LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
Department "Institut für Informatik"
Lehr- und Forschungseinheit Medieninformatik
Prof. Dr. Heinrich Hußmann

Bachelorarbeit

Untersuchung von HTML5 im Unternehmenseinsatz anhand einer Aufgaben-Management-Applikation

Analysis of HTML5 in business environment based on a human task list application

Juliane Keller
Juliane.Keller@campus.lmu.de

Bearbeitungszeitraum:	10. 6. 2011 bis 28. 10. 2011
Externe Betreuer:	Dr. Stefan Tausendpfund, Triantafyllos Kirtzakis
Verantw. Hochschullehrer:	Prof. Dr. Hans Jürgen Ohlbach

Zusammenfassung

Diese Arbeit untersucht neue Elemente von HTML5 und deren möglichen Einsatz in Unternehmen. Es werden dabei Neuheiten wie erweiterte Formulare, Video- und Grafikeinbettung sowie Technologien wie Geolocation, Drag-and-Drop, Web Storage und Web Sockets evaluiert. Außerdem werden die verschieden fortgeschrittenen Browserunterstützungen aufgezeigt. Um HTML5 für einen Unternehmenseinsatz zu analysieren, wurde ein Prototyp zum Aufgaben-Management erstellt. Repräsentativ für eine unternehmenstypische Applikation, kann der Nutzer im Prototypen zwei Prozesse verwenden. Ein Störprozess, bei dem Servicetechniker erstellte Störmeldungen bearbeiten, und ein Urlaubsantragsprozess, bei dem Manager offenstehende Urlaubsanträge genehmigen können. Der Prototyp ist dabei so ausgelegt, dass er mit einer Business Process Management Engine verbunden wird, welche die Verwaltung der Aufgaben übernimmt. Die Applikation verwendet clientseitig HTML, CSS, JavaScript und setzt neue HTML5-Technologien ein. Die Aufgabenliste wird über Web Sockets aktuell gehalten. Die Anzeige der Daten wird mit medialen Inhalten wie Videos und Landkarten angereichert. Die Einsetzbarkeit von HMTL5 in Unternehmen wurde anhand dieses Prototyps in unterschiedlichen Browsern auf PC und mobilen Endgeräten analysiert. Basierend auf dieser Untersuchung ließ sich HTML5 für einen Unternehmenseinsatz bewerten.

Abstract

This thesis examines new HTML5 elements and their possible use in business environment. New units like extended forms, embedding video and graphics and technologies, such as Geolocation, Drag and Drop, Web Storage and Web Sockets are evaluated. In addition, different advanced browser supports are presented. For analyzing HTML5 as a business application, a prototype for task management was programmed. Representative for a typical business application, users can apply two process types to the prototype. With a failure process, a service technician can edit generated failure reports and with a holiday evaluation process a manager can approve or decline open vacation requests. The prototype is set up for connecting to a Business Process Management Engine, which handles the administration of the tasks. On client-side the application uses HTML, CSS, JavaScript and new HTML5 technologies. The task list is kept up to date with Web Sockets. The display of data is enriched with multimedia, such as videos and maps. The applicability of HMTL5 in business environment has been analyzed based on this prototype in different browsers on PC and mobile devices. Based on this study HTML5 could be evaluated for corporate usage.

Dell GmbH
Business and Technology Consulting
Einsteinring 22
85609 Aschheim-Dornach
Tel: +49 (0)89 60 87 66 0
Fax: +49 (0)89 60 87 66 70
www.dell.de/services

Aufgabenbeschreibung Bachelorarbeit

Thema:
Untersuchung von HTML5 im Unternehmenseinsatz anhand einer Aufgaben-Management-Applikation

Aschheim, 10.06.2011

Motivation/Hintergrund

Heutzutage steigen die Anforderungen an die Benutzerfreundlichkeit von Web-Applikationen, was sich im Trend der steigenden Anzahl von Rich Internet Applikationen widerspiegelt. Der neue Standard HTML5 bietet die Möglichkeit, Web-Seiten vom Aufbau standardisierter, browserunabhängiger und benutzerfreundlicher zu gestalten. Speziell bei dynamischen Seiten-Inhalten, welche z.B. bei der Bearbeitung von Workflow-Aufgaben notwendig sind, ist mehr Flexibilität möglich.

Weiterhin gibt es bei vielen Web-Applikationen die Anforderung, wichtige Informationen ad hoc zu empfangen. Dies sind beispielsweise Störwarnungen in der Produktion oder Meldungen für einen mobilen Service-Mitarbeiter. Bisher wurden in solchen Fällen die Web-Seiten meist durch ständiges Polling (wiederholte Anfragen zum Server) aktuell gehalten.

Insbesondere bei prozessgestützten Applikationen mit einer Business Process Management Engine ist eine aktuelle Aufgabenliste notwendig. Mit HTML5 ist nun eine mögliche Alternative gegeben. Unter Verwendung von Web Sockets, was ein wesentliches neues Merkmal von HTML5 ist, können Webseiten eine bidirektionale Verbindung zwischen einer Webseite und dem Server aufbauen. Es können somit Informationen in real-time übermittelt werden.

Insgesamt werden mit HTML5 Anwendungen nutzerfreundlicher und können effektiver realisiert werden.

Aufgabenbeschreibung (Bachelorarbeit)

Im Rahmen der Bachelorarbeit sollen zunächst die Neuerungen von HTML5, wie unter anderem die Funktionalitäten bzgl. standardisierten Seiten-Aufbaus und Web Sockets, untersucht werden. Hierbei soll ebenfalls die Einsatzfähigkeit dieser neuen Standards in aktuellen Browserversionen geprüft werden.

Aufbauend auf die neuen HTML5-Konzepte soll prototypisch eine Applikation zum Aufgaben-Management entwickelt werden, welche folgende Punkte beinhaltet:

- Definition von einfachen Standard-Prozessen
- Anzeige der Aufgabenliste des angemeldeten Benutzers
- Synchronisierung der Aufgabenliste über Web Sockets
- Erstellung von neuen Aufgaben
- Bearbeiten und Weiterschaltung von Aufgaben

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig angefertigt, alle Zitate als solche kenntlich gemacht sowie alle benutzten Quellen und Hilfsmittel angegeben habe.

München, 5. September 2011

. . . . . .

Danksagung

Hiermit möchte ich mich bei Prof. Dr. Hans Jürgen Ohlbach bedanken, der sich bereit erklärt hat, meine Bachelorarbeit zu bewerten. Ohne ihn wäre diese externe Bachelorarbeit mit diesem Thema nicht möglich gewesen.

Außerdem möchte ich mich besonders bei meinen beiden Betreuern von Dell Services Dr. Stefan Tausendpfund und Triantafyllos Kirtzakis bedanken, die mir die Möglichkeit gaben im Unternehmensumfeld meine Bachelorarbeit zu schreiben. Dabei standen sie mir trotz Zeitmangels stets mit guten Ratschlägen zur Seite.

Inhaltsverzeichnis

1 Einleitung **1**

2 Untersuchung neuer HTML5-Technologien **3**
2.1 Neue semantische Elemente . . . 3
2.2 Optimierter Umgang mit Nutzereingaben . . . 5
2.2.1 Neue Eingabetypen mit Validierungsfunktionen . . . 5
2.2.2 Drag-and-Drop . . . 6
2.3 Lokalisierung des Nutzers mit Geolocation . . . 7
2.3.1 Allgemeine Funktionalität . . . 7
2.3.2 Geolocation-Unterstützung der Browser . . . 8
2.4 Neue Formen der Mediendarstellung . . . 8
2.4.1 Einbettung von Video . . . 9
2.4.2 Erstellen von Grafiken und Animationen . . . 10
2.5 Lokale Speicherung mit Web Storage . . . 11
2.6 Echtzeitübertragung von neuen Daten . . . 12
2.6.1 Grundlegende Funktionalität von Web Sockets . . . 12
2.6.2 Sicherheitsprobleme . . . 13
2.7 HTML5 auf mobilen Endgeräten . . . 13
2.8 Zusammenfassung und Fazit . . . 14

3 Entwicklung einer Aufgaben-Management-Applikation **15**
3.1 Zielsetzung . . . 15
3.2 Vorgehensweise . . . 17
3.3 Beschreibung der Applikation . . . 18
3.4 Funktionalitäten und ihre Realisierung . . . 19
3.4.1 Anlegen von Aufgaben . . . 19
3.4.2 Aktualisierung der Informationen durch Web Sockets . . . 20
3.4.3 Signal bei neuem Task . . . 22
3.4.4 Detailangabe zu Aufgaben . . . 23
3.5 Browserunterstützungen des Prototyps . . . 23
3.6 Die Aufgaben-Management-Applikation offline . . . 24
3.7 HTML5 in Unternehmen . . . 25

4 Ausblick **27**

5 Anhang **29**

Abbildungsverzeichnis **II**

Tabellenverzeichnis **IV**

Inhalt der beigelegten CD **VI**

Literaturverzeichnis **VIII**

1 Einleitung

Handys gewinnen in Form von Smartphones zunehmend an Bedeutung und stellen eine Ergänzung zu Computer dar. Das gilt insbesondere für Unternehmen, die ihren Mitarbeitern auch Smartphones zur Verfügung stellen. Dieser Wandel bringt neue Herausforderungen mit sich. Während Computer zu fast 93% mit dem Betriebssystem Windows laufen[30], besteht bei mobilen Endgeräten eine breite Verteilung der Betriebssysteme[1]. In Unternehmen stellt sich damit die Frage, ob für firmeninterne Zwecke genutzte Anwendungen für mobile Geräte nativ oder als Rich Internet Application erstellt werden sollen. Sofern letztere genügend Funktionen bereitstellen, bieten sie vor allem die Vorteile der Plattformunabhängigkeit und Kostenersparnis im Lebenszyklus. HTML5 bildet eine fundamentale Basis für die Entwicklung solcher Applikationen.

Diese Weiterentwicklung von HTML gibt es seit 2004, als die Browserhersteller Apple, Mozilla und Opera die Web Hypertext Application Technology Working Group, kurz WHATWG[1, Seite 20], gründeten. Ziel war es, neue Webtechnologien zu entwickeln. So sollten mit HTML, das ursprünglich für Webseiten im Sinne von Webdokumenten konzipiert wurde, nun auch Webanwendungen mit zahlreichen Interaktionsmöglichkeiten erstellt werden können. Die WHATWG rief das Projekt “Web Applications 1.0” ins Leben, das später zu HMTL5 wurde. Mitte 2009 wurde auch vom World Wide Web Consortium (W3C) eine eigene Arbeitsgruppe für HTML5 erstellt. Seitdem arbeiten beide Gruppen an der Definition des HMTL5-Standards.

Im Januar 2010 wird HTML5 schlagartig publik[2]. Zum einen verkündet das Videoportal YouTube in einem Testmodus Videos mit HTML5 darzustellen[4] und zum anderen stellt Steve Jobs, damaliger Geschäftsführer von Apple Inc., das neue iPad vor, das HMTL5 der Technologie Flash von Adobe vorzieht und Flash grundsätzlich nicht unterstützt. Google Inc. und Apple Inc. wenden sich bewusst von Flash ab - zu Gunsten von HTML5. Damit wird deutlich, dass diese bedeutsamen Konzerne in HTML5 die Technologie der Zukunft sehen. Da beide Firmen jeweils Betreiber von Betriebssystemen für mobile Endgeräte sind (Google von Android, Apple von iOS) scheint HTML5 auch aus ihrer Sicht für den mobilen Sektor geeignet.

Im Laufe der Zeit haben sich viele HTML5-Technologien entwickelt, mit denen man den modernen Ansprüchen an das Web gerecht werden kann. Das W3C hat das Ziel, diese Technologien zu standardisieren und damit eine einheitliche Browserunterstützung zu schaffen.

Den Begriff HTML5 gibt es offiziell nicht mehr. Im Januar 2011 verkündete der Leiter der WHATWG Ian Hickson, dass HMTL5 nun ein “lebender Standard”, mit dynamischen Entwicklungsprozess ist[5], dementsprechend keine Versionsnummer braucht und von nun an wieder HTML heißt. Dies eröffnet die Möglichkeit die Bedeutung des Begriffs in dieser Arbeit neu zu belegen: HTML5 schließt hier alle neuen Technologien ein, die von der WHATWG oder dem W3C zur Verbesserung der Möglichkeiten Webapplikationen zu entwickeln, dienen. Somit werden in dieser Arbeit auch Technologien untersucht, die im engen Zusammenhang mit HTML5 in Verbindung gesetzt werden.

Ziel ist es, zu untersuchen, ob HTML5 in Unternehmen effizient eingesetzt werden kann. Dazu wird zunächst geprüft, ob die Sprache grundsätzlich schon vor der offiziellen Standardisierung 2014 für das Erstellen von Applikationen bereit ist. Denn bestimmte Funktionen werden schon in Browsern unterstützt und manche Spezifikationen sind nahezu Standard. Diese Bachelorarbeit dient dazu, den tatsächlichen Stand abschätzen zu können. Der erste Teil beschäftigt sich mit dem Umgang von HTML5 im allgemeinen Web-Gebrauch. Dazu zählen normale Webseiten und -anwendungen, die im World Wide Web zu finden sind.

Im zweiten Teil der Arbeit wird untersucht, ob HTML5 von Unternehmen zur Entwicklung von Rich Internet Anwendungen eingesetzt werden sollte. Zu diesem Zweck wird eine hierfür entwickelte Aufgaben-Management-Applikation vorgestellt und deren Einsetzbarkeit analysiert.

[1] nach Gartner.com [31]

[2] Nach Google Trends ist die Anzahl der Suchanfragen nach “HTML5” Ende Januar schlagartig gestiegen (Siehe Anhang Abbildung 5.1).

Auf Basis dieser Untersuchungen wird eine Prognose für HTML5 abgeleitet.

2 Untersuchung neuer HTML5-Technologien

Das Interesse der breiten Masse an HTML5 orientiert sich im Wesentlichen an den neuen Medienformen, die nun mit HTML5 dargestellt werden können. Dies verdeutlichen die Anzahlen der Suchanfragen in Google nach Google Trends[33]. Abbildung 2.1 zeigt ein deutlich stärkeres Interesse an dem HTML5 `<canvas>`-Element und an dem neuen `<video>`-Tag, als an Spezifikationen wie Web Sockets und Web Storage. Dennoch haben letztere Technologien gegenüber aktuellen Lösungen entscheidende Vorteile. Im Rahmen dieser Bachelorarbeit wird eine Auswahl an den bisher bekanntesten und medienpräsentesten Technologien untersucht. Dabei handelt es sich um die neuen semantischen Elemente, mit denen eine HTML-Seite besser strukturiert werden kann. Außerdem werden die Funktionalitäten von Web Storage, Web Sockets, Drag-and-Drop, Geolocation, Darstellung von Grafiken und Animationen mit `<canvas>`, neue Formularfunktionen und Einbettung von Videos vorgestellt. Für eine optimale Evaluierung wurde eine HTML5-Webseite erstellt, welche die Technologien beschreibt und implementiert. Anhand dieser konnten die neuesten Versionen der fünf bekanntesten Browser[3] auf ihre jeweilige Unterstützung hin untersucht werden. Im Folgenden werden die ausgewählten Technologien beschrieben und die Erfahrungen mit der Evaluierungswebseite dargelegt. So kann ein Fazit gezogen werden, wie gut die neuen HTML5-Elemente bereits einzusetzen sind.

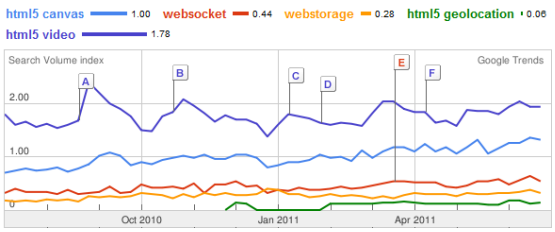


Abbildung 2.1: Häufigkeiten von Suchanfragen unterschiedlicher HTML5-Begriffe nach Google Trends

2.1 Neue semantische Elemente

Semantische Elemente sind Hauptbestandteil von HTML 4.01. Da zur Zeit der Entstehung von HTML 4.01 andere Ansprüche an das Web gestellt wurden als heute, werden mit HTML5 neue Funktionen eingeführt. Die im Folgenden vorgestellten Elemente dienen einer Anpassung an die heutigen Anforderungen an die Struktur einer Webseite.

Eine Webseite hat in der Regel einen Kopf-, Fuß- und einen Navigationsbereich an den Rändern des eigentlichen Inhalts. Bisher besteht eine HTML Seite meist aus `<div>`-Blöcken, deren Attribut `id` den Inhalt des jeweiligen Abschnittes beschreibt. Ein Beispiel dafür ist die Startseite der Homepage der Ludwig-Maximilians-Universität München (LMU)[6] (z.B. `<div id='Hauptnavigation'`). Für Navigationen gibt es das neue `<nav>`-Element, für den Kopfbereich das Element `<header>` und für den Fußbereich den Tag `<footer>`. Außerdem kann der Inhalt in bestimmte Segmente eingeteilt und mit `<section>` oder `<article>` gekennzeichnet werden. So gibt es nun Tags für Bestandteile, die es fast auf jeder Webseite gibt und ein standardisierter Seitenaufbau ist möglich. Es gibt weitere Elemente, die Verbesserungen bieten. Durch das neue `<hgroup>` wird die Beschränkung auf sechs Überschriftenebenen aufgehoben. Nebenbereiche können durch `<aside>` beschrieben werden und es werden die selbsterklärenden Tags `<time>`, `<details>` und `<summary>` eingeführt. Inhalte, die für sich alleine stehen, können nun in einem `<figure>`-Element stehen und mit `<figcaption>` eine Überschrift enthalten. Geeignete Inhalte dafür wären zum Beispiel ein Bild oder Video.

Kritik den neuen Elementen gegenüber wird vor allem in Bezug auf die Spezifikationen laut. Die Elemente `<section>` und `<article>` unterscheiden sich nur in dem Punkt, dass `<article>` das Attribut `pubdate` für den Publikationszeitpunkt erlaubt[7]. Hier wäre eine Zusammenlegung

[3] Im Rahmen der Bachelorarbeit wurden folgende Browserversionen getestet: Google Chrome 12.0.742.91; Firefox 5.0; Windows Internet Explorer9 9.0.8112.16421IC; Opera 11.11; Safari 5.0.5 (7533.21.1). Alle folgenden Details zu Browser beziehen sich auf diese Version, sofern nichts anderes angegeben ist.

der beiden Elemente sinnvoller. Außerdem sind scheinbar intuitiv verwendbare Elemente wie `<nav>` nicht so leicht wie vorgesehen einsetzbar, da dieses zum Beispiel nur für Hauptnavigationen vorgesehen ist[8].

Inhaltlich verändert sich die Seite durch diese Neuerungen nicht. Nur eine neue semantische Struktur wird erstellt und erleichtert es, sich als Webentwickler einen besseren Überblick über die Seite zu verschaffen. Dies hat vor allem für größere Projekte mit mehreren Webentwicklern den Vorteil, dass man sich auch in fremden Codes einfach zurechtfinden kann. Außerdem können Suchmaschinen durch die Strukturierungselemente den Seiteninhalt besser einordnen. So ist dann bekannt, dass der Text in einem `<aside>`-Block weniger den Inhalt der Seite repräsentiert, als der eines `<article>`-Elements. Es handelt sich hierbei nur um semantische Elemente. Dies wird durch die Tatsache bewusst, dass die Position der Elemente auch weiterhin vom Webentwickler selbst mit Stylesheets festgelegt werden muss. Zum Beispiel trifft die intuitive Vorstellung, dass der Header auf der Homepage automatisch oben und der Footer unten ist, nicht zu.

Neben den neuen strukturellen gibt es auch neue funktionale Elemente. Abbildung 2.2 zeigt einen Ausschnitt aus der Evaluierungsseite[4], die hier mit Google Chrome angezeigt wird. Wie man erkennen kann, erfordern diese Tags eine bestimmte Darstellung von den Browsern. Tabelle 2.1 zeigt jedoch, dass die Browserunterstützungen bisher unterschiedlich sind. Die Elemente `<meter>` und `<progress>` sollen als Fortschrittsanzeigen dargestellt werden. Bisher bieten nur Opera und Chrome eine grafische Darstellung. Das Element `<progress>` unterscheidet sich in Opera jedoch nicht von `<meter>`, obwohl es einen Prozess und keinen statischen Fortschrittsgrad darstellen soll. Mit `<mark>` kann ein Text markiert werden. Dies eignet sich für Wörter, die im Text hervorgehoben werden sollen, wie zum Beispiel nach einer erfolgten Suche. Das Element `<ruby>` wird für Annotationen über einem Text eingeführt. Dabei enthält ein `<ruby>`-Element einen Textknoten mit dem eigentlichen Inhalt und einen `<rt>`-Knoten, der die Annotation beinhaltet, die über dem normalen Text stehen soll. Das `<ruby>`-Element kann ein `<rp>`-Kind enthalten, in dem ein Alternativtext platziert ist, für den Fall, dass der Browser das `<rt>`-Element nicht unterstützt. Das Element `<command>` funktioniert wie eine

Abbildung 2.2: Beispiel aus der Evaluierungsseite: Darstellung bestimmter neuer Elemente

	Chrome	Firefox	IE	Opera	Safari
`<command>`	nein	nein	ja	nein	ja
`<mark>`	ja	ja	ja	ja	nein
`<meter>`	ja	nein	nein	ja	nein
`<progress>`	ja	nein	nein	tlw.	nein
`<ruby>`	ja	nein	ja	nein	ja
`<wbr>`	ja	nein	ja	nein	ja

Tabelle 2.1: Browserunterstützung von neuen semantischen Elementen in HTML5

Schaltfläche, wird aber bisher nur vom Internet Explorer und Safari realisiert. `<wbr>` dient optionalen Zeilenumbrüchen, das für ein langes Wort ohne Trennzeichen benutzt werden kann. Der

[4] Im Anhang befindet sich der gesamte Screenshot der Evaluierungsseite (Anhang 5.2)

Browser fügt dann je nach Fensterbreite Zeilenumbrüche an den Stellen ein, an denen sich ein Trennzeichen oder das `<wbr>`-Element befinden. Die neuen semantischen Elemente dienen einem standardisierten Seitenaufbau und ermöglichen dem Webentwickler das Einbinden von häufig genutzten Grafikelementen, wie beispielsweise eines Ladebalkens.

2.2 Optimierter Umgang mit Nutzereingaben

Der Unterschied zwischen Webdokumenten und Webanwendungen besteht im Wesentlichen in der Interaktion mit der Webseite. In Webapplikationen können Nutzer Seiteninhalte durch verschiedene Formen von Eingaben dynamisch ändern. Neben Mausinteraktionen, wie Drag-and-Drop, können auch Eingaben in Formulare den Seiteninhalt verändern. Mit HTML5 werden viele neue Möglichkeiten gegenüber den alten HTML-Formularen eingeführt, vor allem im Bereich clientseitige Validierung. Außerdem liefert HMTL5 eine Drag-and-Drop-API mit der sich Elemente innerhalb einer Seite, aber auch Elemente von verschiedenen Seiten verschieben lassen. Im Folgenden werden die HMTL5-Formularerweiterungen inklusive neuer Eingabetypen und Validierung im Browser und anschließend die Drag-and-Drop-API vorgestellt.

2.2.1 Neue Eingabetypen mit Validierungsfunktionen

Mit der Definition von HTML5 wurden Formulare in zwei Bereichen erweitert: neue Eingabetypen und deren browserinterne Validierung. Eingabefelder können nun Emailadressen, URLs, Telefonnummern, Suchanfragen, Zahlen mit bestimmten Werten, jegliche Formen von Datumsangaben und sogar einen Farbwert erwarten. Es ist auch möglich, sich ein eigenes Eingabeformat mit regulären Ausdrücken zu definieren. Diese Typen ermöglichen eine passende Darstellung der Eingabefelder. Der Browser kann Felder des Typs `search` wie ein browserspezifisches Suchfeld anzeigen und Eingaben für den Typ `date` und `color` können durch interaktive Kalender und Farbverläufe vereinfacht werden. Browser für mobile Endgeräte können die eingeblendete Tastatur an den fokussierten Eingabetyp anpassen und zum Beispiel bei der Eingabe einer Email die Taste des @-Zeichens im Vordergrund darstellen.

Eine Validierung im Browser hat ebenfalls mehrere Vorteile. Zum einen gewinnt man Performanz, da weniger Kommunikation zum Server besteht. Zum Anderen spart man sich Entwicklungsaufwand und -kosten, da Standardvalidierungen vom Browser bereits übernommen werden. Durch angepasste Eingabeformen wie Kalender können falsche Datumseingaben verhindert werden. Außerdem umgeht man die Gefahr, dass clientseitige Validierungen durch eine Deaktivierung von JavaScript nicht ausgeführt werden. Im Vergleich zu serverseitigen Validierungen hat der Nutzer ein schnelleres Feedback. Mit der API kann der Webentwickler Validierungen an eigene Anforderungen anpassen. Er kann unter anderem abfragen, ob eine Eingabe falsch ist und welche Art von Fehler gemacht wurde. Außerdem ist es möglich, eigene Kriterien zu definieren und bei invaliden Eingaben individuelle Fehlernachrichten browsertypisch anzeigen zu lassen oder die fehlerhafte Eingabe durch CSS-Pseudoklassen zu gestalten. Allgemein bieten die HTML5-Formularfunktionen viele Vorteile.

Eingabetyp	Chrome	Firefox	IE	Opera	Safari
`email`	ja	ja	nein	ja	ja
`url`	ja	ja	nein	ja	ja
`number`	ja	nein	nein	ja	ja
`range`	ja	nein	nein	ja	ja
`date`	ja	nein	nein	ja	ja
`color`	ja	nein	nein	ja	ja

Tabelle 2.2: Browserunterstützung der neuen Eingabetypen

Diese Eingabetypen sind mit der Evaluierungsseite getestet worden. Dies hat in den aktuellen Browserversionen keine zufriedenstellende Umsetzung ergeben. Wie man an Tabelle 2.2 sieht, unterstützen Firefox und Internet Explorer keine bis wenig neue Eingabetypen. Kennt ein Browser einen Eingabetypen nicht, interpretiert er das Feld als Typ `text`. Das macht für die Darstellung meist keinen Unterschied (außer für den Typ `range`, der als Slider dargestellt werden sollte), ermöglicht aber keine Validierung. Safari validiert die Eingaben zwar, feuert auch das `invalid`-Event, akzeptiert aber das Senden falscher Eingaben und teilt dem Nutzer nicht mit, dass die Eingabe falsch ist. In diesem Fall müssten Styleattribute (`:invalid`) dem Nutzer eine falsche Eingabe signalisieren und Methoden das Absenden falscher Eingaben verhindern.

Die Darstellung wird browserintern geregelt. Abbildung 2.3 zeigt die Evaluierungsseite, dargestellt mit Opera, der bestimmte Eingabeoptionen für den Nutzer besonders übersichtlich darstellt. Als Beispiel wurde in ein Eingabefeld mit dem Typ `url` das Wort 'google' eingegeben. Der rote Hintergrund wurde mit CSS erstellt, der solange rot ist, bis die Eingabe valide wird. Die Fehlermeldung dagegen ist browserintern und erscheint, wenn eine falsche Eingabe abgeschickt werden soll. Außerdem sieht man Operas optimale Darstellung für Farb- und Datumseingaben. Klickt der Nutzer in ein entsprechendes Feld, erscheinen Popups für eine bequeme Auswahl. Durch Mangel an durchgehender Unterstützung ist heute mit HTML5 noch keine Validierung gewährleistet. Um mit HTML5 clientseitig zu validieren, entsteht noch erhöhter Implementierungsaufwand, da Alternativvalidierungen erstellt werden müssen. Somit können die Vorteile der HTML5 Formulare noch nicht vollständig genutzt werden.

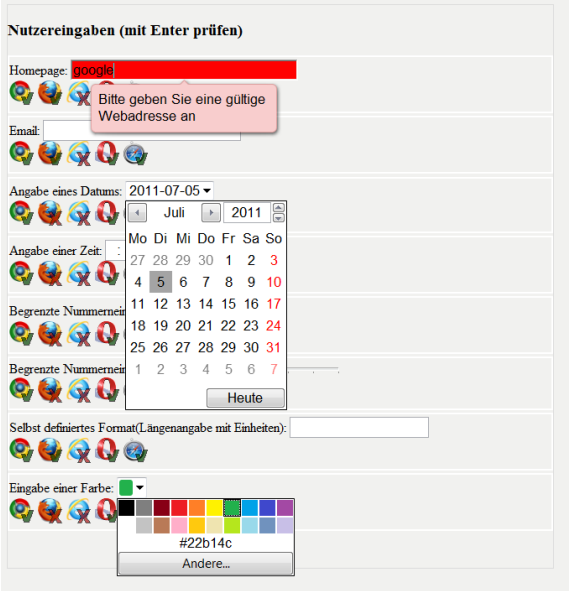


Abbildung 2.3: HTML5-Formular in Opera

2.2.2 Drag-and-Drop

Ian Hickson, Leiter der WHATWG, schreibt über die Drag-and-Drop API auf Twitter: "Die drag-and-drop API ist schrecklich, aber eine Sache spricht für sie: IE6 implementiert sie und auch Safari und Firefox"[29]. Trotzdem ist die Drag-and-Drop API ein wichtiges Element der Neuerungen von HTML5. Die Aussage bezieht sich auf die Tatsache, dass die API aus dem Internet Explorer 5 stammt und nur deshalb in HTML5 übernommen wurde, weil Safari und Mozilla die API bereits adaptiert hatten. Dennoch hat sie einen großen Vorteil: Interoperabilität zwischen verschiedenen Browsern[1, Seite 213].

Für die Implementierung einer HTML5 Drag-and-Drop-Anwendung gibt es sieben Events, die ausgelöst werden, wenn ein Objekt gezogen, fallen gelassen, oder über bestimmte Bereiche bewegt wird. So kann man zum Beispiel beim Auslösen des Events `ondragstart` speichern, was gezogen wird, und beim Event `ondrop` das Element, das gezogen wird, beim Fallenlassen an einer neuen Stelle darstellen und am ursprünglichen Punkt löschen. Dies ist bereits eine einfache Drag-and-Drop-Anwendung[5]. Anlass zur Kritik an der Übersichtlichkeit kann man in dem Punkt anmerken, dass das Event `ondragover` einer Aktion zugewiesen werden muss, ansonsten wird im Bereich zum Ablegen kein Event `ondrop` gefeuert[6].

[5] siehe CD/Quellcode/HTML5Evaluierungsseite/draganddrop.html

[6] Dies wurde im Rahmen von Tests mit der Evaluierungsseite und den untersuchten Browserversionen festgestellt

Diese Funktionalität hätte auch mit einer JavaScript-Bibliothek wie jQuery realisiert werden können. Der große Vorteil der HTML5 Drag-and-Drop-API ist Interoperabilität. Für den Fall, dass diese API Standard wird, und jeder Browser die API unterstützt, sind Drag-and-Drop-Operationen von Browser zu Browser möglich, selbst wenn es sich dabei um unterschiedliche Browser handelt[1, Seite 227]. Außer der Kommunikation zwischen Browsern kann auch eine Drag-and-Drop-Kommunikation zwischen Desktop und Browser mit derselben API stattfinden. So können File-Uploads beispielsweise mit Drag-and-Drop implementiert werden.

Die Implementierungen der API variieren von Browser zu Browser leicht und so ist bisher kein Drag-and-Drop zwischen unterschiedlichen Browsern, sondern nur zwischen zwei Fenstern eines Browsers möglich[7]. In der Evaluierungsseite wurde ein kleines Puzzle mit Drag-and-Drop implementiert und die Möglichkeit, ein Bild vom Desktop in den Browser zu ziehen und darzustellen. Diese Beispiele funktionieren bis auf Opera in allen Browsern.

2.3 Lokalisierung des Nutzers mit Geolocation

Anfangs war die Geolocation API noch offiziell Teil der HTML5-Spezifikation, wurde dann aber später als eine selbstständige ausgelagert. Ende 2010 hat sie den Status “Candidate Recommendation” erhalten und ist damit kurz davor offizieller Webstandard zu werden[9]. Mit Methoden der API kann der Webentwickler die Koordinaten des Nutzers feststellen und diese für vielerlei interessante Dinge einsetzen. Zum Beispiel können die Position des Nutzers, eine Wegbeschreibung, Geschäfte oder andere Orte in einer Karte angezeigt werden. Außerdem kann die Position für ortsgebundene Informationen, wie Wetter, genutzt und bei Erstellung einer Datei, die Bezug zu einem Ort hat, beispielsweise einem Foto, gespeichert werden. Anwendungsfälle dieser Art werden auch in der Geolocation API von W3C für Geolocation angegeben[10]. In den nächsten zwei Punkten wird die API vorgestellt und auf die Unterstützung der Browser eingegangen.

2.3.1 Allgemeine Funktionalität

Mit einfachen Methodenaufrufen kann das Gerät des Nutzers lokalisiert werden. Aus Datenschutzgründen legt das W3C in der Spezifikation fest, dass die Position nur nach ausdrücklichem Einverständnis des Nutzers erhoben und weitergeleitet werden darf. So hat der Nutzer selbst die Wahl, ob er durch Bekanntgabe seiner Position die Vorteile einer Lokalisierung nutzen möchte.

Wie der Browser die Position des Nutzers bestimmt, bleibt ihm selbst überlassen[9]. Unterschiedliche Systemvoraussetzungen und landschaftliche Gegebenheiten der aktuellen Position erfordern unterschiedliche Technologien. Steht ein GPS-Gerät zur Verfügung, können die Koordinaten über Satelliten ermittelt werden, die eine sehr hohe Genauigkeit ermöglichen. Allerdings ist das Signal eher langsam, in Großstädten oft schlecht und in Gebäuden teilweise gar nicht vorhanden. Diese Schwäche ist die Stärke der Lokalisierung durch WiFi-Signale. Es kann anhand von nahegelegenen WiFi-Zugangspunkten und Signalstärken ein relativ genauer Standort aus einer Datenbank ermittelt werden, allerdings abhängig von der Dichte der Router an der gesuchten Position. In Großstädten gibt es viele, in ländlichen Gegenden dagegen weniger Wireless Lan Router. Aus diesem Grund ist die WiFi-Positionierung in ländlichen Gegenden meist ungenau. Manche Browser, wie Opera nutzen zur Positionierung Google Location Services (GLS), denen die ermittelten Standortdaten übertragen werden[11]. GLS sendet dann eine geschätzte Längen- und Breitenangabe zurück. Eine weitere Möglichkeit ist die Lokalisierung über die IP-Adresse. Jeder IP-Adressen-Block repräsentiert einen bestimmten geografischen Bereich. So kann aber nur eine grobe Position festgestellt werden, die meist eine Stadt repräsentiert[12]. Für viele Anwendungen ist dies zu ungenau. Alternativ kann auch auf eine gespeicherte Position des Geräts zugegriffen werden, wenn keine Koordinaten ermittelt werden konnten.

[7] Demonstration dazu ist unter `http://html5-buch.de/html5demos/dragndrop/` zu finden

Der Umgang mit der API ist sehr einfach und benutzerfreundlich. Ruft man `navigator.geolocation.getCurrentPosition(erfolg, misserfolg)` auf, versucht der Browser die Position zu ermitteln. Gelingt dies, wird die Methode `erfolg(position)` (erster Parameter) aufgerufen, wenn nicht, dann die Funktion `misserfolg()` (zweiter Parameter). In dem Parameter `position` sind viele Informationen zur Position enthalten, von geografische Breite und Länge, über Höhe, bis zur Geschwindigkeit, mit der sich das Gerät gerade bewegt. In der Applikation, die im Zuge des Testens von HTML5 entwickelt wurde, erfolgt die Positionierung wie oben beschrieben. Außerdem werden Anweisungen in Form eines dritten Parameters für die Methode `getCurrentPosition(...)` übergeben. Mit `enableHighAccuracy: true` wird der Browser angewiesen, auch auf Kosten der Performanz das Lokalisierungsverfahren anzuwenden, welches die genauste Position liefert. Es wurde zudem mit `timeout:10000` eine Zeitspanne von zehn Sekunden festgelegt, nach der die Misserfolgs-Methode aufgerufen werden soll, falls bis dahin noch keine Position bestimmt werden konnte. Das Timeout bezieht sich jedoch nicht auf den Benutzer. Wünschenswert wäre ein Zeitlimit auch für die Frage nach Berechtigung, denn sollte der Nutzer diese Frage übersehen, hängt die Applikation in einer Warteschleife. Als letzte Anweisung wird mit `maximumAge:0` dem Browser mitgeteilt, dass dieser keine gespeicherten Positionen verwenden darf.

Im folgenden Abschnitt wird auf die Browserunterstützung im Detail eingegangen.

2.3.2 Geolocation-Unterstützung der Browser

Alle Browser haben die Methode `getCurrentPosition` bereits implementiert und können eine Position ermitteln. Dennoch konnten während der Tests mit der Evaluierungsseite Unterschiede und Probleme festgestellt werden. Hat der Browser nur IP-Adressen für die Lokalisierung zur Verfügung, kann Safari keine Position ermitteln und ruft die Misserfolgs-Methode auf. Die restlichen Browser können die Aufenthaltsstadt feststellen und setzen die Ortsmarkierung in deren Zentrum. Haben die Browser die Möglichkeit mit WiFi die Position zu ermitteln, können alle Browser die Koordinaten meistens nahezu genau bestimmen. Es kommt in wenigen Fällen vor, dass manche Browser trotz WiFi-Zugang die genaue Position nicht ermitteln können und stattdessen das Stadtzentrum anzeigen. Diese Unterschiede wurden am selben Ort mit derselben WLAN-Verbindung unabhängig vom Browser in Einzelfällen festgestellt.

Probleme gibt es auch mit gespeicherten Positionen. Obwohl eigentlich durch den Code festgelegt ist, dass die Position immer neu ermittelt werden muss, werden bei Chrome und Opera auf alte Daten zurückgegriffen. Google Chrome gibt an, die Standortdaten zu speichern, um sie einfacher abrufen zu können. Diese Daten werden regelmäßig aktualisiert[26]. Somit wird die Anweisung, keine gespeicherte Positionen zu verwenden, übergangen und unter genannten Umständen führt dies zu Ungenauigkeiten. Daher können Ortswechsel in kurzer Zeit nicht erkannt werden. Dieses Verhalten hat jedoch kaum Nachteile, da die Methode `getCurrentPosition` nur für die einmalige Erfassung der Koordinaten und somit meistens für Geräte mit festem Standort gedacht ist. Eine ständige Erfassung der Koordinaten soll mit `watchPosition` ermöglicht werden. Dies dient dazu, Bewegungen zu erfassen. Diese Methode wird jedoch noch von keinem der Browser implementiert. Lediglich das iPhone, so Peter Kröner[1], realisiert diese Methode.

Für die einmalige Erfassung der Position ist die Unterstützung von Geolocation geeignet, sofern ein WiFi-Signal zur Verfügung steht, trotz selten auftretender Ungenauigkeiten. Für Anforderungen mit hoher Genauigkeit und zeitlicher Auflösung, wie zum Beispiel für Navigationssysteme, sind die aktuellen Implementierungen der Browser noch nicht geeignet.

2.4 Neue Formen der Mediendarstellung

Bisher wird im Bereich Darstellung von Multimedia in Webseiten meistens auf alternative Lösungen zurückgegriffen. Um interaktive und dynamische Inhalte in einer HTML-Seite darstellen zu

können, wird mehrheitlich auf Flash zurückgegriffen. Für Flash und auch für die meisten anderen Alternativen, muss der Nutzer Plugins installieren, um die Seite richtig nutzen zu können. Plugins haben unterschiedliche Nachteile. Zum einen vermindern sie die Stabilität und Geschwindigkeit des Browsers. Es führt oft zu Inkompatibilitäten mit den Versionen des Browsers und Sicherheitsaspekte müssen beachtet werden, da Plugins meist ohne Wissen des Nutzers starten und unter Umständen Zugriff auf Systemressourcen haben. Durch die Einführung verschiedener Medienformen mit HTML5 ist mehrfach die Diskussion aufgekommen, ob Plugins, vor allem Flash, überflüssig werden könnten[8]. Das W3C ist bemüht, mit HTML5 alle Funktionalitäten zu implementieren, die für Webapplikationen notwendig sind. In den folgenden zwei Abschnitten wird aufgezeigt, wie man mit HTML5 Videos, Grafiken und Animationen anzeigen und erstellen kann.

2.4.1 Einbettung von Video

Adobe wirbt für den Flash Player mit einer reibungslosen Darstellung von hochwertigen Videos[14]. Bisher war das mit Standard-HTML nicht möglich. Mit HTML5 können nun Videos mit dem neuen Element `<video>` eingebunden werden.

```
1 <video width="200px" src="video.mp4" controls>
2   Der Browser unterstützt den neuen Video-Tag nicht
3 </video>
```

Abbildung 2.4: Beispielcode, mit dem ein Video in HTML eingebunden werden kann

Abbildung 2.4 zeigt in einem Beispiel die Anwendung des neuen `<video>`-Elements, in dem mit dem Attribut `width` die Breite des Videos auf `200px` gesetzt wird. Mit `controls` blendet man Steuerknöpfe ein, die browserabhängig angezeigt werden. Außerdem kann man Videos beim Laden automatisch starten (`autoplay`) und auch endlos laufen lassen (`loop`)[16]. Unterstützt der Browser das Videoelement nicht, übergeht dieser die Tags mit allen Attributen und stellt nur den Text zwischen dem öffnenden und schließenden Tag dar (Zeile 2). Somit kann dort ein Alternativcode platziert und wenn gewünscht, doch auf Plugins zurückgegriffen werden. Theoretisch bringt das `<video>`-Element alle nötigen Funktionen mit sich. Doch praktisch scheitert es noch an den Browserherstellern, die sich bisher nicht auf ein einheitliches Videoformat einigen konnten[9]. OpenSource Browser unterstützen nur den Theora Codec im `ogg`-Container und Internet Explorer oder Safari nur den H.264-Codec im `mp4`-Container. Der Unterschied zwischen den beiden Formaten liegt hauptsächlich im Kostenbereich. H.264 unterliegt Patenten und kostet dementsprechend für den Browser. Der Vorteil am Codec H.264 ist, dass er speziell für Geräte mit schwächeren CPUs entwickelt wurde. Für Handys und Smartphones ist dieses Format somit besonders gut geeignet[10]. Theora wird seit 2004 nicht weiter entwickelt, aber unterliegt keinen Lizenzen und ist daher kostenlos[2, Seite 84]. Solange keine Einigung im Format gefunden wird, muss man für eine durchgehende Browserunterstützung jedes einzubindende Video in beiden Formaten vorliegen haben. Abbildung 2.5 zeigt wie Alternativquellen mit dem neuen `<source>`-Element angegeben werden können. So sucht sich der Browser heraus, welches Video er anzeigen kann. Wie man Tabelle 2.3 entnehmen kann, unterstützt nur Google Chrome beide Formate. Safari hat in der aktuellen Version einen Fehler, denn offiziell kann Safari HTML5-Videos anzeigen. Allerdings funktioniert die Anzeige momentan weder bei eigenen Implementierungen noch bei anderen Test-

[8] Eine öffentliche Diskussion findet zum Beispiel zwischen Steve Jobs und Adobe statt, und geht darüber, warum Flash nicht mehr auf IPods, IPads und IPhones unterstützt wird. (Steve Jobs Stellungnahme zu den Beweggründen ist unter [15] zu finden)

[9] Stand August 2011

[10] Hier liegt der Vorteil am Dekodieren des Videos auf externen Chips und nicht auf der CPU. Auch Blu-ray verwendet diese Technik und kann so Videos in sehr guter Qualität abspielen

```
1 <video width="200px" controls>
2   <source src="video.mp4"/>
3   <source src="video.ogg"/>
4   Der Browser unterstützt den neuen Video-Tag nicht
5 </video>
```

Abbildung 2.5: Beispielcode für Alternativquellen mit dem source-Element

seiten, wie YouTube Html5-Videos[11]. Firefox und Opera unterstützen nur Videos mit dem Codec Theora und der Internet Explorer nur Videos mit dem Codec H.264.

Eingabetyp	Chrome	Firefox	IE	Opera	Safari
Theora	ja	ja	nein	ja	nein
H.264	ja	nein	ja	nein	nein

Tabelle 2.3: Übersicht über Browser und Videoformate

2.4.2 Erstellen von Grafiken und Animationen

Häufig wird bei der Darstellung und Interaktionen mit Grafiken auf Flash zurückgegriffen. Dies soll nun auch in HTML5 ermöglicht werden. Hierzu wird das `<canvas>`-Element eingeführt, das mit JavaScript-Funktionen das Anzeigen und Bearbeiten von 2D Grafiken erlaubt.

Ein `<canvas>`-Element ist eine Art Zeichenleinwand, auf der mit JavaScript-Funktionen gemalt werden kann. Es wird analog zum `<video>`-Element eingebunden. Der Browser führt ebenfalls den Alternativcode zwischen den Tags aus, falls dieser das Element nicht kennt. Mit JavaScript kann man auf das Element wie in Abbildung 2.6 dargestellt zugreifen. Mit dem Objekt

```
1 var canvas=document.getElementsByTagName("canvas")[0];
2 var context=canvas.getContext("2d");
```

Abbildung 2.6: JavaScript-Zugriff auf Canvas-2D-Kontext

`context` (Zeile 2) können nun verschiedene Formen gezeichnet werden. Ein ausgemaltes Rechteck wird zum Beispiel mit
`context.fillRect(linkeObereEckeX, linkeObereEckeY, breite, hoehe)` erstellt. Analog dazu kann man mit der Angabe von einzelnen Punkten Pfade generieren. Kurven können mit quadratischen Kurven, Bézierkurven oder mit Methoden wie `arcTo()` erstellt werden. Mit diesen Funktionen kann man einfache bis komplexe Zeichnungen realisieren. Farben kann man beliebig definieren, es gibt Methoden für lineare und runde Farbübergänge und Bilder können ebenfalls auf der Oberfläche dargestellt werden. Selbst Videos können auf der Zeichenfläche angezeigt und durch genannte Funktionen bearbeitet werden. Für die Evaluierungsseite wurde ein Bild mit einem Auto erstellt (siehe Abbildung 2.7). Der Bereich um das Rad wurde mit der Methode `arcTo` und die restlichen Rundungen der Karosserie mit `quadraticCurveTo` gezeichnet. Wie man an Abbildung 2.7 sehen kann, implementieren Internet Explorer und Opera die Funktion `arcTo` falsch und das Auto wird nicht korrekt dargestellt.

Mit Key- und Mouse-Events kann man auf Nutzereingaben reagieren und somit Zeichenprogramme und Spiele implementieren. Ein Beispiel für ein gelungenes Zeichenprogramm, das mit `<canvas>` realisiert wurde und die Stärke von HTML5 demonstriert ist “Harmony”[12]. Für die Eva-

[11] Auf http://www.youtube.com/html5 kann man einstellen, dass man wenn verfügbar, die YouTube-Videos mit HTML5 sehen möchte.

[12] http://mrdoob.com/projects/harmony

luierungsseite wurde ein einfaches Zeichenprogramm erstellt, das verdeutlicht, wie Nutzerinteraktionen mit dem `canvas`-Element möglich sind. Im Anhang befindet sich ein Screenshot der Evaluierungsseite mit entsprechendem Zeichenprogramm (Abbildung 5.3). Auch Animationen sind möglich. Einfache JavaScript-Methoden, die das `canvas` in Zeitintervallen immer neu gestalten, lassen Animationen entstehen. Auf der Evaluierungsseite wird das Auto alle 70 Millisekunden neu gezeichnet. Einfache Konturen, die Häuser repräsentieren, wurden an der x-Achse entlang verschoben. So kann mit wenigen Methodenaufrufen eine kleine Animation eines fahrenden Autos erstellt werden. Bis auf die unterschiedlichen Implementierungen der `arcTo`-Methode sind die Browserunterstützungen gut und `canvas` kann bereits eingesetzt werden.

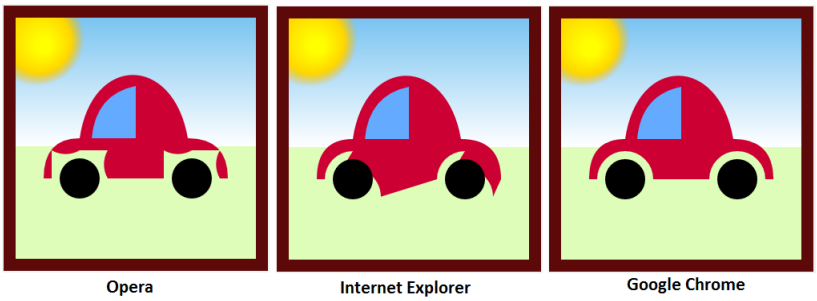


Abbildung 2.7: Darstellung der arcTo()-Methode im Vergleich

Das `<canvas>`-Element scheint dennoch etwas von den Funktionen eingeschränkt. Aktuell gibt es nur einen 2D-Kontext, ein 3D-Kontext soll jedoch noch entwickelt werden. Es gibt bereits einige JavaScript-3D-Engines, unter anderem WebGL, die eindrucksvolle Interaktionen mit 3D-Szenarien in Browsern ermöglichen. Bisher ist dennoch ein Ende von Flash nicht abzusehen. Im Bereich Video und Darstellung einfacher Grafiken wird HTML die Oberhand gewinnen, da hier die Vorteile überwiegen. Für spezielle Oberflächen und Benutzerschnittstellen ist Flash momentan noch besser geeignet. Vermutlich werden beide Technologien ihre jeweiligen Anwendungsbereiche finden, für die sie eingesetzt werden.

2.5 Lokale Speicherung mit Web Storage

Webapplikationen nutzen oft lokalen Speicher um spezifische Benutzerinformationen auf dem Client zu speichern, meist mit HTTP Cookies. Diese haben entscheidende Nachteile. Zum einen sind Webapplikationen mit HTTP Cookies langsamer, da bei jedem HTTP Request die Cookies übertragen, gelesen, eventuell bearbeitet und wieder empfangen werden, selbst wenn die Cookies in diesem Request nicht gebraucht werden. Ein weiterer Nachteil ist, dass pro Cookie nur vier Kilobyte Speicherplatz zur Verfügung gestellt werden[2, Seite 127]. Mit HTML5 Web Storage wird nun ein alternativer Standard geschaffen. Es werden Schlüssel-Werte-Paare auf dem Client gespeichert. Der große Vorteil Cookies gegenüber ist, dass eine ständige Übertragung an den Server entfällt. Die Daten werden dauerhaft gespeichert, können aber ähnlich wie die Cookies im Browser gelöscht werden. Server haben mit Web Storage keinen direkten Zugriff auf den Client. Sie haben lediglich die Möglichkeit `read`- oder `write`-Anfragen zu senden und können damit nur selbst erstellte Storagedaten lesen und bearbeiten. Dabei werden nur Strings gespeichert.

Die Größe des zur Verfügung stehenden, lokalen Speicherplatz, hängt vom Browser ab. Meist werden insgesamt fünf Megabyte, im Internet Explorer aber zehn Megabyte Speicherplatz zur Verfügung gestellt. Falls die Situation eintritt, dass der Speicherplatz voll ist, kann kein zusätzlicher Speicherplatz geschaffen werden. Erst wenn der Nutzer manuell im Browser Webdaten löscht, ist wieder Speicherplatz vorhanden[2, Seite 132].

Es gibt zwei Arten von Web Storage: sitzungsbezogene und dauerhafte Speicherung. Bei ersterem handelt es sich um `sessionStorage` und bei letzterem um `localStorage`. Die Daten von `sessionStorage` sind abhängig von einem geöffneten Fenster und sind auch nur während dessen

Lebensdauer verfügbar. Durch eine solche Speicherung kann man mit mehreren Fenstern dieselbe Seite öffnen, unterschiedliche Daten eingeben und speichern lassen. Mit Cookies wäre dies nicht richtig zu realisieren. Hier würde die Gefahr bestehen, dass sich die gespeicherten Inhalte der einen Seite mit den der anderen Seite vermischen würden. Der Speichertyp `localStorage` hat eine vergleichbare Funktion wie Cookies: eine dauerhafte, browserabhängige Speicherung der Daten.

Die Implementierung ist sehr einfach. Da `sessionStorage` und `localStorage` das gleiche Interface implementieren, ist ihre Benutzung identisch. Um einen Eintrag in einem Storage zu erstellen, ruft man lediglich `sessionStorage.key=wert` auf. Man kann auch die Methode `setItem(key, value)` aufrufen. Werte fragt man analog mit `sessionStorage.key` ab. Ist der geforderte Schlüssel nicht vorhanden wird `null` zurückgegeben. Der zu speichernde Wert wird immer als String abgelegt, unabhängig vom ursprünglichen Typ. Ruft man in JavaScript `localStorage.length` auf erhält man die Anzahl der gespeicherten Schlüssel-Werte-Paare. Es gibt auch ein Storage-Event, das gefeuert wird, sobald im Storage etwas gelöscht, hinzugefügt oder geändert wurde[13].

Alle Browser unterstützen diese Spezifikation bereits sehr gut und es können mit Web Storage lokal Daten abgelegt werden. Auf lange Zeit wird die Standardfunktionalität nicht ausreichend sein. Insbesondere die Key-Value-Paare und die begrenzte Speichergröße (5 MB) schränken den Einsatz dieser Lösung ein. Eine mögliche Alternative hierzu ist zum Beispiel IndexedDB.

2.6 Echtzeitübertragung von neuen Daten

Das Web wird immer schneller und interaktiver. Für bestimmte Anwendungen ist es essentiell, ihren Inhalt in Echtzeit zu synchronisieren. Bisher wird das meist mit Polling, Long Polling oder Streaming realisiert. Da aber Polling unnötig hohen Datentransfer mit sich zieht, und Long Polling und Streaming offene Verbindungen zu Servern erfordern, sind diese Lösungen nicht optimal. Im Folgenden werden Web Sockets vorgestellt, die eine Echt-Zeit-Aktualisierung und eine Lösung für genannte Probleme bieten.

2.6.1 Grundlegende Funktionalität von Web Sockets

Mit Web Sockets sind Kommunikationen zwischen Client und Server über einen bidirektionalen Kommunikationskanal möglich. Dazu wird ein neues auf TCP basierendes Web-Socket-Protokoll eingeführt. Ruft ein Client eine Seite auf, findet wie immer ein HTTP-Handshake statt und wenn ein Web Socket dabei erstellt werden soll, folgt ein Upgrade vom HTTP-Protokoll zum Web-Socket-Protokoll[3, Seite 2]. Die Verbindung ermöglicht dann einen Datenaustausch von sowohl Server- als auch Clientseite.

Welche Vorteile Web Sockets im Vergleich zu Polling besitzen, stellen Peter Lubbers und Frank Greco in einem Artikel über Web Sockets dar[23]. Sie vergleichen zwei Webapplikationen, die jede Sekunde eine Anfrage an den Server senden und errechnen die Größe der Daten der Header von HTTP-Anfrage und HTTP-Antwort. Das Ergebnis ist, dass für 10000 Clients, die jede Sekunde pollen, 66 Megabits pro Sekunde übertragen wurden (66Mbps) und bei 100000 Clients 665 Megabits pro Sekunde. Eine Webapplikation, die dieselben Daten über Web Sockets empfängt, verursacht einen viel kleineren Datentransfer: bei 10000 Clients 0,153 Kilobits und bei 100000 Clients 1,526 Kilobits pro Sekunde. Im Test wird auch die Verzögerungszeit untersucht und festgestellt, dass die von Web Sockets nur ein Drittel der Verzögerungszeit der Polling-Applikation verursacht[23].

Um für eine HTML-Seite eine Web-Socket-Verbindung zu implementieren, muss mit `var webSocket=new WebSocket(url)` ein Objekt des Typs Web Socket erstellt werden, wobei mit `url` die URL des Web Sockets mitgegeben werden muss. Für vier Events kann

[13] Das Event konnte jedoch nur der Internet Explorer auslösen

man dem Web Socket Aktionen zuweisen: `onopen`, `onmessage`, `onerror` und `onclose`. Mit `webSocket.onmessage=function(event){alert(event.data)}` würde der empfangene String ausgegeben, sofern eine Verbindung zu einem Web Socket steht.

2.6.2 Sicherheitsprobleme

Um die aufgezeigten Vorteile nutzen zu können, braucht man eine Web-Socket-Unterstützung sowohl im Browser, als auch im Server. Im Server ist eine solche durch eine Vielzahl an Möglichkeiten gegeben. Der Nutzer kann zwischen implementierten Web-Socket-Servern und Bibliotheken wählen, mit denen man sein eigenes Web-Socket-Servlet programmieren kann. Die Unterstützung auf der Browserseite bereitet momentan größere Probleme. Nachdem im Dezember 2010 sowohl Firefox als auch Opera die schon bestehende Implementierung von Web Sockets deaktivierten[24], unterstützen momentan nur die Browser Google Chrome und Safari Web Sockets standardmäßig. Die Deaktivierung ist auf Grund einer Untersuchung von Adam Barth et al. geschehen, der entdeckte, dass die Verbindung unter bestimmten Umständen angreifbar ist[3]. Das Problem liegt hauptsächlich an dem neuen Protokoll. Dieses definiert einen Handshake auf Basis der Upgradefunktion von HTTP. Das ist insofern problematisch, da viele Proxies den Upgrademechanismus nicht implementieren. Trotzdem kommt teilweise eine Verbindung zustande, die dann Sicherheitsrisiken darstellt, denn über sie konnten mehrere schädliche Aktionen ausgeführt werden[3]. Auch W3C selbst warnt in der Web-Socket-Spezifikation vor diesen Problemen:

> Following HTTP procedures here could introduce serious security problems in a Web browser context. For example, consider a host with a Web Socket server at one path and an open HTTP redirector at another. Suddenly, any script that can be given a particular Web Socket URL can be tricked into communicating to (and potentially sharing secrets with) any host on the Internet, even if the script checks that the URL has the right hostname.[21]

Dennoch kann man die Web-Socket-Unterstützung in Firefox und Opera manuell aktivieren. Somit hat lediglich der Internet Explorer noch keine Implementierungen für Web Sockets. Trotz dargestellter Vorteile gegenüber Polling, sollten Web Sockets nicht eingesetzt werden, solange die Sicherheit der Nutzer nicht gewährleistet ist.

2.7 HTML5 auf mobilen Endgeräten

Um HTML5 rundum bewerten zu können, muss auch die Einsetzbarkeit im mobilen Bereich getestet werden[14]. Dazu wurde erneut die Evaluierungsseite mit drei unterschiedlichen mobilen Browsern getestet[15]. Grundsätzlich können alle Browser auf dem entsprechenden, mobilen Gerät die Seite mit den neuen HTML5-Elementen sehr gut darstellen.

Die Unterstützung der semantischen Elemente, die in Tabelle 2.1 aufgeführt sind, in den mobilen Browsern, entspricht genau der Unterstützung der Standardbrowser. Nur das Element `<wbr>` konnte mobil nicht getestet werden, da dort die Fenstergröße nicht an die Bildschirmgröße angepasst wird. Im Bereich der Eingabevalidierungen verhalten sich die mobilen Browser ebenfalls ähnlich zu den zuvor getesteten anderen Browser. Opera Mobile unterstützt jede Eingabe und realisiert auch mobil Datums- und Farbeingaben mit praktischen Hilfsfenstern. Das iPad mit Safari Mobile passt teilweise die Tastatur an den Eingabetyp an. So wird beim Typ `email` das Zeichen `@` und beim Typ `url` das Zeichen `/` gleich erreichbar eingeblendet. Geolocation und Web Storage funktionieren analog zur Unterstützung der Standard-Browser.

[14] Ein Überblick zu den mobilen Unterstützungen ist auf Abbildung 5.2 im Anhang zu finden

[15] Dabei handelt es sich um folgende Browserversionen: Opera Mobile 11.00 für Android, Firefox 5.0 für Android, Safari 5.0.2 für IPad

Die Funktion Drag-and-Drop kann zur Zeit auf mobilen Browser nicht ausgeführt werden, da die Funktion belegt ist. Bei Smartphones mit Touchdisplay werden mit den Fingern getätigte Pfade zum Verschieben des Bildausschnitts verwendet. Aus diesem Grund kann auch kein mobiler Browser HTML5 Drag-and-Drop realisieren. Auch beim Zeichenprogramm mit `canvas` kann man zwar Punkte, jedoch keine Striche malen. Die Darstellung des `canvas`-Elements ist ansonsten gut mit lediglich kleinen Unterschieden. Die Rundungen in Firefox werden beispielsweise grob gepixelt dargestellt, während Opera auch im Mobilen die `arcTo`-Methode falsch implementiert. Safari und Opera haben dagegen Probleme mit der Animation beim Bewegen des Bildausschnitts, da währenddessen die Animation hängen bleibt.

Die Darstellung von Videos ist bei Safari nicht möglich, bei Safari Mobile dagegen schon. Hier können Videos mit dem Codec H.264 abgespielt werden. Nur `autoplay` und `loop` funktionieren nicht. Bei Opera ist die Videofunktion kritikwürdig. Das Attribut `autoplay` wird dort ebenfalls nicht unterstützt. So werden Videos ohne Steuerknöpfe nicht abgespielt. Das Attribut `loop` wird zwar erkannt, führt aber zu ernsthaften Problemen. Möchte man ein Video in Opera wiedergeben, wird das Video nach Drücken auf “play” im Vollbildmodus gewechselt. Will man zur Seite zurückkehren, ist das mit `loop` nicht möglich, da sofort wieder in den Vollbildmodus zurückgekehrt wird. Opera Mobile unterstützt im Gegensatz zu Opera nur H.264 und nicht Theora.

Web-Socket-Verbindungen können nur mit Opera Mobile und Safari Mobile hergestellt werden. In Firefox kann man die Unterstützung einschalten, Verbindungen kommen dennoch nicht zustande[16]. Allgemein ist die HTML5-Unterstützung auf mobilen Browsern genau so ausgereift wie die der Standard-Browser. Grundsätzlich gut, aber von Browser zu Browser sehr unterschiedlich.

2.8 Zusammenfassung und Fazit

Mit den neuen Formularfunktionen, Videoeinbettung, `<canvas>`, Geolocation, Web Sockets und Web Storage ist HTML eine viel mächtigere Sprache für Webapplikationen geworden. Damit wird sie den modernen Ansprüchen an Internetanwendungen gerecht. Auch im mobilen Bereich kann HTML5 bereits erfolgreich eingesetzt werden. Es wird deutlich, dass der Browser mit mehr Funktionen angereichert wird. Mit der Realisierung der neuen HTML5 Features geht es voran, jedoch ungleichmäßig. Die Abbildung 5.1 im Anhang zeigt eine Tabelle mit jeder beschriebenen neuen Technologie von HTML5 und die Unterstützungen der Browser. Semantische Elemente, Canvas und Web Storage werden sehr gut unterstützt und können bereits eingesetzt werden. Für Drag-and-Drop muss lediglich für Opera eine Alternativlösung gefunden werden. Die Einbettung der Videos hat, wie oben beschrieben, die Problematik der Formate. Wenn man aber jedes Video sowohl mit Codec H.264 als auch mit Theora vorliegen hat, kann ein Video in allen Browsern außer Safari dargestellt werden. Die neuen Formularfunktionen und die Unterstützung der Geolocation API sind noch nicht zufriedenstellend. Während die Browser bei der Geolocation zumindest die wichtigste Funktion realisieren, ist die Unterstützung der neuen Eingabetypen und Validierungen sehr unterschiedlich. Hier reicht die Unterstützung von gar nicht (Internet Explorer) über sehr beschränkt (Firefox) bis zu komplett (Chrome, Opera). Safari macht in einer anderen Richtung eine Ausnahme und validiert alle getesteten Eingabetypen, reagiert aber nicht auf falsche Eingaben. Wer die neuen Formulare verwenden möchte, muss somit nahezu für jeden Browser Alternativlösungen programmieren.

Eine Internetseite mit allen HTML5-Funktionen, die von allen Browsern optimal dargestellt wird, gibt es momentan noch nicht. Dennoch bietet HTML5 in seiner aktuellen Ausprägung sehr viele neue Möglichkeiten und wird in Zukunft sicherlich weiter Verbreitung finden.

[16] Ergebnis aus Test mit eigener Aufgaben-Management-Applikation und mit Test auf `http://html5demos.com/web-socket`

3 Entwicklung einer Aufgaben-Management-Applikation

In Unternehmen ist Aufgaben-Management in vielen Bereichen notwendig. Es gibt Aufgaben, die von unterschiedlichen Mitarbeitern bearbeitet werden müssen. Das erfordert Speicherung aller Aufgaben, ihre Zuordnung an bestimmte Angestellte und ein korrekter Ablauf der Bearbeitungen. In Produktionsbetrieben mit vielen Maschinen können zum Beispiel solche Aufgaben das Beheben von Störungen beinhalten, welche von Werkern gemeldet und von internen oder externen Servicetechnikern behoben werden müssen. Ein Servicetechniker ist meist nicht in der Nähe des Problems und bei einer hohen Anzahl an Störungsmeldungen ist es nicht möglich, alle sofort zu bearbeiten. Die Aufgaben werden folglich gesammelt, damit sie nacheinander abgearbeitet werden können.

Auch in Betrieben ohne Maschinen gibt es Bedarf für ein Aufgaben-Management. In einer Firma fallen viele administrative Aufgaben an, wie Urlaubsanträge oder Lohnverhandlungen.

Um genannte Aufgaben zu koordinieren, wird oft eine Business Process Management Engine (BPM-Engine) verwendet, die Prozesse und Aufgaben verwaltet. Nutzer können auf ihren Endgeräten über die Aufgaben-Management-Anwendung, die über das Internet mit einer BPM-Engine verbunden ist, Aufgaben erstellen. Je komplexer die Aufgaben sind, desto mehr Informationen werden gebraucht, um diese bearbeiten zu können. Bei Störmeldungen kann die Position des Problems nötig oder zumindest hilfreich sein. Außerdem können Bilder oder Videos dem Servicetechniker mehr Informationen bieten, als es mit Beschreibungen möglich wäre. Bisher können solche Informationen in HTML ohne Plugin zum Teil nicht dargestellt werden. Eine weitere Funktion, die eine solche Anwendung beinhalten sollte, ist ein auditives Nachrichtensignal beim Eingehen einer neuen Aufgabe, um den Servicetechniker von diesem Ereignis in Kenntnis zu setzen. Bisher ist das mit ActiveX oder anderen Zusatzprogrammen nur kompliziert zu realisieren.

Die wichtigste Anforderung an eine solche Anwendung ist die Aktualität der Aufgabenliste. Entweder muss der Betroffene fortlaufend die Anzeige seiner Aufgaben aktualisieren, vernachlässigt er dies, gerät der Arbeitsverlauf ins Stocken, oder die Anwendung fragt mit Polling selbstständig nach neuen Aufgaben und verursacht dadurch unnötig großen Datenverkehr.

Demnach wurde anhand dieser Anforderungen und Beobachtungen ein Prototyp zum Aufgaben-Management in HTML5 erstellt, der folgendes umsetzt:

- Möglichkeit zum Erstellen einer Störmeldung und eines Urlaubsantrags
- Anzeige der Aufgaben
- Selbstständige Aktualisierung der Aufgabenliste
- Abspielen eines Signals bei Eintreffen neuer Aufgaben
- Anzeige von Videos und Bilder
- Anzeige einer Karte der aktuellen Position

Durch die Entwicklung einer solchen Applikation mit HTML5 kann die Einsetzbarkeit von HTML5 in Unternehmen untersucht werden. Im Folgenden wird die Applikation beschrieben und dabei auf Implementierungsvorgänge eingegangen.

3.1 Zielsetzung

Der Fokus ist die Evaluierung der HTML5-Technologien im Unternehmenseinsatz anhand einer Aufgaben-Management-Applikation. In einer solchen können Aufgaben erstellt werden, die von bestimmten Nutzern oder Personengruppen beansprucht und anschließend bearbeitet werden können. Dazu können Benutzer ihre Aufgabenliste abrufen. Es wurde ein Prototyp erstellt, der für das Aufgaben-Management zwei Prozesse ermöglicht. Zum einen ist dies ein Urlaubsantragsprozess, der in jedem größerem Unternehmen vorhanden ist.

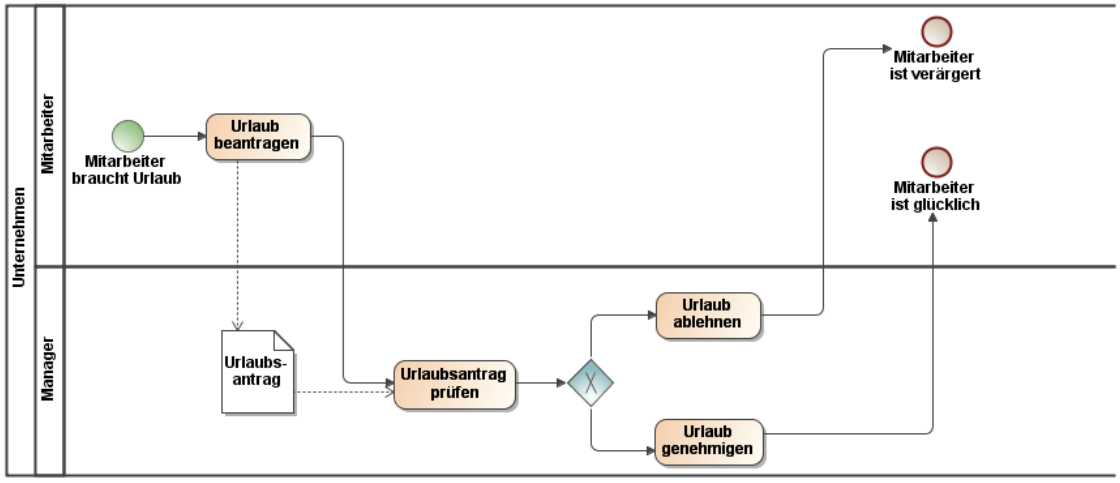


Abbildung 3.1: Veranschaulichung eines Urlaubsantragsprozesses

Abbildung 3.1 zeigt den Ablauf des Urlaubsantragsprozess. Ein Mitarbeiter einer Firma kann Urlaub beantragen. Es wird intern eine Aufgabe für die Manager erstellt. Diese können eine Liste aller noch nicht beantworteten Urlaubsanträge einsehen und sie einzeln genehmigen oder ablehnen.

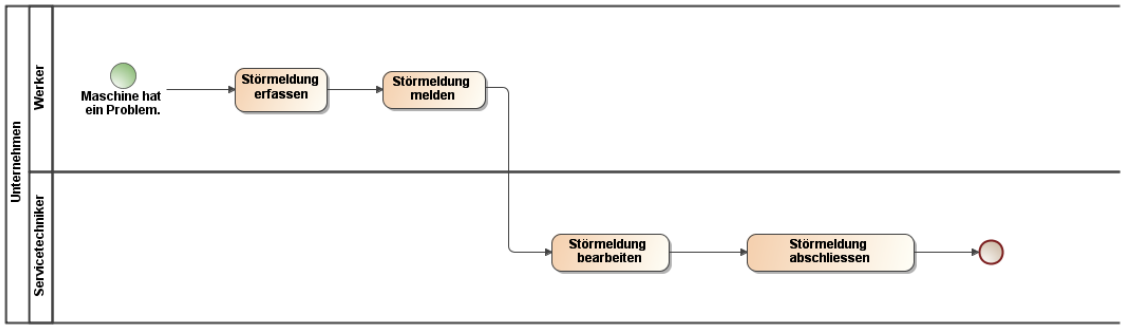


Abbildung 3.2: Veranschaulichung eines Störmeldungsprozesses

Der Prototyp ist auch für ein weiteres Szenario ausgelegt, das in Abbildung 3.3 dargestellt wird.

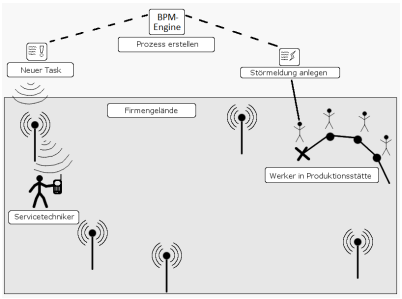


Abbildung 3.3: Ablauf einer Störmeldung

Es handelt sich dabei um eine große Produktionsfirma, die viele Maschinen auf großer Fläche verteilt, benötigt. Der graue Bereich der Abbildung stellt das Firmengelände dar. Auf diesem befinden sich mehrere Maschinen und Produktionsgeräte. Sie sind auf der rechten Seite des Bildes angedeutet. Werker arbeiten an den Maschinen. Erfolgt ein Fehler oder Problem, kann ein Werker eine Störmeldung anlegen. Der Weg dieser Meldung ist über dem Firmengelände dargestellt. Die BPM-Engine empfängt die erstellte Störmeldung und generiert einen Prozess. Dieser ist in Abbildung 3.2 dargestellt. Es wird eine neue Aufgabe für die Servicetechniker angelegt, der ein mobiles Gerät hat, mit dem er über WiFi Daten empfangen kann. Auf dem Gelände sind mehrere W-LAN Router verteilt, die einen durchgehenden Internetzugang auf dem Gelände ermöglichen. Der Servicetechniker kann sich seine Aufgaben und solche, die noch keiner

Person zugewiesen worden sind, anzeigen lassen und deren Details abrufen. Diese wurden vom Werker beim Erstellen einer Störmeldung, angegeben. Der Prototyp ermöglicht eine Erfassung der Position und eines Videos, das die Störung darstellt. Im Folgenden wird die Entwicklung des Prototyps und inwieweit HTML5 die genannten Anforderungen realisieren kann, beschrieben.

3.2 Vorgehensweise

Die Anwendung wurde mit dem Architekturmuster Model-View-Controller realisiert, sie ist demnach in drei Bereiche aufgeteilt. Die Komponente View ist für die Darstellung der Informationen verantwortlich. Sie wird in diesem Fall mit HTML5 Seiten realisiert, die JavaScript verwenden. Die Seiten kommunizieren über eine Web-Socket-Applikation mit dem Controller, der für die Programmsteuerung verantwortlich ist. Der Controller ist eine Java-Klasse, die die Methodenaufrufe der View an das Model, einer BPM-Engine, weiterleitet. Die Prozess-Engine wiederum sendet Änderungen an die Klasse, die die Web Sockets verwaltet, und diese teilt die Änderungen dem User mit. Das primäre Ziel dieser Arbeit ist, HTML5 auf realitätsnahe Anwendbarkeit zu untersuchen. Somit liegt der Fokus auf der View des Prototyps und der Web-Socket-Verbindung, und nicht auf

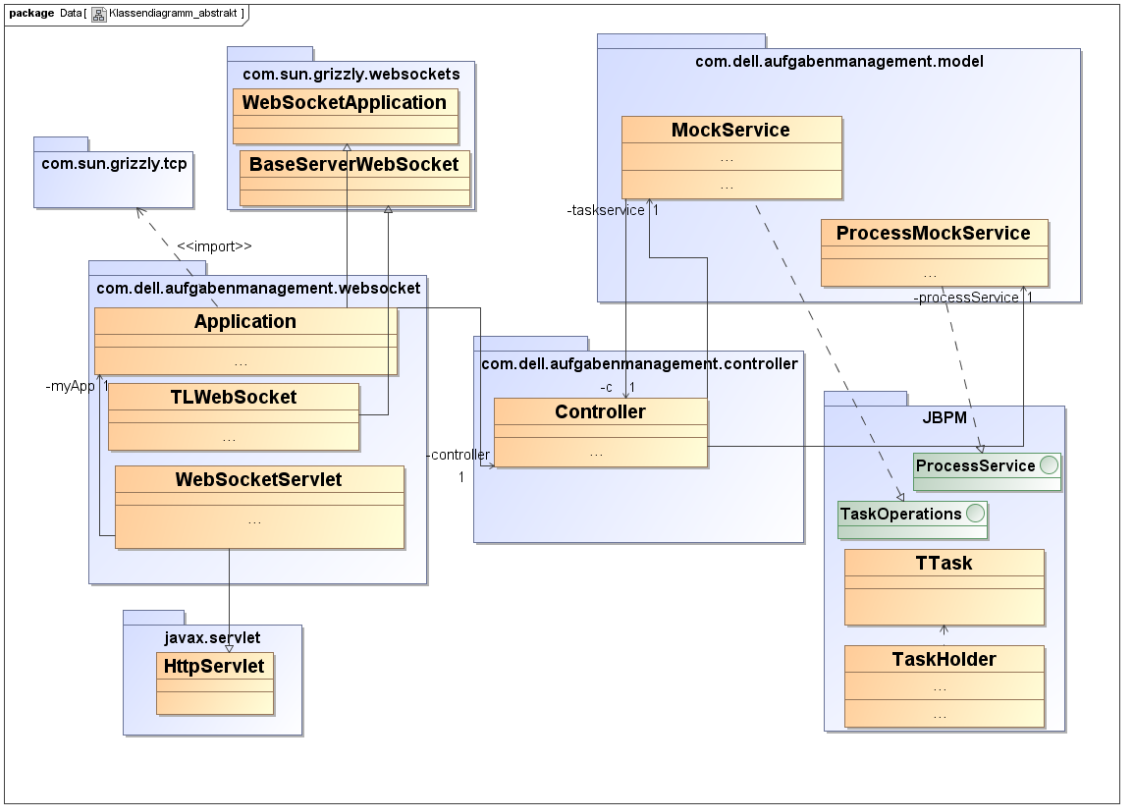


Abbildung 3.4: Klassendiagramm des Prototyps

der Anbindung einer existierenden Work-Flow-Engine. In diesem Prototyp wurde sie demnach über Mockklassen simuliert. Somit ist sichergestellt, dass die Methodenaufrufe des Controllers später bei Verwendung des BPM-Services nicht geändert werden müssen und bei Bedarf kann der Zugriff auf die BPM-Engine aktiviert werden. Auf die Verwaltung, Speicherung und Erstellung der Prozesse und Aufgaben wird in dieser Arbeit nicht näher eingegangen. Abbildung 3.4 zeigt das beschriebene Modell des Prototyps in einem Klassendiagramm.

3.3 Beschreibung der Applikation

Ein Überblick über die realisierten Webseiten findet sich auf Abbildung 5.4 im Anhang und sie werden im folgenden detailliert vorgestellt.

Um den Nutzer eindeutig zu identifizieren, findet zu Beginn ein Login auf der Seite `login.html` statt. Im produktiven Einsatz wäre die Eingabe eines Namens und eines Passworts erforderlich. Bei der Registrierung würde erfasst werden, zu welcher Personengruppe der Nutzer gehört. Im Prototyp wird dieser Vorgang vereinfacht. Abbildung 3.5 zeigt die Loginseite des Prototyps. Dort gibt es drei mögliche Nutzer, die sich mit der Personengruppe anmelden. Auf Passworteingabe wird verzichtet. Mögliche Personengruppen sind im Prototyp auf "Werker", "Servicetechniker" und "Manager" begrenzt. Jede Personengruppe kann unterschiedliche Aktionen vornehmen.

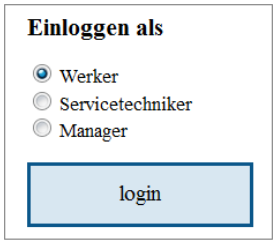


Abbildung 3.5: Login-Interface

Zum Starten einer Störmeldung steht dem Werker ein Formular `stoerProzess.hmtl` zur Verfügung, das in Abbildung 3.7(b) dargestellt wird. Darin erfasst er einen Kurztext und eine detaillierte Beschreibung des aufgetretenden Störfalls. Zudem bietet die Seite die Möglichkeit, den Sachverhalt anhand eines Videos oder Bildes durch Angabe einer URL zu dokumentieren und mit Hilfe der Geolocation die genaue Position des Problems anzugeben. Mit "Störmeldung anlegen" werden die Daten an das Web Socket gesendet und damit der Störprozess gestartet.

Der Servicetechniker kann die Aufgaben auf der Seite `showTasks.html` einsehen. Abbildung 3.6 zeigt das Interface der Task-Ansicht. Für jede Aufgabe werden bestimmte Attribute angezeigt: Die ID als eindeutige Identifizierungsnummer, eine kurze Beschreibung, das Erstellungsdatum und das Datum, an dem die Aufgabe erledigt sein soll. Zusätzlich kann man den Zustand und die Aufgabenart sehen und wer für die Aufgabe verantwortlich ist, beziehungsweise welche Personengruppe sie bearbeiten kann. Jeder Nutzer kann nur Aufgaben einsehen, die ihm zugewiesen sind. In diesem Fall können Servicetechniker nur Störmeldungen und Manager nur Urlaubsanträge bearbeiten und bekommen nur entsprechende Aufgaben angezeigt.

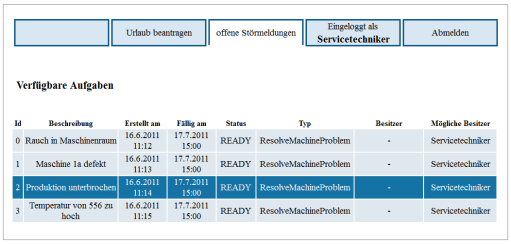
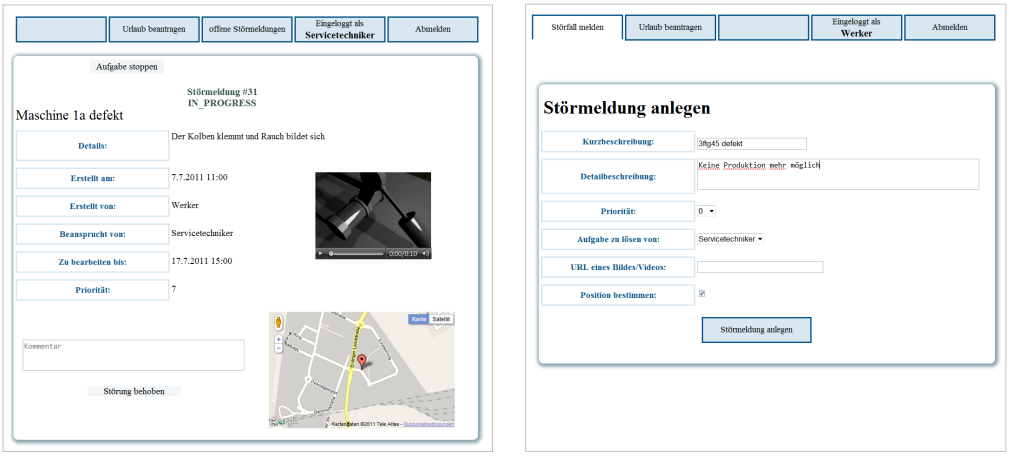


Abbildung 3.6: Taskansicht

Zur Prüfung einer Aufgabe wählt der Servicetechniker eine aus und gelangt dadurch auf eine weitere Seite (`taskinDetail.html`). Darin werden dem Benutzer eine vollständige Liste aller Details zu dieser Aufgabe angezeigt. Abbildung 3.7 (a) zeigt die Detailansicht einer Störmeldung. In der Überschrift kann man die ID, den Status und den Aufgabentyp sehen. Neben dem Kurztext, der schon in der Aufgabenliste ersichtlich ist, wird auch eine detaillierte Beschreibung der Aufgabe angezeigt. Hinzukommend zu den Informationen der Task-Ansicht, kann man auch Priorität und eine Karte mit der Position des Problems angezeigt bekommen. Außerdem wird ein Video oder Bild dargestellt. Der Servicetechniker übernimmt die Aufgabe (Aufgabe beanspruchen) und startet den Prozess (Aufgabe starten). Anhand der dargestellten Informationen analysiert er nun den Störfall und behebt das Problem an der Maschine. Die eingeleiteten Maßnahmen protokolliert er in einem Textfeld. Durch "Störung beheben" wird der Text gespeichert und eine Nachricht an die BPM-Engine gesendet. Die Aufgabe ist damit abgeschlossen (Status `completed`).

Werker und Servicetechniker können auf `urlaubsProzess.html` einen Urlaub beantragen. Es werden hierbei Start- und Enddatum erfasst und die Aufgabe dem Manager zugewiesen. Der Manager kann analog zum Servicetechniker die Aufgaben einsehen. Bei der Bearbeitung kann er nun den Urlaubsantrag genehmigen oder ablehnen.



(a) Detailansicht (b) Erstellungsformular

Abbildung 3.7: Detailansicht und Erstellungsformular einer Störmeldung

3.4 Funktionalitäten und ihre Realisierung

Der Prototyp verfügt über die wichtigsten Funktionen, die eine Aufgaben-Management-Applikation bereitstellen sollte und zusätzlich über einige praktische Erweiterungen. Dabei wurde alles mit HTML5 und JavaScript entwickelt, ohne auf externe Plugins zurückzugreifen. Im Laufe der nächsten Kapitel werden die Features beschrieben und auf die Realisierung eingegangen.

3.4.1 Anlegen von Aufgaben

Beim Anlegen einer Störmeldung werden alle Details erfasst. Für eine optimale Präsentation der Informationen zu dem Störfall kann die URL eines Bildes oder Videos angegeben werden. Hierfür wird der Eingabetyp `url` verwendet. Die automatische Validierung überprüft, ob die URL mit `http` beginnt. Eine weitere zusätzliche Information, die einem solchen Szenario nutzt, ist die genaue Position der Störung. Diese könnte der Werker zwar manuell eingeben, durch die neue Geolocation API kann dies nun auch automatisch passieren und erlangte Koordinaten können genutzt werden, um die Position in einer Karte anzuzeigen. Erstellt ein Werker eine Störmeldung, kann er angeben, ob die Position mit übermittelt werden soll. Ist dies der Fall, wird versucht, seine Koordinaten zu ermitteln und bei Erfolg an die BPM-Engine weitergeleitet. Wie in Punkt 2.3.1 dargelegt gibt es kein Timeout für den Fall, dass der Nutzer nicht auf die Erlaubnisfrage reagiert. Dies ist hier ein Problem, denn in einem solchen Fall würde die Anwendung in einer Endlosschleife hängen. Da die Applikation aber nur für ausgewählte Personen ist, kann davon ausgegangen werden, dass diese Nutzer der Applikation standardmäßig erlauben, die Position zu bestimmen. Das Eingabefeld für den Titel der Störmeldung wird mit dem Attribut `required` versehen und somit kann ohne eine Eingabe die Störmeldung nicht angelegt werden.

Die Erstellung eines Urlaubsantrags läuft analog zum Anlegen einer Störmeldung, es werden jedoch keine Videos oder Positionen benötigt, dagegen aber Zeitangaben des Urlaubsbeginns und Urlaubsendes. Dafür kann man den Input-Typ `date` nutzen. Im Falle von Opera würde dabei für die Auswahl des Datums ein praktischer Kalender angezeigt (siehe Kapitel 2.2.1). Da die HTML5-Validierung bei vollständiger Unterstützung die Datumsangaben überprüft, muss in diesem Fall nur eine zusätzliche Validierung erstellt werden. Es wird geprüft, ob das Enddatum nicht vor dem Anfangsdatum liegt. Ist die Eingabe invalide, kann diese mit der Methode `setCustomValidity(String fehlermeldung)` als fehlerhaft gekennzeichnet werden. Das Absenden wird dann mit einer browsertypischen Fehlermeldung mit dem vom Nutzer definierten Text

blockiert. Dadurch muss sich der Webentwickler nicht mehr um darstellerische Einzelheiten kümmern.

3.4.2 Aktualisierung der Informationen durch Web Sockets

Es ist äußerst wichtig, dass die Aufgabenliste ohne direktes Handeln des Nutzers aktuell ist. Damit der unnötige Datentransfer des Pollings nicht entsteht, werden HTML5 Web Sockets verwendet. Für den Prototyp ist die Web-Socket-Verbindung mit Klassen der Grizzly-Bibliothek[17] des Servers Glassfish 3.1[18] implementiert.

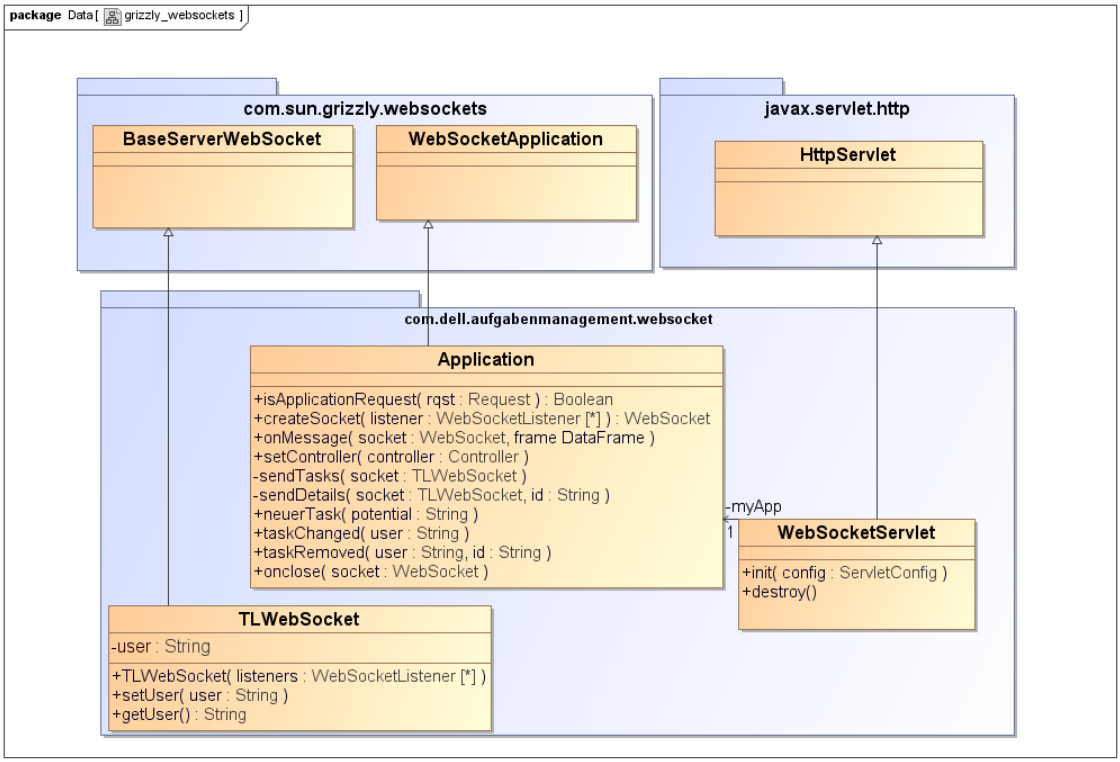


Abbildung 3.8: Nötige Klassen für einen Web-Socket-Server

Für die Socketverbindung wird ein Web-Socket-Servlet verwendet, das von der Klasse `HttpServlet` erbt. Dieses Java Servlet wird standardmäßig mit einem Präfix erstellt. Wird das Servlet initiiert, registriert es eine Web-Socket-Applikation bei einer Web-Socket-Engine. Diese Applikation (wird im Folgenden `Application` genannt) hält später alle offenen Web Sockets. Außerdem wird der `Application` eine Instanz der Klasse `Controller` mitgegeben, damit sie später die Informationen zu den Tasks einfordern kann. Wird von einer HTML5-Seite ein Web Socket erstellt, initialisiert die `Application` ein `TLWebSocket`, das von `WebSocket` erbt. Außerdem hält es den Namen des Nutzers, von dem dieses Web Socket erstellt wurde. Die `Application` reagiert auf Nachrichten, die von registrierten Web Sockets kommen. Sendet ein Web Socket eine Nachricht, wird in der `Application` die Methode `onMessage(WebSocket socket, DataFrame frame)` aufgerufen, mit dem sendenden Socket und der gesendeten Nachricht. Entsprechend des Inhalts der Nachricht kann die `Application` reagieren und Aktionen ausführen. Werden beispielsweise Tasks angefordert, ruft die `Application` Methoden im `Controller` auf, um entsprechende Aufgaben zu bekommen. Abbildung 3.8 zeigt den eben erklärten Zusammenhang zwischen den

[17] Der Server wurde nach einem Beispiel von Grizzly programmiert. Siehe CD
[18] Verfügbar unter `http://glassfish.java.net/downloads/3.1-final.html`

Klassen. Um Nachrichten an ein Web Socket zu senden, ruft die `Application` lediglich die Methode `send(String nachricht)` des `WebSockets` auf. Alle bei ihr registrierten Sockets hält die `Application` in einer Liste.

Auf Seite des Clients ist die Implementierung einfach. Jede HTML-Seite, die Daten empfangen oder senden möchte, erstellt als erstes ein Web Socket, über das sie Strings an die Klasse `Application` senden kann. Sendet die `Application` eine Nachricht zurück, wird in der HTML-Seite das JavaScript-Event `onmessage` mit dem gesendeten String aufgerufen. Durch Senden bestimmter Befehle, wie `"startProcess"` oder `"getTasks"` an die `Application` werden indirekt Methoden aufgerufen. Die `Application` untersucht den empfangenen String auf Inhalt und reagiert entsprechend. Parameter werden ebenfalls in den String eingebaut. In diesem Fall werden Parameter durch "#" getrennt an den Methodennamen angehängt. Da die `Application` immer weiß, von welchem Web Socket die Nachricht kommt, kann sie diese Information verwenden, um eventuell angeforderte Daten gleich zurückzusenden.

In jeder HTML-Seite des Prototyps (außer `login.html`) wird beim Laden der Seite ein Web-Socket mit der entsprechenden Servlet-URL erstellt. Sobald das Event `onopen` gefeuert wird, steht die Verbindung und Nachrichten in Form von Strings können gesendet werden. Die HTML-Seite sendet unmittelbar danach seinen Nutzernamen an die `Application`. Diese reagiert wie in Abbildung 3.9 dargestellt darauf.

```
1 //data ist der empfangene String
2 else if(data.contains("authenticate#")){
3   String user=data.substring(data.indexOf("#")+1);
4   //socket ist das Socket von dem data gesendet wurde
5 socket.setUser(user);
```

Abbildung 3.9: Codeausschnitt des Authentifizierungvorgangs des Nutzers

Die `Application` empfängt von einem ihr bekannten Web Socket das Schlüsselwort "authenticate" mit der Nutzer-Id (Zeile 2), extrahiert die Nutzer-Id aus dem String (Zeile 3) und speichert sie anschließend in das Web Socket, von dem die Nachricht kam (Zeile 5). Somit steht die Verbindung und das Web Socket ist bei der `Application` mit Name registriert. Ab jetzt können Aufgaben angefordert, erstellt oder bearbeitet werden.

Abbildung 3.10 zeigt zur Verdeutlichung einen schematischen Ablauf der Erstellung einer Aufgabe und die darauf folgenden Aktionen. Erstellt ein Werker über die Seite `stoerProzess.html` eine Störmeldung, wird über das Web Socket ein String mit "startProcess#..." an die `Application` gesendet (1,2). Diese ruft im `Controller` die Methode `startProcess(...)` mit allen erhaltenen Parametern auf (3). Danach sucht sie für alle Nutzer, für die diese Aufgabenerstellung relevant ist, ihre zugehörigen Web Sockets und sendet an sie den String "newTaskGenerated" (7,8). Empfängt die Seite `showTasks.html` diesen String (9), fordert sie alle Aufgaben erneut an, um eine aktuelle Aufgabenliste darstellen zu können. Dafür wird der String "getTasks" an die `Application` gesendet (10,11). Diese fordert entsprechende Aufgaben beim Controller an (12). Nachdem die `Application` vom `Controller` alle Aufgaben empfangen hat (13), filtert sie diese so, dass nur die für den entsprechenden Nutzer, gemäß Nutzername, übrig bleiben (14)[19] und sendet diese an das Web Socket zurück (15,16).

Mit Web Sockets kann man nur Bytes und Strings, aber keine Liste von Objekten versenden. Um nicht jedes Detail jeder Aufgabe einzeln oder als langen konkatenierten String zu versenden, wird im Prototyp die Bibliothek JSON[20] verwendet. Mit dieser Bibliothek kann ein Objekt mit Key-Werte-Paaren erstellt und dieses in einen String umgewandelt werden. Die HMTL-Seiten können den empfangenen String zurück in ein JSON-Objekt wandeln und dann einfach die Werte

[19] Dieses Vorgehen ist Teil der Alternativlösung. Normalerweise würde die BPM-Engine die Tasks filtern

[20] Verfügbar unter `http://www.json.org/`

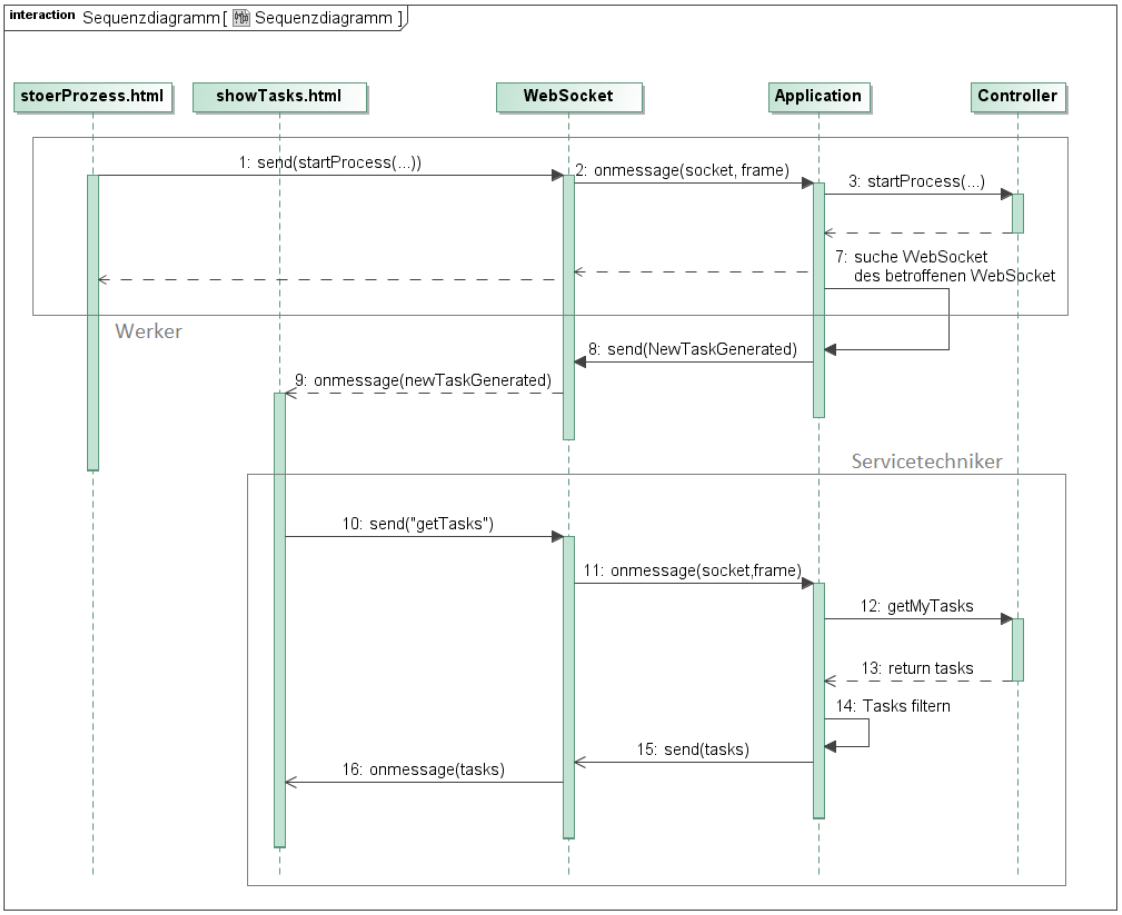


Abbildung 3.10: Schematischer Ablauf einer Taskanforderung

für definierte Keys abfragen. Die `Application` packt somit jeden einzelnen Task in ein JSON-Objekt und schickt dieses als String an das Web Socket, dass die Tasks angefordert hat. Dort wird der empfangene String zurück in ein JSON-Objekt gewandelt und es kann einfach auf die Attribute zugegriffen werden.

Es wäre auch denkbar, alle Tasks einem einzigen JSON-Objekt hinzuzufügen und nur dieses an das Socket zu senden. Diese Variante wurde bewusst nicht gewählt, da im Falle von sehr vielen Tasks der String zu lang werden könnte.

3.4.3 Signal bei neuem Task

Bekommt eine Seite die Nachricht, dass ein neuer Task erstellt wurde, wird dieses Ereignis mit einem Sound untermalt, damit die Nachricht auch bemerkt wird, wenn der Nutzer gerade nicht auf den Bildschirm sieht. Mit HTML 4.01 allein wäre das nicht möglich gewesen. Man hätte externe Plugins wie Flash, Windows Media Player oder ActiceX einbinden müssen. HTML5 ermöglicht ein einfaches und benutzerfreundliches Abspielen einer Audiodatei. Bindet man das `<audio>`-Tag ohne Steuerelemente ein, ist es unsichtbar. Mit DOM kann man auf das `audio`-Element einfach zugreifen und mit der Methode `play()` die Datei abspielen. Abbildung 3.11 zeigt wie im Prototyp eine noch einfachere Variante gewählt wird, die für genau solche Zwecke bereitgestellt wurde. Wenn die HTML-Seite die Nachricht `"newTaskGenerated"` vom `webSocket` empfängt (Zeile 2-3), dann wird vorrübergehend ein Audioobjekt erstellt und gleich abgespielt (Zeile 4). Anschließend werden die neuen Tasks angefordert (Zeile 5). Klickt man dann auf den neuen Task, wird der

```
1 //Es wurde eine Nachricht empfangen
2 webSocket.onmessage=function(event){
3   if(event.data=="newTaskGenerated"){
4     new Audio('audio/signal.wav').play();
5     neueTasksAnfordern();
6 }
```

Abbildung 3.11: Codeausschnitt zum Abspielen einer Audiodatei

Nutzer auf die Seite `taskInDetail.html` weitergeleitet und die Details angezeigt.

3.4.4 Detailangabe zu Aufgaben

Neben Standardinformationen wie Kurzbeschreibung, Detailbeschreibung und Priorität kann eine Störmeldung auch weitere Details enthalten. Eine Störung einer Maschine kann durch ein Bild oder sogar durch ein Video besser beschrieben werden. Je besser der Servicetechniker über das Problem informiert ist, desto besser kann er sich auf das Problem einstellen und abschätzen, wie weiter verfahren werden muss. Da Videos viel Speicherbedarf erfordern, kann man das Video selbst nicht versenden. Deswegen wurde im Prototyp die Möglichkeit eingerichtet, eine URL zu einem Video anzugeben. So könnte der Werker ein Video des Problems aufnehmen, auf eine firmeninterne Plattform hochladen und den entsprechenden Link in die Störmeldung einfügen. Der Werker kann analog dazu auch eine URL zu einem Bild angeben. In der Detailansicht wird überprüft, ob die Endung der URL die eines Bildes oder Videos ist und erstellt daraufhin ein entsprechendes Element. Das Video wird dann mit dem HTML5-Element `<video>` eingebunden und für den Servicetechniker in der Detailansicht der Tasks dargestellt.

Sofern Längen- und Breitenangaben vorhanden sind, kann eine Position in einer Karte dargestellt werden. Mit Hilfe der JavaScript-Bibliothek von Google[21] kann eine Karte einfach und kostenlos eingebunden werden. Die beiden Koordinaten definieren dabei eine Position auf der Karte, welche durch einen Marker kenntlich gemacht wird. Die Karte wird zentriert und mit einer ausreichenden Auflösung (Zoom) dargestellt. Dies ist ein gutes Zusammenspiel zwischen der Geolocation API und Google Maps und ideal für Anwendungen wie die hier entwickelte Applikation.

3.5 Browserunterstützungen des Prototyps

Theoretisch bringt HTML5 viele, für Unternehmen nützliche Neuheiten mit. Dennoch ist die Unterstützung von HMTL5 noch nicht vollständig. In diesem Punkt wird beschrieben, wie die Browser die eingesetzten Technologien und damit den Prototyp realisieren können. Abbildung 5.3 im Anhang stellt als Übersicht die Unterstützungen dar. Der Internet Explorer kann noch keine WebSocket-Verbindung aufbauen und wurde somit in dieser Untersuchung nicht beachtet. Grundsätzlich können alle anderen Browser die Applikation zufriedenstellend bis sehr gut darstellen und ausführen.

Safari stellt bisher die schlechteste Unterstützung dar und erfordert einige Alternativimplementierungen. Der Browser verhält sich bei unbekannten Elementen unüblich. Im Prototypen wird mit dem Element `progress` ein Ladebalken dargestellt und je nach Bedarf darauf zugegriffen und ein- und ausgeblendet. Browser, die dieses Element nicht kennen, sollten den Inhalt zwischen den Tags als Text darstellen[22]. Safari dagegen ignoriert das Element komplett und bleibt dementsprechend hängen, wenn man später auf das Element `progress` zugreift. Somit musste für die Erstellung dieses Elements eine Browserweiche eingebaut werden. Das Verhalten im Bezug auf das

[21] Im Prototyp wurde `http://maps.google.com/maps/api/js?sensor=false&language=de` eingebunden
[22] Firefox stellt entsprechend das Alternativwort “lädt” anstatt des Balkens dar

Element `audio` war vergleichbar und auch hier musste eine Weiche eingefügt werden, da Safari das Audioelement noch nicht unterstützt. Die Web-Socket-Verbindung dagegen ist gut. Ein großes Problem, das nicht gelöst werden konnte, ist, dass Safari nach dem Abmelden die zuletzt erstellte Aufgabe ein weiteres Mal generiert und damit diese Aufgabe doppelt vorhanden ist. Es kommt außerdem vor, dass Safari beim Abmelden die Aufgabe nicht ein weiteres, sondern bis über 10000 weitere Male erstellt. Vor allem bei Safari Mobile auf dem IPad konnte dieses Verhalten registriert werden. Scheinbar nur, wenn man eine URL mit der Endung `mp4` für das Video angibt. Für dieses fatale Fehlverhalten konnte keine Erklärung gefunden und das Problem nicht behoben werden.

Google Chrome hatte dieses Problem ebenfalls. Seit dem Update auf Version 13 wird beim Abmelden keine Aufgabe mehr doppelt erstellt und seit dem ist das zuvor genannte Problem in Chrome nicht mehr aufgetreten. Mit Version 12 kam bis zu 50% keine Verbindung zustande. Durch das Update hat sich die Web-Socket-Verbindung ebenfalls stabilisiert. Man kann bei Google Chrome in Version 13 lediglich kritisieren, dass die selbstdefinierten Fehlermeldungen nicht angezeigt werden.

Firefox zeigt diese Fehlermeldungen zumindest als Tooltip an, kennt aber den Typ `date` nicht. Mit Firefox muss der Nutzer demnach das Datum per Hand eingeben. Aus diesem Grund musste zusätzlich eine Validierung der Eingabe hinzugefügt werden, damit man im Firefox sichergehen kann, dass das richtige Datumsformat eingegeben wird. Die Web-Socket-Unterstützung muss manuell eingestellt werden, dann kann über Firefox eine stabile Web-Socket-Verbindung aufgebaut werden.

Für Opera müssen die Web Sockets ebenfalls manuell aktiviert werden. Ansonsten ist Opera der Musterbrowser dieser Anwendung. Mit dem praktischen Kalender als Eingabehilfe für die Datumsangaben sind die Formulare optimal und die selbstdefinierten Fehlermeldungen geben dem Nutzer Rückmeldungen über falsche Eingaben.

Für einen Einsatz in Unternehmen ist eine fehlerlose und nutzerfreundliche Anwendung notwendig. Von den fünf Browsern erfüllt nur Opera diese Anforderungen. Es wird deutlich, dass aktuell keine umfangreiche HTML5-Anwendung geschrieben werden kann, die von allen Browsern gleichermaßen dargestellt wird. Hat ein Unternehmen diesen Anspruch, ist HTML5 noch nicht die richtige Sprache. Für das Szenario mit den Störmeldungen wäre es genügend, wenn ein Browser die Anwendung unterstützen kann und die Mitarbeiter diesen auf ihren Arbeitsgeräten nutzen. Insgesamt konnte erfolgreich eine Kommunikation zwischen mobilem Gerät, Server und Laptop mit allen Browsern, die Web Sockets unterstützen, hergestellt werden.

3.6 Die Aufgaben-Management-Applikation offline

Das Szenario, für das der Prototyp erstellt wurde, schließt nicht aus, dass der Nutzer unter Umständen keine Internetverbindung hat. Für die Applikation wäre es von großem Nachteil, zu solchen Zeitpunkten nicht anwendbar zu sein. Im Rahmen von HTML5 ist eine Offline API entwickelt worden, mit der man Dateien auf dem Client speichern lassen kann, damit die Anwendung auch ohne Internetverbindung nutzbar ist. In einer Aufgaben-Management-Applikation sollten Aufgaben auch offline erstellt werden können. Diese werden dem Server, sobald eine Verbindung wieder hergestellt wurde, übermittelt. Die Aktualisierung der Aufgaben dagegen kann nur online geschehen, da die Informationen nur über den Server ermittelt werden können.

Damit Aufgaben auch offline erstellt werden können, werden bei Aufruf der Login-Seite alle dafür relevanten Seiten lokal beim Nutzer gespeichert. Dies geschieht mit dem Application Cache von HTML5. Mit dem Attribut `manifest="cache.manifest"` für das Element `<html>` wird auf die angelegte Manifest-Datei verwiesen. Diese beschreibt, welche Seiten für die Offlineverwendung lokal gespeichert werden sollen. Die Manifest-Datei für den Prototypen wird in Abbildung 3.12 dargestellt. Mit `NETWORK` (Zeile 3) wird eine zwingende Internetverbindung für die darauffolgenden Dateien angegeben. `CACHE` (Zeile 7) signalisiert, dass alle darunterstehenden Dateien für einen Offline-Gebrauch lokal gespeichert werden müssen. Mit `FALLBACK` (Zeile 15)

wird eine Datei angegeben, die angezeigt werden soll, wenn die aufgerufene Seite aufgrund einer fehlenden Internetverbindung nicht verfügbar ist (in diesem Fall z.B. wenn `showTasks.html` oder `taskInDetail.html` aufgerufen werden). Werden Dateien einmal im Application Cache abgelegt, werden diese immer aufgerufen, unabhängig von einer eventuell bestehenden Internetverbindung. Damit der Client Änderungen von im Cache liegenden Dateien realisiert, muss eine regelmäßige Überprüfung der Manifest-Datei implementiert werden. Sobald sich die Manifest-Datei verändert hat, werden die Seiten neu gespeichert. Im Prototyp wird der Cache aktualisiert, wenn das Event `updateready` vom Application Cache ausgelöst wird. Diese Funktionalitäten können bis auf den Internet Explorer alle Browser umsetzen[34] (Stand August 2011). Der Online-Status kann mit `navigator.onLine()` abgefragt werden. Google Chrome gibt mit diesem Methodenaufrufen den korrekten Internetstatus zurück, während Firefox und Opera nur `false` für offline zurückgeben, wenn der Nutzer manuell den Offline-Modus wählt. Wird die Internetverbindung gekappt, liefern Firefox und Opera noch immer `true`.

Erstellt der Nutzer offline eine Aufgabe, wird diese mit Web Storage lokal auf dem Client gespeichert. Bei jedem Reload werden gespeicherte Aufgaben an den Server gesendet, sofern der Nutzer online ist und anschließend aus dem Speicher gelöscht. Um den Zwischenschritt mit dem Reload zu vermeiden eignet sich das `ononline`-Event, das ausgelöst wird, sobald der User online geht. Dieses Event wird jedoch noch von keinem Browser realisiert.

cache.manifest

```
1  CACHE MANIFEST
2
3  NETWORK:
4  showTasks.html
5  taskInDetail.html
6
7  CACHE:
8  stoerProzess.html
9  urlaubsProzess.html
10 javascripts/main.js
11 javascripts/createProcess.js
12 stylesheets/menu.css
13 stylesheets/formular.css
14
15 FALLBACK:
16 / offline.html
```

Abbildung 3.12: Manifest-Datei cache.manifest für den Prototypen

Implementiert man all diese Offline-Funktionalitäten für den Prototyp, kann Opera ihn nicht mehr ausführen. Firefox und Chrome dagegen schon, sind dabei aber zum Teil fehlerhaft oder instabil. Aus diesem Grund wurde die Offline-Funktionalität nicht in die finale Version des Prototypen eingebaut[23]. Dennoch wurde durch die Versuche mit der Offline API aufgezeigt, dass Rich Internet Anwendungen mit besserer Browserunterstützung zukünftig auch offline angewendet werden können.

3.7 HTML5 in Unternehmen

Für einen realistischen Unternehmenseinsatz müsste die Anwendung noch mit einer BPM-Engine verknüpft und alle fehlenden Funktionalitäten implementiert werden. JavaServer Faces (JSF) könnten die Applikation dynamischer gestalten. Hier gibt es bereits HTML5-Komponenten und die Version 2.2 wird offiziell HTML5 unterstützen[28]. Der Prototyp erfüllt dennoch genug Funktionen, um eine Unternehmensapplikation zu repräsentieren. Durch ihn kann die Einsatzfähigkeit von HTML5-Applikationen in Unternehmen eingeschätzt werden. Wie anfangs in dieser Arbeit aufgeführt, befinden sich Unternehmen im Wandel und benutzen zunehmend mobile Endgeräte. Bei diesen tritt das Problem auf, dass verschiedene mobile Betriebssysteme ähnlich hohe Marktanteile aufweisen. Abbildung 3.13 zeigt die Verteilung nach Gartner von 2011. Mit der Annahme, dass die Mitarbeiter zunehmend selbst das Smartphone auswählen wollen oder aufgrund unterschiedlicher Tätigkeiten

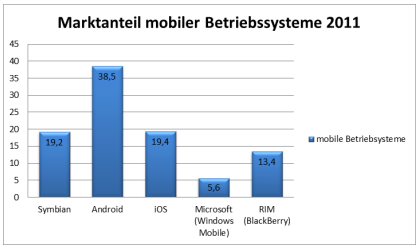


Abbildung 3.13: Übersicht über Marktanteile der Betriebssysteme von mobilen Endgeräten nach Gartner

[23] Auf der CD befindet sich eine weitere Version des Prototypen mit den aufgeführten Offline-Funktionalitäten

verschiedene Smartphones benötigen, haben Unternehmen in ihrem Betrieb mehrere mobile Betriebssysteme zu beachten. Somit muss die Wahl der Art von Anwendungen für mobile Geräte genau geplant werden. Verwendet man native Applikationen müssen diese für jedes Betriebssystem einzeln entwickelt werden, was hohen Aufwand und Kosten mit sich zieht. Rich Internet Anwendungen stellen eine Alternative dar, da sie plattformunabhängig sind. Die Untersuchung mit dem Prototypen hat gezeigt, dass HTML5 genug Möglichkeiten stellt, eine Rich Internet Applikation auch für Unternehmen zu realisieren. Es gibt bei HTML5-Applikationen für mobile Endgeräte dennoch zwei Nachteile. Zum einen erfordern sie momentan immer eine Internetverbindung, sonst sind sie nicht funktionstüchtig. Dieses Problem kann mit besserer Browserunterstützung der HTML5 Offline API behoben und Applikationen, die nicht zwingend eine Internetverbindung dauerhaft brauchen[24] können, auch offline verfügbar werden. Dies führt aktuell zum weitaus größeren Problem: Es gibt bisher keine einheitliche Browserunterstützung. Damit wird der Vorteil der Plattformunabhängigkeit aufgehoben, da man trotz HTML5 unterschiedliche Lösungen finden muss. Dies ist jedoch ein temporäres Problem. Wenn HTML5 Standard wird, können durch HMTL5 für Unternehmen Rich Internet Applikationen erstellt werden. Das ermöglicht die Erstellung plattformunabhängiger Anwendungen auch für mobile Endgeräte. Somit ist ein standardisiertes HTML5 eine optimale Lösung für den Wandel in Unternehmen.

[24] Zum Beispiel ein Zeichenprogramm

4 Ausblick

Durch diese Untersuchung konnten die HTML5-Elemente als gute Theorie mit schlechter Browserunterstützung analysiert werden. Es wurde aufgezeigt, dass die Unterstützung der mobilen Browser nahezu gleich fortgeschritten ist, wie die der Standard-Browser, mit Ausnahme von technischen Unterschieden, wie dem des Touchdisplays, das eine Unterstützung von HTML5 Drag-and-Drop verhindert. Ausschlaggebend für den Erfolg von HTML5 ist, wie sich die Browserunterstützungen weiter entwickeln werden. Während dem Erstellen der Arbeit konnte bereits ein positiver Trend und ein immer schneller werdender Entwicklungszyklus der Browser festgestellt werden. Einige Browser brachten während der Untersuchung Versionen heraus, die die Unterstützung von HTML5 verbesserten. Die Browser scheinen um den besten HTML5-Support zu wetteifern. Firefox zum Beispiel hat in diesem Jahr schon drei neue Versionen herausgebracht und an der nächsten wird bereits gearbeitet. Jede Version bezog auch neue Unterstützungen von HTML5 ein. Selbst Microsoft, das mit dem Internet Explorer bisher weniger Standards unterstützte, brachte mit dem Internet Explorer 9 einen Browser heraus, der sich auch mit den anderen in Sachen HTML5 messen kann. Die Zukunft von HTML5 scheint demnach aus mehreren Gründen gesichert. Zum einen besteht die WHATWG aus mehreren Browserherstellern und damit ist ihr Interesse an HTML5 deutlich. Außerdem entwickeln große Webportale wie Google und Facebook HTML5-Apps und gehen damit das Risiko ein, dass ältere Browser diese nicht darstellen können. Dies zeigt ein großes Interesse auch auf Seite der Entwickler. Ein weiterer Aspekt, der für HTML5 spricht, ist die Plattformunabhängigkeit, die besonders im mobilen Bereich ein großer Vorteil ist. Wenn HTML5 Standard ist, wird auch eine einheitliche Browserunterstützung vorhanden sein und den Vorteilen von HMTL5 wird nichts mehr im Wege stehen.

5 Anhang

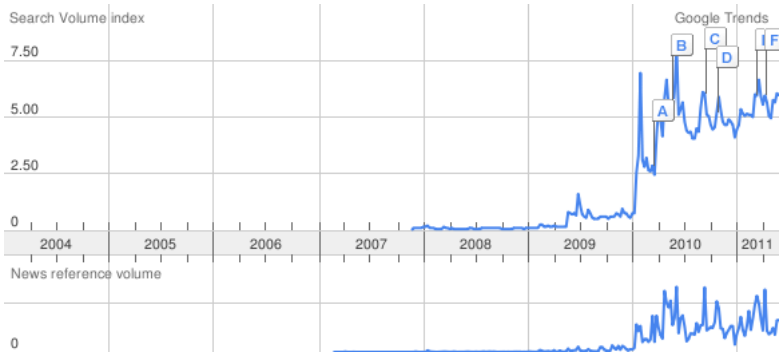


Abbildung 5.1: Verteilung der Suchanfragen zu HTML5 nach Google Trends

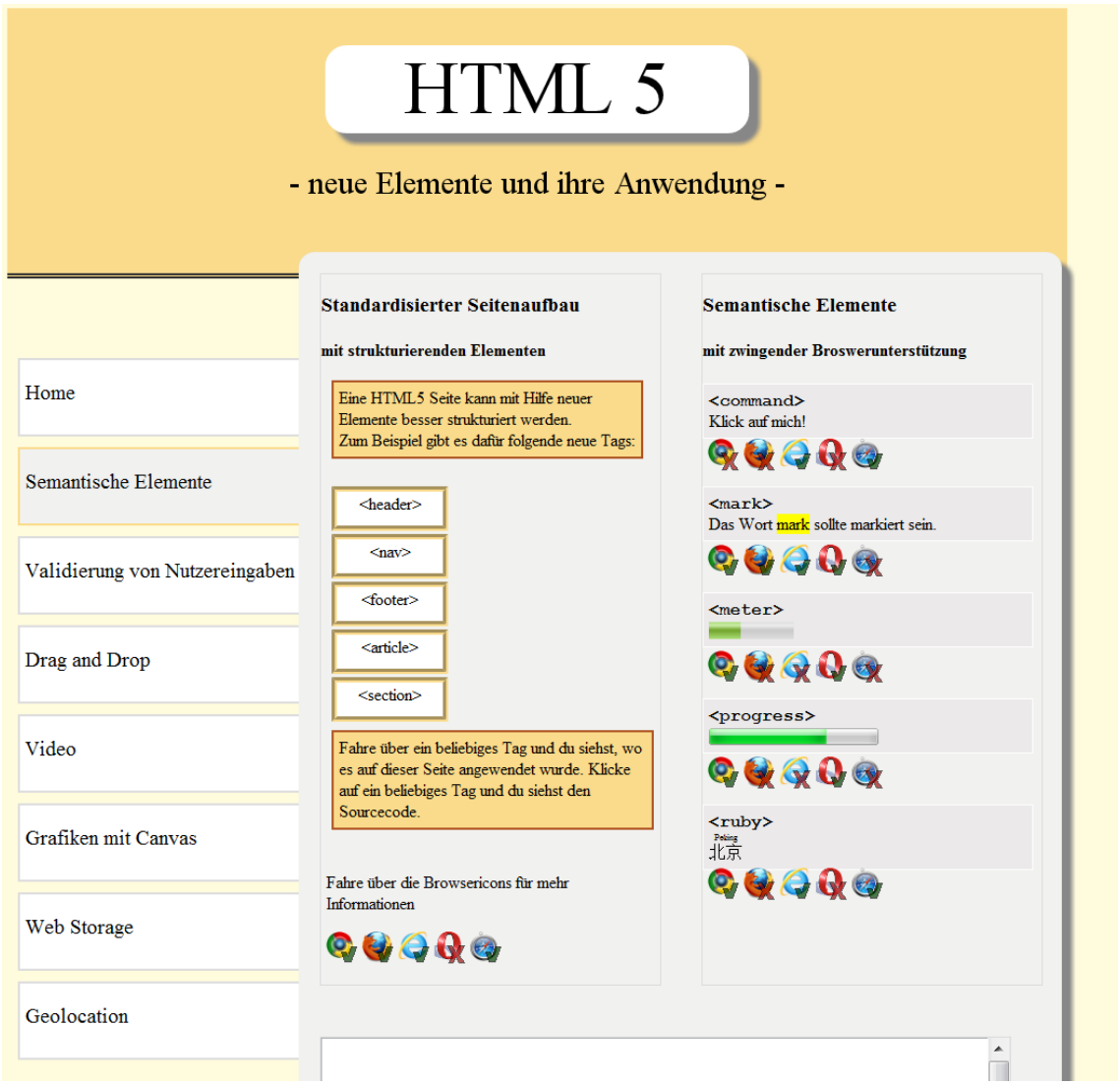


Abbildung 5.2: Screenshot der Seite seitenaufbau.html der Evaluierungsseite mit Google Chrome geöffnet

Abbildung 5.3: Screenshot der Zeichenfunktion der Evaluierungsseite mit Firefox geöffnet

Untersuchtes Element	Browser	realisiert	Kommentar
Semantische Elemente	Chrome	Teilweise	-
	Firefox	Teilweise	-
	Internet Explorer	Teilweise	-
	Opera	Teilweise	-
	Safari	Teilweise	-
HTML5-Formulare	Chrome	Ja	-
	Firefox	Teilweise	Nur `url` und `email`
	Internet Explorer	Nein	-
	Opera	Teilweise	Nur `email` und ein selbstdefiniertes Format
	Safari	Ja	Falsche Eingaben werden nicht als solche dargestellt und ein Absenden falscher Eingaben wird nicht verhindert.
Drag-and-Drop	Chrome	Ja	-
	Firefox	Ja	-
	Internet Explorer	Ja	-
	Opera	Nein	-
	Safari	Ja	-
Video	Chrome	Ja	Unterstützt H.264- und Theora-Videos
	Firefox	Ja	Unterstützt nur Theora-Videos
	Internet Explorer	Ja	Unterstützt nur H.264-Videos
	Opera	Ja	Unterstützt nur Theora-Videos
	Safari	Nein	Offiziell ja, im Test jedoch nicht
Canvas	Chrome	Ja	-
	Firefox	Ja	-
	Internet Explorer	Ja	Aber: falsche Implementierung der `arcTo`-Methode
	Opera	Ja	Aber: falsche Implementierung der `arcTo`-Methode
	Safari	Ja	-
Web Storage	Chrome	Ja	-
	Firefox	Ja	-
	Internet Explorer	Ja	-
	Opera	Ja	-
	Safari	Ja	-
Geolocation	Chrome	Teilweise	-
	Firefox	Teilweise	-
	Internet Explorer	Teilweise	-
	Opera	Teilweise	-
	Safari	Teilweise	Kann keine Position nur mit IP-Adresse finden
Web Sockets	Chrome	Ja	-
	Firefox	Ja	Aber per default deaktiviert
	Internet Explorer	Nein	-
	Opera	Ja	Aber per default deaktiviert
	Safari	Ja	-

Tabelle 5.1: Übersicht über Browserunterstützungen der neuen Elemente (Browserversionen: Google Chrome 12.0.742.91; Firefox 5.0; Windows Internet Explorer9 9.0.8112.16421IC; Opera 11.11; Safari 5.0.5(7533.21.1))

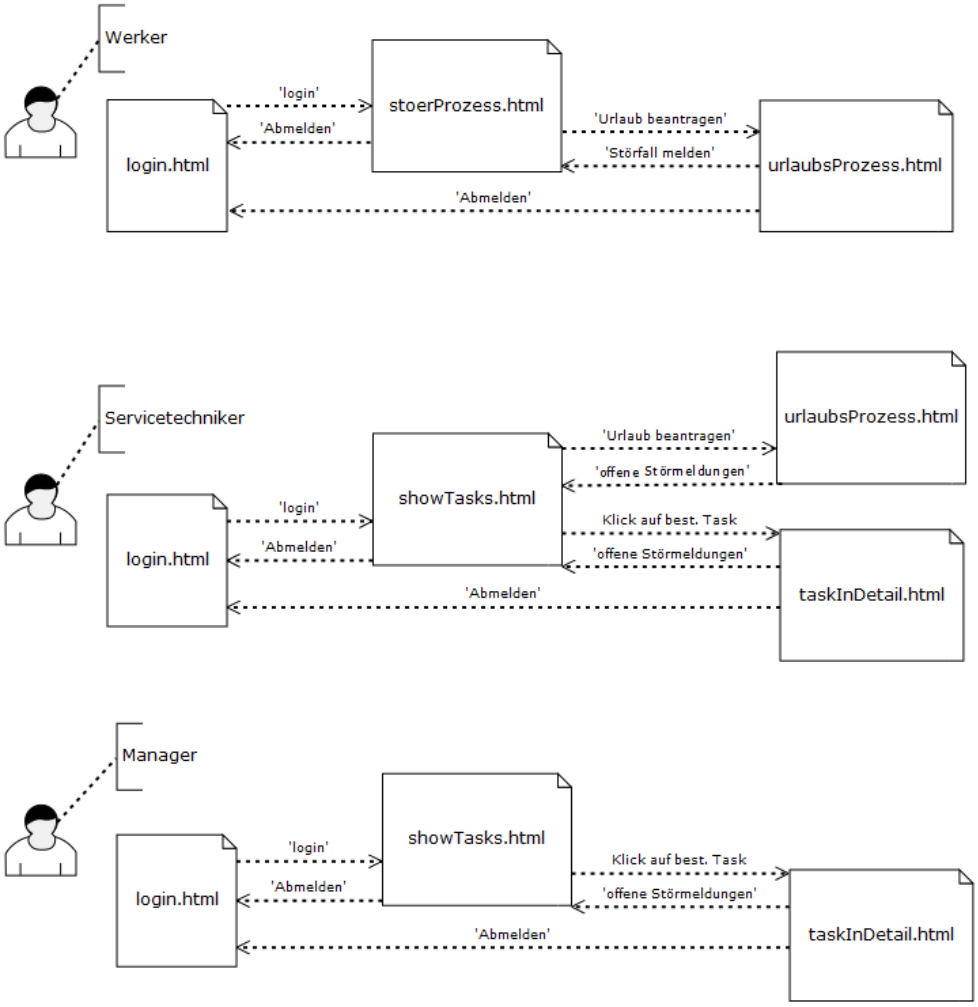


Abbildung 5.4: Übersicht der Zusammenhänge von den HTML-Seiten je nach Nutzer

	Firefox	Safari	Opera
Semantische Elemente	✓	✓	✓
`<command>`	-	✓	-
`<mark>`	✓	-	✓
`<meter>`	-	-	✓
`<progress>`	-	-	ja, aber kein bewegter Balken
`<ruby>`	-	✓	-
Validierung allgemein	eine falsche Eingabe wird abgeschickt. Nur wenn die falsche Eingabe nicht mehr fokussiert ist, werden Fehler angezeigt	invalid-Event wird gefeuert, aber keine Browser-Fehlermeldungen	✓
`email`	✓	✓	✓
`url`	✓	✓	✓
`number`	-	✓	teilweise
`range`	-	-	✓
`color`	-	✓	✓
`date`	-	✓	✓
`time`	-	✓	✓
Drag-and-Drop	Grundsätzlich auf Touchscreens nicht möglich		
`<video>`	✓Theora	✓H.264. Aber `autoplay` geht nicht	✓H.264. Aber `autoplay` geht nicht und `loop` führt zu Endlosschleife, da Vollbild
`<canvas`	✓	✓	✓
Bild	grobpixelig	gut	gut
Zeichenprogramm	Grundsätzlich nicht möglich Striche zu malen, nur Punkte		
Animation	grobpixelig	✓, pausiert aber bei Verschieben des Bildausschnitts	✓, pausiert aber bei Verschieben des Bildausschnitts
Web Storage	✓	✓	✓
Geolocation	✓aber Markierung von Google Maps wird nicht angezeigt	✓	✓
Web Sockets	-	✓	✓

Tabelle 5.2: Browserunterstützungen der mobilen Browser (Browserversionen: Opera Mobile 11.00 für Android; Firefox 5.0 für Android, Safari Mobile 5.0.2 für Ipad

	Firefox	Chrome	Opera	Safari
Störmeldung erstellen	✓	funktioniert, aber mit benutzerdefinierten Validierungen werden keine Fehlermeldungen angezeigt	✓	funktioniert, aber nach Verlassen der Seite wird der zuletzt erstellte Task ein weiteres Mal generiert; Keine Fehlermeldungen
Urlaubsantrag	funktioniert, aber keine Unterstützung des Eingabetyps `date`	funktioniert, aber ohne selbstdefinierten Fehlermeldungen	✓	funktioniert, aber keine Unterstützung des Eingabetyps `date`
Anzeige der Taskliste	✓	✓	✓	✓
Abspielen des Tons bei neuer Aufgabe	✓	✓	✓	Nein
Detailanzeige + Aufgabe bearbeiten	✓	✓	✓	funktioniert, aber es kann kein Video angezeigt werden
Zusatzinformationen	-	-	-	Der Inhalt des Elements `progress` kann nicht angezeigt werden; sehr selten werden zum Teil über 10000 Aufgaben beim Abmelden erstellt
Offline-Betrieb	funktioniert, aber instabil und es kann nur der manuell eingestellte Internetstatus erkannt werden	✓	Daten werden für den Offline-Gebrauch gespeichert; Programm kann durch Fehler beim Anmeldemenu nicht mehr ausgeführt werden	nein, sehr instabil

Tabelle 5.3: Browserunterstützungen von Funktionen des Prototyps (Browserversionen: Google Chrome 13.0.782.215m; Firefox 5.0; Opera 11.50; Safari 5.0.5(7533.21.1))

Abbildungsverzeichnis

2.1 Häufigkeiten von Suchanfragen unterschiedlicher HTML5-Begriffe nach Google Trends . . . 3
2.2 Beispiel aus der Evaluierungsseite: Darstellung bestimmter neuer Elemente . . . 4
2.3 HTML5-Formular in Opera . . . 6
2.4 Beispielcode, mit dem ein Video in HTML eingebunden werden kann . . . 9
2.5 Beispielcode für Alternativquellen mit dem source-Element . . . 10
2.6 JavaScript-Zugriff auf Canvas-2D-Kontext . . . 10
2.7 Darstellung der arcTo()-Methode im Vergleich . . . 11
3.1 Veranschaulichung eines Urlaubsantragsprozesses . . . 16
3.2 Veranschaulichung eines Störmeldungsprozesses . . . 16
3.3 Ablauf einer Störmeldung . . . 16
3.4 Klassendiagramm des Prototyps . . . 17
3.5 Login-Interface . . . 18
3.6 Taskansicht . . . 18
3.7 Detailansicht und Erstellungsformular einer Störmeldung . . . 19
3.8 Nötige Klassen für einen Web-Socket-Server . . . 20
3.9 Codeausschnitt des Authentifizierungvorgangs des Nutzers . . . 21
3.10 Schematischer Ablauf einer Taskanforderung . . . 22
3.11 Codeausschnitt zum Abspielen einer Audiodatei . . . 23
3.12 Manifest-Datei cache.manifest für den Prototypen . . . 25
3.13 Übersicht über Marktanteile der Betriebssysteme von mobilen Endgeräten nach Gartner . . . 25
5.1 Verteilung der Suchanfragen zu HTML5 nach Google Trends . . . 29
5.2 Screenshot der Seite seitenaufbau.html der Evaluierungsseite mit Google Chrome geöffnet . . . 30
5.3 Screenshot der Zeichenfunktion der Evaluierungsseite mit Firefox geöffnet . . . 31
5.4 Übersicht der Zusammenhänge von den HTML-Seiten je nach Nutzer . . . 33
5.5 Ordnerstruktur der beigelegten CD . . . VI

Tabellenverzeichnis

2.1 Browserunterstützung von neuen semantischen Elementen in HTML5 4
2.2 Browserunterstützung der neuen Eingabetypen 5
2.3 Übersicht über Browser und Videoformate 10
5.1 Übersicht über Browserunterstützungen der neuen Elemente (Browserversionen: Google Chrome 12.0.742.91; Firefox 5.0; Windows Internet Explorer9 9.0.8112.16421IC; Opera 11.11; Safari 5.0.5(7533.21.1)) 32
5.2 Browserunterstützungen der mobilen Browser (Browserversionen: Opera Mobile 11.00 für Android; Firefox 5.0 für Android, Safari Mobile 5.0.2 für Ipad 34
5.3 Browserunterstützungen von Funktionen des Prototyps (Browserversionen: Google Chrome 13.0.782.215m; Firefox 5.0; Opera 11.50; Safari 5.0.5(7533.21.1)) . 35

Inhalt der beigelegten CD

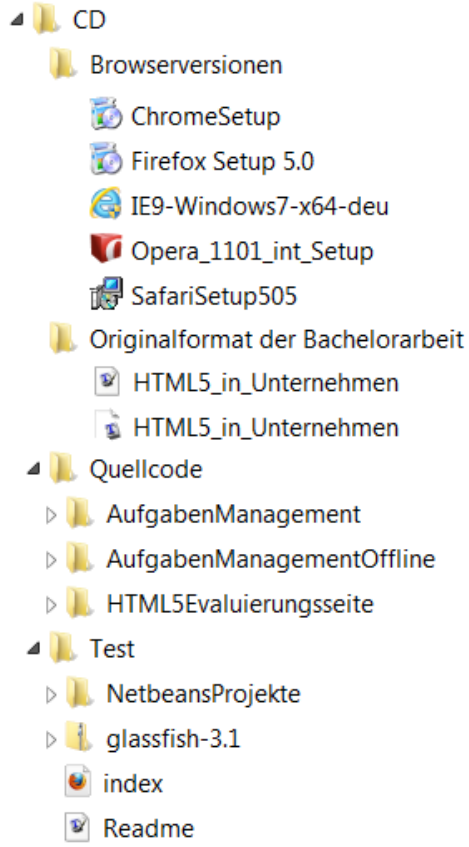


Abbildung 5.5: Ordnerstruktur der beigelegten CD

Die CD enthält vier Ordner. Da immer schneller neue Browserversionen herausgebracht werden, wurden im Ordner `Browserversionen` Setupdateien der untersuchten Browserversionen abgelegt. Dies ist insbesondere für den Browser Firefox relevant, da für ihn kurz vor Fertigstellung der Arbeit die Version 6 herausgebracht wurde, die Web Sockets effektiv nicht unterstützt. Um den Prototypen mit dem Firefox zu testen muss aktuell (Stand Anfang September) die Version 5 benutzt werden. Der Ordner `Originalformat der Bachelorarbeit` beeinhaltet die Latex- und PDF-Version der Bachelorarbeit. Im Ordner `Quellcode` befinden sich drei Unterordner für jedes Projekt. `AufgabenManagement` enthält alle Dateien mit Quellcode des Prototypen. `AufgabenManagementOffline` die des offline-fähigen Prototypen. Im Ordner `HTML5Evaluierungsseite` befinden sich alle Dateien mit Quellcode der Seite, die für die Evaluierung von HTML5-Elemente erstellt wurde, und auf die sich die Bachelorarbeit im ersten Teil bezogen hat. Die kompletten Projekte, die während der Entwicklung in Netbeans 6.9.1 entstanden sind, sind im Ordner `Test/NetbeansProjekte` abgelegt. Es gibt auch eine Möglichkeit, die Projekte ohne Netbeans mit dem sich auf der CD befindenen Server Glassfish zu testen. Folgen Sie dazu bitte den Anweisungen der Datei `Readme.pdf` im Ordner `Test`.

Literatur

[1] P. Kröner: HTML5. Webseiten innovativ und zukunftssicher. Open Source Press, München, 2010

[2] M. Pilgrim: HTML5 Up and Running. O'Reilly Media, Inc., Sebastopol, 2010

[3] L. Huang, E. Chen, A. Barth, E. Rescorla, C. Jackson: Transparent Proxies: Threat or Menace? http://webkebi.zany.kr:9001/lib/exe/fetch.php?media=cate_web:websocket_transparent_proxies.pdf, accessed Juni 15, 2011.

[4] J. Lowensohn: YouTube begins HTML5 rollout http://m.news.com/2166-12_3-10438578-248.html, accessed Juni 14, 2011.

[5] I. Hickson: HTML is the new HTML5 http://blog.whatwg.org/html-is-the-new-html5, accessed Juni 14, 2011.

[6] Ludwig-Maximilians-Universität München http://www.uni-muenchen.de/index.html, accessed Juni 14, 2011.

[7] W3C Schools: HTML5 New Elements http://www.w3schools.com/html5/html5_new_elements.asp, accessed Juni 14, 2011.

[8] D. Cederholm et al.: Guide to HTML5 Hiccups http://www.zeldman.com/superfriends/guide/, accesssed Juni 14, 2011.

[9] D. Melanchthon: Gastartikel: HTML5-Geolocation http://blogs.technet.com/b/sieben/archive/2011/03/10/gastartikel-html5-geolocation.aspx, accessed Juni 14, 2011.

[10] A. Popescu: Geolocation API Specification http://www.w3.org/TR/geolocation-API/, accessed Juni 14, 2011.

[11] Opera Software: Geolocation http://help.opera.com/Windows/11.10/de/geolocation.html, accessed Juni 14, 2011.

[12] W3C Browser Geolocation http://whatismyipaddress.com/w3c-geolocation, accessed Juni 14, 2011.

[13] Webanalyse: Verbreitung von Webtechnologien und Plugins http://www.webmasterpro.de/portal/webanalyse-technologien.html, accessed Juni 14, 2011.

[14] Adobe Flash Player http://www.adobe.com/de/products/flashplayer/, accessed Juni 14, 2011.

[15] S. Jobs: Thoughts on Flash http://www.apple.com/hotnews/thoughts-on-flash/, accessed Juni 14, 2011.

[16] W3C: HTML5 <video> Tag http://www.w3schools.com/html5/tag_video.asp, accessed Juni 14, 2011.

[17] http://www.evodion.de/opencms/export/evodionIT/General/Glossar/p39.html, accessed Juni 14, 2011.

[18] http://www.google.com/intl/de-DE/events/io/2009/sessions/GoogleWavePoweredByGWT.html, accessed Juni 14, 2011.

[19] http://www.w3.org/Protocols/rfc2616/rfc2616-sec8.html#sec8.1.4, accessed Juni 14, 2011.

[20] http://www.golem.de/1104/82797.html, accessed Juni 14, 2011.

[21] I. Hickson: The WebSocket API http://dev.w3.org/html5/websockets/, accessed Juni 14, 2011.

[22] http://de.wikipedia.org/wiki/WebSocket, accessed Juni 14, 2011.

[23] P. Lubbers, F. Greco: HTML5 Web Sockets: A Quantum Leap in Scalability for the Web http://websocket.org/quantum.html, accessed Juni 15, 2011.

[24] C. Heilmann: WebSocket disabled in Firefox 4 http://hacks.mozilla.org/2010/12/websockets-disabled-in-firefox-4/, accessed Juni 15, 2011.

[25] Heise.de: W3C drückt bei HTML5 aufs Tempo http://www.heise.de/developer/meldung/W3C-drueckt-bei-HTML5-aufs-Tempo-1250697.html, accessed Juni 30, 2011.

[26] Google Chrome-Hilfe: Standortfreigabe https://www.google.com/support/chrome/bin/answer.py?answer=142065&hl=de, accessed Juni 22, 2011.

[27] WHATWG: What's this I hear about 2022 http://wiki.whatwg.org/wiki/FAQ#What.27s_this_I_hear_about_2022.3F, accessed Juni 30, 2011.

[28] JAXenter: Wir werden HTML5 offiziell in JSF 2.2 unterstützen http://it-republik.de/jaxenter/artikel/Wir-werden-HTML5-offiziell-in-JSF-2.2-unterstuetzen-3862.html, accessed Juli 12, 2011.

[29] Twitter.com: Ian Hickson http://twitter.com/#!/Hixie/status/4075253361,accessed Juli 21, 2011.

[30] WebHits.de http://www.webhits.de/deutsch/index.shtml?webstats.html,accessed Juli 25, 2011.

[31] Gartner, Inc: Gartner Says Android to Command Nearly Half of Worldwide Smartphone Operating System Market by Year-End 2012 http://www.gartner.com/it/page.jsp?id=1622614, accessed Juli 25, 2011.

[32] http://www.heise.de/newsticker/meldung/Google-Apps-bald-nur-noch-mit-HTML5-1255379.html

[33] http://www.google.de/trends?q=html5+canvas%2C+websocket%2C+webstorage%2Chtml5+geolocation%2C+html5+video&ctab=0&geo=all&date=ytd&sort=0, accessed Juli 5, 2011.

[34] http://www.html5rocks.com/en/tutorials/appcache/beginner/, accessed August 16, 2011.