel Definition

// Abbildung 3

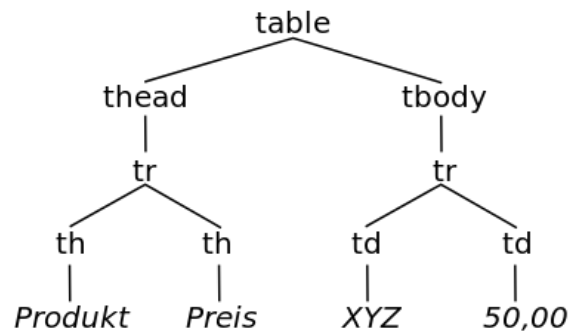


Abbildung 3: DOM Beispielbaum

// Ereignisse

// Übersicht einiger wichtiger Events

- onabort (bei Abbruch)
- onblur (beim Verlassen)
- onchange (bei erfolgter Änderung)
- onclick (beim Anklicken)
- ondblclick (bei doppeltem Anklicken)
- onerror (im Fehlerfall)
- onfocus (beim Aktivieren)
- onkeydown (bei gedrückter Taste)
- onkeypress (bei gedrückt gehaltener Taste)
- onkeyup (bei losgelassener Taste)
- onload (beim Laden einer Datei)
- onmousedown (bei gedrückter Maustaste)
- onmousemove (bei weiterbewegter Maus)
- onmouseout (beim Verlassen des Elements mit der Maus)

- onmouseover (beim Überfahren des Elements mit der Maus)
- onmouseup (bei losgelassener Maustaste)
- onreset (beim Zurücksetzen des Formulars)
- onselect (beim Selektieren von Text)
- onsubmit (beim Absenden des Formulars)
- onunload (beim Verlassen der Datei)

jQuery

// jQuery Bibliothek beinhaltet Elementselektion, Funktionen zum DOM, Animationen und Effekte, AJAX Funktionalitäten

// Selektoren

// Ereignisse - unterschiede zum JS Standard bei der Definierung, Einfachheit

// Übersicht der wichtigsten Funktionen zu Events

- .bind – Handler an Event binden
- .on – Handler an Event binden
- .blur – Ereignis, wenn ein Element den Fokus verliert
- .click – Klick mit der Maustaste
- .dblclick – Doppelklick mit der Maustaste
- .hover – Mauszeiger bewegt sich über ein Element
- .mousemove – Mauszeiger bewegt sich in einem Element
- .keypress – eine Taste der Tastatur wird gedrückt
- .keyup – eine Taste der Tastatur wird losgelassen
- .change – ein Formularfeld wird verändert

// DOM-Manipulation

// AJAX

2.4 ABAP

// Herkunft/Entstehung

// Grundlagen

// Wichtige Elemente (OpenSQL)

2.5 SAP UI5 Framework

2.5.1 Definition

// Aufbauend auf jQuery, AJAX, HTML5/CSS3 [Ant14]

2.5.2 Architektur

// Einführung in SAPUI5 S. 123 [Ant14]

// Abbildung 4

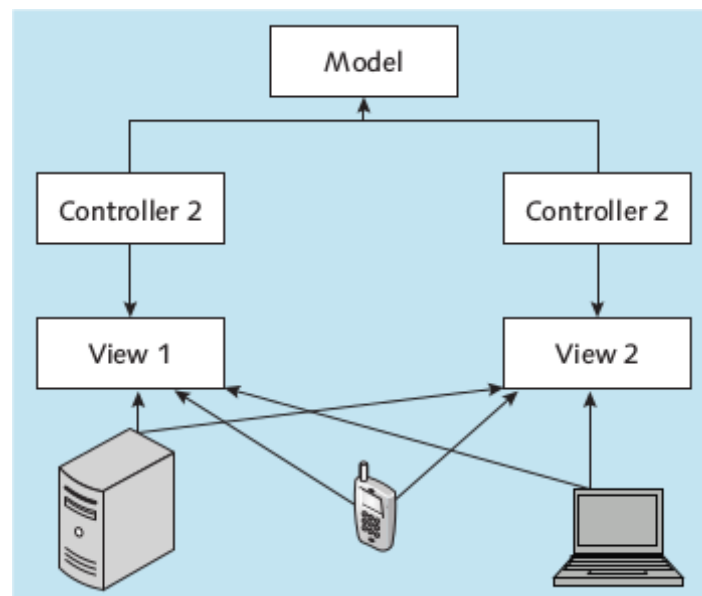


Abbildung 4: Model-View-Controller-Architekturmuster[Ant14]

2.5.3 OData Protokoll

// Einführung in SAPUI5 S. 168

// SAP Netweaver Gateway OData Services

3 Software Ergonomie

// Beleg für die Wichtigkeit von Software Ergonomie
// Kurze Übersicht über das Themenfeld Software Ergonomie
// Wichtigsten Aspekte nennen und näher erläutern

3.1 Definition

Kognitionspsychologie // Modellierung und Simulation von menschlichen Denk- und Wahrnehmungsprozessen

Arbeitsphysiologie, Industrieanthropologie // Beschäftigung mit grundlegenden menschlichen Fähigkeiten zur Informationsaufnahme und Informationsverarbeitung

Arbeitspsychologie // Untersuchung der Wechselbeziehungen zwischen Arbeit, deren Schnittstellen und psychischen Faktoren (unter anderem Arbeitszufriedenheit und -unlust)

3.2 DIN EN ISO 9241

// DIN Norm zur Software Ergonomie
// Die 7 Grundsätze der Dialoggestaltung:

- Aufgabenangemessenheit
- Selbstbeschreibungsfähigkeit
- Erwartungskonformität
- Fehlertoleranz
- Steuerbarkeit
- Individualisierbarkeit
- Lernförderlichkeit

DIN EN ISO 14915

// Erweiterung der ISO 9241

3.3 Analyse Methoden

Eye Tracking // Funktionsweise und Ergebnis

Mouse Clicking // Funktionsweise und Ergebnis

3.4 SAP Technologien in Bezug auf Software Ergonomie

3.4.1 Business Server Pages

// Business Server Pages (BSP) ist old school Technik

// geklaut von Java Server Pages (JSP)

3.4.2 Web Dynpro for ABAP

// Aktuelle Technik

// ABAP Code generiert HTML

// statischer und dynamischer Teil

3.4.3 SAP Fiori / SAP UI5 / SAP Screen Personas

// cutting edge

// aktuelle SAP UI Strategie

// SAP Präsi Chart Fiori/SP renew, etc. pp

// SAP Fiori einerseits Name des Themes/Guideline

// andererseits Bündel der gängigsten TAs/GPs als fertige

// Mobile First/Responsive Design Applikationen

// SAP UI5 - SAPs Framework zur Entwicklung von eigenen Applikationen im Fiori Style

// Nicht zu tief auf JS, HTML etc eingehen, dass kommt im nächsten Kapitel

// SAP SP - Zusätzliche Schicht um Standard Dynpro zu Personalisieren und so

4 Fallbeispiel SAP UI5

Lorem ipsum dolor sit amet.

4.1 Beschreibung

// Frontend - Browser, Elemente
// Backend - JSON, OData Model
// Analyse der wichtigen Arbeitsschritte

4.2 Hilfsmittel

4.2.1 Entwicklungsumgebung

// Kurze Beschreibung der Entwicklungsumgebung
// Sprich Eclipse, SE80, Chrome Dev-tools, Debugger
// Neptune Application Designer

4.2.2 UI Design und Prototyping

// Wireframing als Prototyping
// Abbildung Wireframesketcher

4.3 Implementierung

4.3.1 View

// Auszugsweise Coding bringen um bestimmte Elemente aus der Theorie zu zeigen
// Generellen Aufbau der Views erklären
// Kapselung wird dadurch verdeutlicht
// Listing 18

```
1  sap.ui.jsview("abat.Mockup.view.App", {  
2  
3    getControllerName: function () {  
4      return "abat.Mockup.view.App";  
5    },  
6  
7    createContent: function (oController) {  
8      // to avoid scroll bars on desktop  
9      this.setDisplayBlock(true);
```

```

10
11      // create app
12      this.app = new sap.m.SplitApp();
13
14      // load the master page
15      var master = sap.ui.xmlview("Master", "abat.Mockup.view.Master");
16      master.getController().nav = this.getController();
17      this.app.addPage(master, true);
18
19      // load the empty page
20      var empty = sap.ui.xmlview("Empty", "abat.Mockup.view.Empty");
21      this.app.addPage(empty, false);
22
23      // wrap app with shell
24      return new sap.m.Shell("Shell", {
25          title : "{i18n>ShellTitle}",
26          showLogout : false,
27          app : this.app
28      });
29  }
30  });

```

Listing 18: Root View der Applikation

```

// Master/Detail Applikation mit Fragment und Chart View Aufbau
// TODO: Visio Diagramm oder vergleichbares erstellen
// sap.ui.view
// --- sap.m.Shell
// --- sap.m.SplitApp
// --- sap.m.Page
// --- sap.m.Bar
// --- sap.m.Bar
// --- sap.m.List
// --- sap.m.ObjectListItem
// --- sap.m.Bar
// --- sap.m.Page
// --- sap.m.ObjectHeader
// --- sap.m.ObjectAttribute
// --- sap.m.ObjectAttribute
// --- sap.m.ObjectAttribute
// --- sap.m.ObjectAttribute
// --- sap.m.ObjectStatus

```

```
// — — — —- sap.IconTabBar
// — — — —- sap.IconTabFilter
// — — — —- sap.ui.core.Fragment
// — — — —- sap.ui.core.FragmentDefinition
// — — — —- sap.viz.ui5.Bar
// — — — —- sap.IconTabFilter
// — — — —- sap.ui.core.Fragment
// — — — —- sap.ui.core.FragmentDefinition
// — — — —- sap.viz.ui5.Bar
// — — — —- sap.m.Bar
```

4.3.2 Model und Controller

```
// die Verbindung von beiden Anhand von Coding zeigen
// TODO: OData Modell einbinden
// Listing 19
```

```
1  ...
2  // JSON Modell an die Root View binden
3  var oModel = new sap.ui.model.json.JSONModel("model/mock.json");
4  oView.setModel(oModel);
5
6  // OData Modell
7  var oModel = new sap.ui.model.odata.ODataModel(<URL>);
8  oView.setModel(oModel);
9
10 // I18N(Lokalisierung) Modell
11 var i18nModel = new sap.ui.model.resource.ResourceModel({
12     bundleUrl : "i18n/messageBundle.properties"
13 });
14 oView.setModel(i18nModel, "i18n");
15
16 // Geraetespezifisches Modell
17 var deviceModel = new sap.ui.model.json.JSONModel({
18     isPhone : jQuery.device.is.phone,
19     listMode : (jQuery.device.is.phone) ? "None" : "SingleSelectMaster",
20     listItemType : (jQuery.device.is.phone) ? "Active" : "Inactive"
21 });
22 deviceModel.setDefaultBindingMode("OneWay");
23 oView.setModel(deviceModel, "device");
24 ...
```

Listing 19: Component.js - Datenmodell an die Root View binden

4.3.3 Backend

```
// ABAP Stack der den RESTful Service bereitstellt zeigen  
// Beispielhafte Implementation des HTTP Responses
```

5 Analyse

5.1 Heatmap

// Angewandte Analyse mit Heatmap

5.2 UI-Objekte

// Mobile First/Responsive Design

5.3 PLATZHALTER

// PLATZHALTER

6 Schluss

Lorem ipsum dolor sit amet.

Zusammenfassung // Arbeitsgebiete, Produktions & Dienstleistungsbereiche
// Arbeitsergebnisse
// Projektziele, Projektergebnisse, Projekttermine
// Mitwirkungszeiträume
// Liste aller selbst wahrgenommen Aufgaben und Tätigkeiten
// Projektmeilensteine
// Ablauforganisation & Beteiligte
// Arbeitsformen, Arbeitsmittel, Arbeitsabläufe
// Kommunikations- / Informationsgewohnheiten
// Auswertung relevanter Literatur
// Themen aus Lehrveranstaltungen

Bewertung // Wesentliche Erkenntnisse und Erfahrungen
// Folgerungen und Konsequenzen
// Vorschläge für Verbesserung und Veränderung
// Auswirkungen auf persönliche Berufs- und Karriereplanung
// Bezug zum Studium
// hilfreiche Studieninhalte
// neu gewonnenes Interesse

7 Quellenverzeichnis

- [Ant14] ANTOLOVIC, Miroslav: *Einführung in SAPUI5: [Einführung in das SAP UI Development Toolkit für HTML5 ; moderne Benutzeroberflächen gestalten und erweitern ; Programmiermodell, Controls und UI-Elemente in der Praxis einsetzen]*. 1. Aufl. Bonn [u.a.] : Galileo Press, 2014 (SAP PRESS). – ISBN 9783836227537 und 3836227533
- [CG12] CLEMENS GULL, Stefan M.: *HTML5-Handbuch: [die neuen Features von HTML5 ; Webseiten für jedes Endgerät: Media Queries für mobile Devices ; so setzen Sie anspruchsvolle Web-Layouts mit HTML5 und CSS um ; umfangreicher Referenzteil für HTML und CSS zum Nachschlagen ; zukunftsorientierte Webseiten erstellen]*. 2., aktualisierte und erw. Aufl. Haar bei München : Franzis, 2012 (Know-how ist blau). – ISBN 3645601511 und 9783645601511
- [Krö11] KRÖNER, Peter: *HTML5: Webseiten innovativ und zukunftssicher ; [neu: Web Workers, File API, IE 9 uvm.]*. 2. Aufl. München : Open Source Press, 2011 (Professional reference). – ISBN 3941841343 und 9783941841345
- [Sch11] SCHULTEN, Lars: *Durchstarten mit HTML5: [tauchen Sie ein in die Webentwicklung der Zukunft]*. 1. Aufl. Köln [u. a.] : O'Reilly, 2011. – ISBN 9783897215719
- [Sela] SELFHTML: *Physische Auszeichnungen im Text*. <http://de.selfhtml.org/html/text/physisch.htm>. – Zugriff: 01.12.2014, Archiviert mit WebCite®: <http://www.webcitation.org/6UUqSW9ao>
- [Selb] SELFHTML: *Suchmaschinenoptimierung*. <http://wiki.selfhtml.org/wiki/Suchmaschinenoptimierung>. – Zugriff: 04.12.2014, Archiviert mit WebCite®: <http://www.webcitation.org/6UZA8MTZe>
- [Selc] SELFHTML: *Suchmaschinenoptimierung*. http://wiki.selfhtml.org/wiki/HTML/Tutorials/HTML5-Grundstruktur#Fallback_f.C3.BCr_.C3.A4ltre_Internet_Explorer. – Zugriff: 05.12.2014, Archiviert mit WebCite®: <http://www.webcitation.org/6Ub23bCc1>

Anhang

A GUI

Ein toller Anhang.

Screenshot

Unterkategorie, die nicht im Inhaltsverzeichnis auftaucht.

Erklärung

Hiermit versichere ich, dass ich meine Abschlussarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Datum:

.....

(Unterschrift)