HTML5

Seminararbeit

Johannes Varga, 0702030

Wien, 19. Juni 2013

Universität:	Wirtschaftsuniversität Wien
LV-Name:	SBWL Kurs V - Management Information Systems (IS-Projektseminar)
LV-Nr.:	4387
LV-Leiter:	ao. Univ. Prof. Mag. Dr. Rony G. Flatscher
Semester:	Sommersemester 2013

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Erklärung

Ich versichere,

- dass ich die Seminararbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfe bedient habe.
- dass ich dieses Seminararbeitsthema bisher weder im In- noch im Ausland (einer Beurteilerin/ einem Beurteiler zur Begutachtung) in irgendeiner Form als Prüfungsarbeit vorgelegt habe.
- dass diese Arbeit mit der vom Begutachter beurteilten Arbeit übereinstimmt.

Ort, Datum

Unterschrift

1 Einleitung

Im Rahmen dieser Seminararbeit wird das Thema *HTML5* behandelt. Es wird versucht, den gesamten Werdegang von Anbeginn des *World Wide Web* bis hin zu *HTML5* aufzuzeigen und die diversen Einflussfaktoren auf die Weiterentwicklung darzustellen. Doch es wird nicht nur auf die geschichtliche Entwicklung eingegangen, sondern auch auf die Neuheiten, die *HTML5* mit sich bringt.

Der **erste Teil** der Seminararbeit beschäftigt sich mit dem geschichtlichen Verlauf. Die Entwicklung der Auszeichnungssprache *HTML* unterlag über die Jahre hinweg einigen Turbulenzen, die erhebliche Auswirkungen auf den Werdegang hatten. Streitigkeiten, Abspaltungen und sogar (Browser-) Kriege wurden um die Auszeichnungssprache *HTML* und dem damit eng verbundenen *World Wilde Web geführt*. Bis es zum heutigen Stand der Dinge kam, war es also ein weiter Weg. Doch wie sieht der heutige Stand der Dinge überhaupt aus? Was macht *HTML5* aus und welche Elemente zählen eigentlich zu *HTML5*? Auf alle diese Fragen wird im ersten Teil der Seminararbeit eingegangen.

Im **zweiten Teil** werden die Neuerungen von *HTML5* vorgestellt. Mit *HTML5* wurden einige neue Elemente eingeführt, veraltete Elemente entfernt, aber auch schon vorhandene Elemente aus Vorgängerversionen erweitert, oder neu erfunden. Um dem Umstand Rechnung zu tragen, dass *HTML5* als Applikationsplattform eingesetzt werden kann, wurden ebenfalls neue *APIs* eingeführt. Die Hauptthemen des zweiten Teils sind also alle Neuerungen, die mit *HTML5* einher gehen.

Nachdem die verlockenden Neuerungen von HTML5 im zweiten Teil vorgestellt wurden, stellt sich natürlich die Frage: Kann ich die Neuerungen heute schon in der Praxis einsetzen? Der **dritte und letzte Teil** der Seminararbeit versucht unter anderem diese Frage zu beantworten. Doch es ist nicht alles Gold was glänzt. Am Ende der Seminararbeit werden noch ein paar Kritikpunkte angeführt, die sich nicht auf HTML5 per se beziehen, sondern eher auf das Umfeld der *HTML5* Weiterentwicklung.

2 Einführung

2.1 Geschichtliche Entwicklung

2.1.1 Die Anfänge

Die wohl bekannteste Person im Zusammenhang mit dem *World Wide Web* ist *Tim Berners-Lee*. Tim Berners-Lee studierte an der *Universität von Oxford* (englisch: „University of Oxford“) Physik und arbeitete ab 1984 als Informatiker am *Genfer Forschungszentrum für Teilchenphysik CERN*. Ein Problem, das am CERN vorherrschte war, dass sich der Informationsaustausch aufgrund einer unterschiedlichen Netzwerkinfrastruktur sehr schwer realisieren ließ [Wikg].

Um diesem Problem entgegenzuwirken, entwickelte Tim Berners-Lee eine eigene Software namens *Enquire* (deutsch: „sich erkunden“), die sich einer von ihm definierten Auszeichnungssprache bediente. Diese Auszeichnungssprache basierte auf dem Standard von *SGML* (*Standard Generalized Markup Language*)[1]. Im Jahr 1988 entschloss sich Tim Berners-Lee aus dieser Software ein System zu entwickeln, das computerübergreifend zum Einsatz kommt. Er reichte 1989 einen Projektvorschlag am CERN ein. 1990 bekam das Projekt seinen endgültigen Namen *World Wide Web*. [Selb; Pag]

Im gleichen Jahr erstellte er ebenfalls die 3 Spezifikationen, die für das System ausschlaggebend waren:

- **HTTP Protokoll** (*Hypertext Transfer Protocol*)
  Zur Kommunikation zwischen Web-Server und Web-Client.
- **URI** (*Universal Resource Identifier*)
  Adressierung beliebiger Dateien und Datenquellen im Internet.
- **HTML** (*Hypertext Markup Language*)
  Auszeichnungssprache für Web-Dokumente.

Neben den 3 oben angeführten Hauptelementen des *World Wide Web*, entwickelte Tim Berners-Lee ebenfalls den ersten Web-Server (*httpd*) bzw. den ersten Web-Browser (*WorldWideWeb*, später: *Nexus*) für das Betriebssystem *NeXTStep*. [Wiki; Selb]

[1] SGML ist eine Meta-Sprache, mit der es möglich ist Auszeichnungssprachen zu definieren.

2.1.2 Der Browserkrieg

Die Entwicklung im Bereich des *World Wide Web* schritt nach den Anfängen durch Tim Berners-Lee rasch voran. Besonderen Erfolg hatte das Unternehmen *Netscape*, mit seinem Gründer *Marc Andreessen*. In den 1990er Jahren dominierte der von diesem Unternehmen entwickelte Web-Browser *Netscape Navigator* den Browsermarkt. Im Jahr 1995 erreichte der *Netscape Navigator* sogar einen Marktanteil von 80 Prozent. [Pak08; Web]

Motiviert durch die schnelle Verbreitung der Web-Browser und der damit verbundenen Änderung der Computernutzung, auch in den breiteren Bevölkerungsschichten, begann *Microsoft* ebenfalls mit der Entwicklung eines eigenen Web-Browsers namens *Internet Explorer*. Ab dieser Entscheidung wurde eine Zeit eingeläutet, die heute als Zeit des Browserkrieges bekannt ist. Die beiden Unternehmen *Microsoft* und *Netscape* versuchten mittels unterschiedlicher Weiterentwicklungen der beiden Produkte sich vom Konkurrenten abzuheben und sich somit Marktanteile zu sichern bzw. zu verteidigen. Ein Vorteil, den das Unternehmen *Microsoft* ausnutzte war, dass das von Microsoft vertriebene Produkt *MS Windows* ein weit verbreitetes Betriebssystem war und somit eine gute Gelegenheit bot, den *Internet Explorer* mit dem Betriebssystem zu koppeln, um somit den Marktanteil des *Internet Explorers* zu erhöhen. Diesem Vertriebskanal war zu verdanken, dass der *Internet Explorer* 2002 einen Marktanteil von über 90 Prozent erreichte. Dieses Vorgehen wurde jedoch durch einen US-Kartellprozess sanktioniert und *Microsoft* musste hohe Strafzahlungen leisten.

Durch die hohe Verbreitung des *Internet Explorers* wurde der *Netscape Navigator* zunehmend vom Markt verdrängt. Die hohe Verbreitung führte sogar soweit, dass gewisse Funktionen, die von Websites angeboten wurden, nur in Verbindung mit dem *Internet Explorer* nutzbar waren. Der *Internet Explorer* (Version 6) wich jedoch teilweise erheblich vom HTML Standard ab. [Wika]

Im Jahr 1998 wurde der Quelltext des *Netscape Navigators* veröffentlicht. Ab diesem Zeitpunkt wurde nicht nur die Weiterentwicklung in die Hände der Open Source Gemeinde gegeben, sondern der Web-Browser wurde ebenfalls umbenannt in *Firefox*. Es wurde ebenfalls das *Mozilla Projekt* ins Leben gerufen, das sich bis heute um die Weiterentwicklung des Web-Browsers *Firefox* kümmert. [Pak08]

In den 2000er Jahren stieg die Bedeutung alternativer Browser immer weiter an. Als Beispiel wäre der schnelle Aufstieg des Browsers *Chrome* zu nennen, der vom Unternehmen *Google* seit 2008 veröffentlicht wird und immer größeren Anklang bei den Endbenutzern findet. Ein weiterer Vertreter ist der Web-Browser *Opera*, der seit September

2005 von einem norwegischen Unternehmen angeboten wird. [Wikb; Wikf]

2.1.3 Der “neue“ Browserkrieg

Seit dem Aufkommen alternativer Browser wurde der Browserkrieg nicht mehr so aggressiv geführt wie in den frühen Jahren des *World Wide Web*. Dennoch gibt es Anzeichen für ein erneutes Aufflammen des Browserkrieges. Wie im Blogeintrag von *Mozilla* zu lesen ist, wird befürchtet, dass *Microsoft* wieder einmal seinen hohen Verbreitungsgrad hinsichtlich vorinstalliertem Betriebssystem ausnutzt, um Drittanbieter von Web-Browsern auszuschließen.

Das neue Betriebssystem *Windows for the ARM processor architecture* (*Windows RT*), das überwiegend in Tablet-PCs eingesetzt wird, beinhaltet Beschränkungen, die besagen, dass keine Web-Browser von Drittanbietern in der “klassischen“ Desktop Oberfläche eingesetzt werden dürfen. Daraus folgt, dass dieses Privileg nur dem Internet Explorer vorbehalten ist. [vgl. And12]

Bei *Mozilla* befürchtet man, dass das ein weiterer Schritt in Richtung Plattform Lock-in ist. Gleichzeitig wird an *Microsoft* appelliert, sich diese Limitierung nochmals zu überlegen. Auch *Google* schloss sich den Aussagen von Mozilla an. [Sha12]

2.2 W3C und WHATWG

2.2.1 Entstehung des W3C

Nachdem *Tim Berners-Lee* die Grundbausteine für das *World Wide Web* gelegt hatte, begannen mehrere Unternehmen Web-Browser zu entwickeln. Wie unter 2.1.2 beschrieben, führte das zu einem regelrechten Krieg um die Marktführerschaft, wobei sich die Unternehmen durch eigene, proprietäre Technologien abzugrenzen versuchten.

Tim Berners-Lee war schon von Beginn an bewusst, dass eine inkonsistente Nutzung seiner 3 Haupttechnologien (*HTML*, *URI* und *HTTP*) in die falsche Richtung führen würde, nämlich weg von dem universellen Charakter des *World Wide Web*. Infolgedessen, wurde am 1. Oktober 1994 das *W3C* (*World Wide Web Consortium*) gegründet. Das *W3C* machte sich zur Aufgabe, einen einheitlichen Standard gemeinsam mit Mitgliedern der einzelnen IT-Unternehmen zu schaffen und weiterzuentwickeln. Der Gedanke hinter diesem gemeinsamen Standard ist, dass jedem Menschen, egal welche Software-, Hardwarekomponenten oder Netzwerkinfrastruktur vorhanden ist, eine globale einheitliche Kommunikationsform zur Verfügung stehen soll. [Wikj; W3oa; W3od]

Das *W3C* besteht zur Zeit aus 79 Kern-Mitgliedern (Stand: 2011) und 287 Mitgliedern (Stand: 29. März 2013), die bei unterschiedlichen IT-Unternehmen beschäftigt sind. [W3ob; W3oc]

2.2.2 Entwicklung von HTML bis HTML5

Die Entwicklung der HTML Spezifikation schritt in den Anfangsjahren rasch voran. HTML 2.0, 3.2, 4.0 und 4.01 wurden schnell hintereinander definiert. Die Version 4.01 (verabschiedet: 1999) stellt dabei die letzte große Spezifikation dar, die auf *SGML* basiert.

Tabelle 1: Entwicklung von HTML zu HTML5 (W3C)

Bezeichnung	**Datum**	**Beschreibung/Erneuerungen**
HTML	3. November 1992	Erste HTML Spezifikation; nur auf Text ausgerichtet
HTML	30. April 1993	Fette und kursive Darstellung; Bildintegration
HTML 2.0	November 1995	Neue Elemente; u.a. Formulare
HTML 3.2	14. Januar 1997	Tabellen, Textfluss um Bilder, Einbindung von Applets
HTML 4.0	18. Dezember 1997	Stylesheets, Skripte, Frames; Trennung in Strict, Frameset und Transitional
HTML 4.01	24. Dezember 1999	Mehrere kleine Korrekturen an HTML 4.0
XHTML 1.0	26. Januar 2000	Neuformulierung von HTML 4.01 mit Hilfe von XML
XHTML 1.1	31. Mai 2001	Aufteilung in Module; Wegfall von Frameset und Transitional
XHTML 2.0	26. Juli 2006 (geschlossen)	Sollte nicht mehr auf HTML 4.01 basieren; Trennung von Auszeichnung und Stil; zu Gunsten von HTML5 eingestellt
HTML5	17. Dezember 2012 (Candidate Recommendation)	Auf Basis von HTML 4.01 und XHTML 1.0

Aufgrund der Tatsache, dass die Auszeichnungssprache XML immer beliebter bei den Web-Entwicklern wurde bzw. immer mehr Unternehmen XML als Grundlage für ihre Dateiformate verwendeten, entschloss sich das *W3C*, unter Verwendung von XML, HTML neu zu definieren. Doch nicht nur die Kompatibilität, sondern auch die Einfach-

heit von XML im Gegensatz zu SGML war ausschlaggebend für die Umstellung. Um die Übersichtlichkeit zu bewahren, wurde nicht die vorhandene Namensgebung weiter verwendet, sondern der neue Name *XHTML* eingeführt. [Wikd; Wikc; Seld]

In der Entwicklung von HTML wurden immer wieder Zusatzerweiterungen des HTML Standards vorgestellt. Manche dieser Spezifikationen fanden jedoch keine allgemeine Zustimmung. Beispielsweise wurden die Spezifikationen HTML+ und HTML 3.0 niemals von den Web-Browser Herstellern implementiert.

XHTML 1.0 führte keine großen Neuerungen ein, sondern hatte vorwiegend die Aufgabe, die auf der Meta-Sprache SGML basierende Version HTML 4.01, unter Verwendung von XML, neu zu definieren. XHTML 1.0 stellt somit die gleiche Funktionalität wie HTML 4.01, nur unter der Berücksichtigung des XML Regelwerkes, bereit.

Die Weiterentwicklung innerhalb des *W3C* konzentrierte sich ab diesem Zeitpunkt nur noch auf XML-basierte HTML Standards. Als nächster Schritt wurde eine Modularisierung angestrebt. Der XHTML Standard wurde in verschiedene Module aufgesplittet, die unter dem XHTML 1.1 Standard zusammengefasst wurden.

Set der Version HTML 4.01 gab es keine großen Änderungen im Sprachbestand der Spezifikationen. Das W3C strebte eine Neuentwicklung an. Die XHTML 1.1 Spezifikation sollte von Grund auf neu entwickelt werden und den Namen XHTML 2.0 tragen. [Ber+12b]

Ein großes Problem war, dass die neue Spezifikation XHTML 2.0 sehr strikt nach dem XML-Regelwerk definiert wurde und viele Websites, die auf XHTML 1.1 aufbauten, nicht ohne erhebliche Änderungen auf XHTML 2.0 umgestellt werden konnten. Die fehlende Abwärtskompabilität, sowie die hohe Komplexität des XHTML 2.0 Standards, kam bei vielen Web-Entwicklern nicht gut an. [Sela]

Bei einem W3C Workshop im Jahr 2004 wurde von den Web-Browser Herstellern Mozilla und Opera ein Vorschlag eingebracht, der sich mit der Weiterentwicklung von HTML 4 beschäftigte. Das *W3C* wies den Vorschlag zurück, mit der Begründung, dass der Standard weiterhin auf XML basieren sollte. Kurz darauf schlossen sich die Firma Apple, Mozilla und Opera zu der *WHATWG* (*Web Hypertext Application Technology Working Group*) zusammen und entwickelten ab diesem Zeitpunkt eine eigene Spezifikation, die auf HTML 4.01 bzw. auf XHTML 1.0 basiert. Sie wurde zuerst unter dem Namen *Web Applications 1.0* und später unter *HTML5* veröffentlicht.

Im Jahr 2006 kamen die Mitglieder des *W3C* jedoch zu dem Entschluss, eine Arbeitsgruppe zu gründen, die auf der Basis eines *Fork*[2] der *WHATWG* Spezifikation die HTML Spezifikation weiterentwickeln soll. Somit gab es innerhalb des *W3C* zwei Arbeitsgruppen, die unterschiedliche Ziele verfolgten. Eine Arbeitsgruppe arbeitete an XHTML 2.0 und die andere an HTML5. Im Jahr 2009 gab das *W3C* bekannt, dass die Arbeit an XHTML 2.0 eingestellt wird und nur noch die HTML5 Arbeitsgruppe weitergeführt wird. [Hic13f; Ber+12b]

2.2.3 Entwicklung der Spezifikationen

Beide Arbeitsgruppen (*W3C* und *WHATWG*) arbeiten seit 2009 an ihrer eigenen HTML5 Spezifikation. Obwohl beide Arbeitsgruppen auf der gleichen Grundlage arbeiten, hat sich eine unterschiedliche Herangehensweise an die Erstellung der Spezifikationen herauskristallisiert. Das *W3C* hat sich zur Aufgabe gemacht, fertige *Recommendations* (deutsch: „Empfehlungen“) zu veröffentlichen, die als abgeschlossene Dokumente verabschiedet werden.

Die *WHATWG* bezeichnet ihre Spezifikation als „Living Standard“. Der *Living Standard* ist eine einzige Spezifikation, die ständig weiterentwickelt wird. Die Spezifikation wird in enger Zusammenarbeit mit den Web-Browser Herstellern weiterentwickelt und es werden ständig neue Sprachelemente aufgenommen, oder auch wieder entfernt. [Hic13f]

Am 19. Januar 2011 gab die *WHATWG* bekannt, dass sie ab diesem Zeitpunkt ihre Spezifikation nicht mehr als „HTML5“, sondern nur mehr als „HTML“ bezeichnet. Dieser Schritt unterstreicht das Konzept der *WHATWG* dahingehend, dass sie die Spezifikation als einen fortlaufenden, dynamischen Prozess ansieht. [Hic11]

Um den Web-Entwicklern ihre Arbeit zu vereinfachen wurden *Web Developer Editionen* eingeführt. Der Begriff *Web Developer Edition* bezeichnet eine spezielle Spezifikationsversion, die sich vorwiegend an Web-Entwickler richtet und weniger an Web-Browser Hersteller. Von diesen speziellen Spezifikationen existiert jeweils eine pro Arbeitsgruppe (*W3C* und *WHATWG*). [Ber+13; Hic13a]

[2] Die vorhandene Spezifikation der *WHATWG* wurde „kopiert“ und unabhängig weiterentwickelt.

2.3 Abgrenzung von HTML5

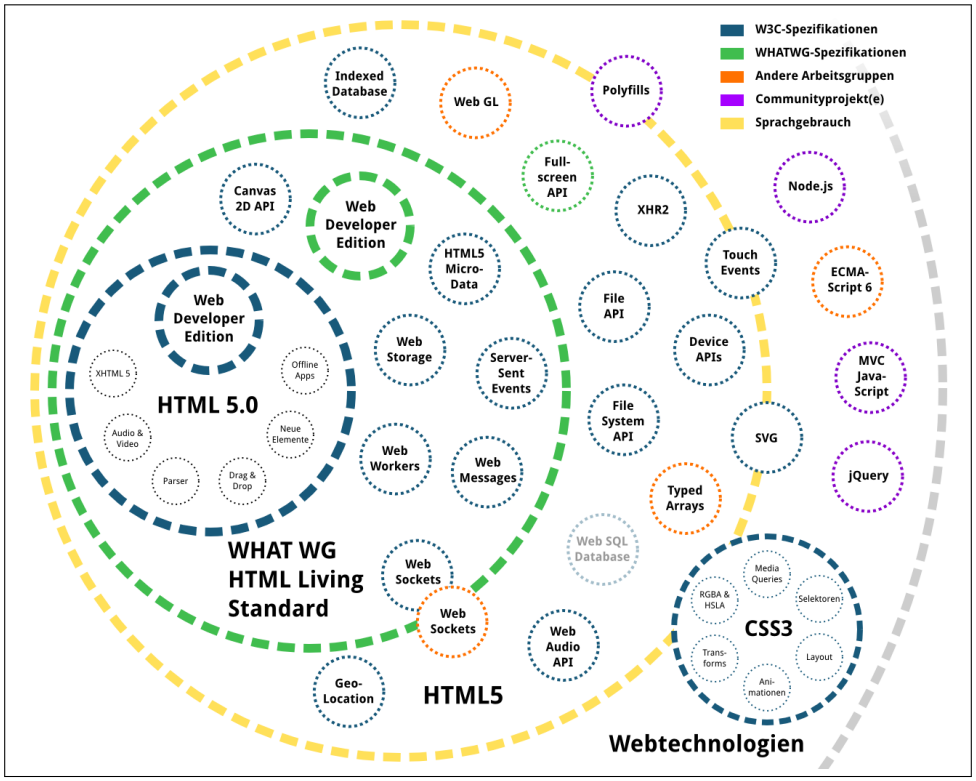

Abbildung 1: HTML5 Universum (Quelle: [Kö13])

Wie im Abschnitt 2.2.3 beschrieben, existieren zum derzeitigen Zeitpunkt (Mai 2013) zwei Arbeitsgruppen, die sich mit HTML5 Spezifikationen beschäftigen.

In Abbildung 1 sieht man, dass derzeit ein gewisser „Wildwuchs“ bei den unterschiedlichen Spezifikationen vorhanden ist. Manche Technologien werden vom *W3C* und gleichzeitig von der *WHATWG* spezifiziert. Andere Elemente wie das *Canvas 2D API* sind zwar in der *WHATWG* Spezifikation vorhanden, jedoch nicht in der *W3C* Spezifikation. Dazu kommen noch Elemente wie *Geolocation*, die in keiner der beiden Spezifikationen zu finden sind, jedoch im allgemeinen Sprachgebrauch ebenfalls zu HTML5 gezählt werden. Diesem „Wildwuchs“ ist zu verdanken, dass sich die beiden Hauptspezifikationen (*W3C* und *WHATWG*) nur mehr als Teilmenge innerhalb des HTML5 Universums eingliedern und nicht mehr, wie von Tim Berners Lee in der Gründungsphase festgelegt, als einheitliche Grundlage fungieren.

Das umfangreiche Verständnis des Begriffs HTML5 schließt also Techniken mit ein, die eigentlich nichts mehr mit den Hauptspezifikationen zu tun haben. Diverse JavaScript Frameworks wie *node.js* oder *jQuery* bzw. *CSS3* (*Cascading Style Sheets Level 3*) werden

oft in einem Atemzug mit HTML5 genannt. Obwohl diese Technologien streng genommen nicht als HTML5 spezifiziert sind, vervollständigen sie jedoch das Bild von HTML5 als Plattform für Web-Applikationen.

Schon die bei der Abspaltung (*WAHTWG* von *W3C*) resultierende ursprüngliche Namensgebung (*Web Applications 1.0*) der neuen Spezifikation zeigt, wohin die Reise gehen soll. HTML entwickelt sich von einer Beschreibung von statischen Elementen, hin zu einer Plattform für Web-Applikationen. Besonders die drei Technologien *HTML5*, *CSS3* und *JavaScript* werden vorwiegend in Kombination eingesetzt, um Web-Applikationen zu erstellen. [Kö11; Ant12]

Eine strikte Trennung, was genau HTML5 ist und was nicht, wird immer schwieriger. Dazu trägt vor allem die rasende Entwicklung im Bereich der Web-Browser bei. Aus diesem Grund werden im Rahmen dieser Seminararbeit ausgewählte HTML5 Elemente beschrieben, die dem allgemeinen Sprachgebrauch zuzuordnen sind. Teile, wie *CSS3* oder diverse *JavaScript Frameworks* werden hier jedoch nicht behandelt.

3 HTML5 Elemente und APIs

3.1 Doctypes

Von SGML/XML werden standardmäßig keine Elemente für HTML definiert. Will jedoch ein Web-Browser ein auf SGML/XML basierendes Dokument anzeigen, ist es wichtig, dass sich der Web-Browser und der Web-Server auf gewisse Konventionen in Bezug auf Elemente, Attribute oder auch auf die generelle Struktur des Dokumentes einigen.

Dieser „Vertrag“ wird bewerkstelligt durch eine sogenannte *DTD* (*Document Type Definition*). Eine DTD ist eine Definition, die angibt, an welche Regeln sich der Web-Browser halten soll, wenn er ein HTML Dokument verarbeitet.

Ein *Doctype* bezeichnet eine Anweisung, die dem Web-Browser signalisiert, an welche DTD er sich halten soll. Der Doctype muss in der ersten Zeile einer HTML Datei angegeben werden. Bei XHTML Dokumenten ist es jedoch möglich, eine Angabe über die XML Version[3] vor dem Doctype einzufügen [Seld].

Bei früheren Versionen von HTML (basierend auf SGML), musste zwingend eine Referenz, meist in Form einer *URL* (*Uniform Resource Locator*), auf eine DTD angegeben werden. Da HTML5 nicht mehr auf SGML basiert, fällt diese Referenzangabe weg. [W3sa]

[3] Optionale Angabe bei XHTML Dateien: *<?xml version="1.0ëncoding=ütf-8"?>*

Der Doctype in HTML5 ist im Vergleich zu den Vorgängerversionen erheblich vereinfacht worden. Die nachfolgende Aufzählung zeigt die einzelnen Bestandteile des HTML5 Doctypes:

- **„<!DOCTYPE“** - case-insensitive
- Ein oder mehrere **Leerzeichen**
- **„html“** - case-insensitive
- (**DOCTYPE legacy string** oder **obsolete permitted DOCTYPE string**) - optional und im Normalfall nicht notwendig
- Kein, ein oder mehrere **Leerzeichen**
- **„>“** - Schließende spitze Klammer

Das Listing 1 zeigt den HTML5 Doctype im Vergleich zu den Doctypes der Vorgängerversionen. Bei HTML 4.01 und XHTML 1.0 gibt es drei verschiedene Varianten des Doctypes, die jeweils andere Auszeichnungsregeln in HTML Dokumenten verlangen. [Selc]

- **Strict**
  Der Einsatz der HTML Elemente wird bei der *strict* Variante sehr streng nach Spezifikation gehandhabt. Kleinste Abweichungen wie: öffnendes * * Tag ohne schließendes * * Tag führen zu einer Verletzung der Syntaxregeln. Weiters werden einige Elemente und Attribute, die für Formatierungen eingesetzt werden, nicht unterstützt.
- **Transitional**
  Die *transitional* Variante ist eine „entschärfte“ Version von *strict*, bei der Text innerhalb des *<body>* Tags auch ohne umschließender Tags möglich ist. Es werden ebenfalls einige wenige Attribute für die Formatierung unterstützt.
- **Frameset**
  Mit dieser Variante des Doctypes wird dem Web-Browser signalisiert, dass das HTML Dokument *Frames* als Strukturierungselemente einsetzt[4]

[4]Frames wurden vor allem in den 1990er Jahren oft für die Strukturierung von Websites eingesetzt. Heute (2013) werden Frames nur noch sehr selten eingesetzt und werden von HTML5 auch nicht mehr unterstützt.

Listing 1: Doctype Definitionen in HTML Dateien

```
1  <!-- HTML5 -->
2  <!DOCTYPE html>
3
4  <!-- HTML 4.01 Strict -->
5  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/
       strict.dtd">
6
7  <!-- HTML 4.01 Transitional -->
8  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/
       TR/html4/loose.dtd">
9
10 <!-- HTML 4.01 Frameset -->
11 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/
       html4/frameset.dtd">
12
13 <!-- XHTML 1.0 Strict -->
14 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/
       xhtml1/DTD/xhtml1-strict.dtd">
15
16 <!-- XHTML 1.0 Transitional -->
17 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
       TR/xhtml1/DTD/xhtml1-transitional.dtd">
18
19 <!-- XHTML 1.0 Frameset -->
20 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/
       xhtml1/DTD/xhtml1-frameset.dtd">
21
22 <!-- XHTML 1.1 -->
23 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/
       DTD/xhtml11.dtd">
```

Wie im Abschnitt 2.2.2 erklärt, baut XHTML 1.0 auf HTML 4.01 auf. In der Praxis bedeutet das, dass ein XHTML Dokument nach XML konformen Regeln erstellt sein muss, um als *wohlgeformt* zu gelten [W3c]. Der Doctype von XHTML 1.1 entspricht dem Doctype von XHTML 1.0 strict, jedoch mit der Möglichkeit Module einzubinden. [W3sa]

Der Begriff *wohlgeformt* (englisch: „well-formed“) bedeutet, dass ein HTML Dokument nach SGML/XML Regeln erstellt wurde und sich auch an diese Regeln hält. Eine Steigerung wird durch die Betitelung eines Dokumentes mit *valide* (englisch: „valid“) erreicht. Durch valide wird signalisiert, dass sich das Dokument an alle speziellen, von SGML/XML abgeleiteten Regeln hält, die in der DTD definiert wurden. Die Regeln in der DTD bilden die „grammatikalische“ Grundlage für das Dokument.

3.2 Neue Elemente in HTML5

3.2.1 Grundlegendes zu HTML Elementen

Ein HTML Element besteht grundsätzlich aus einem *öffnenden Tag* (deutsch: „Marke“) und einem *schließenden Tag*. Innerhalb dieser beiden Tags befindet sich der *Inhalt des HTML Elements*. *Tags* werden immer mit einer spitzen Klammer geöffnet (<) und auch wieder geschlossen (>). Das schließende Tag wird durch einen Schrägstrich (englisch: „slash“), nach der öffnenden spitzen Klammer eingeleitet (</).

Es existieren ebenfalls Elemente, die *keinen Inhalt* besitzen. Diese Elemente bestehen nur aus einem Tag, der vor der schließenden spitzen Klammer einen Schrägstrich beinhaltet (/>) und somit symbolisiert, dass dieses Element nur aus einem Tag besteht. In HTML5 ist dieser schließende Schrägstrich bei Elementen mit nur einem Tag nicht mehr notwendig.

Als *Attribute* werden Angaben innerhalb eines Tags bezeichnet, die das jeweilige Element weiter spezifizieren. [W3sb; W3sj]

Das Listing 2 zeigt mehrere HTML Elemente zur Veranschaulichung.

Listing 2: Syntax von HTML Elementen

```
1  <!-- Das ist ein Kommentar -->
2
3  <!-- Ein HTML Element bestehend aus einem öffnenden und einem schließenden Tag -->
4  <p>Das ist der Inhalt des HTML Elements</p>
5
6  <!-- Ein HTML Element, das keinen Inhalt besitzt -->
7  <img src="tux.png" width="250px" height="300px" />
8
9  <!-- Ein HTML5 Element, das nicht geschlossen werden muss -->
10 <img src="tux.png" width="250px" height="300px">
```

Der im Abschnitt 3.1 behandelte Doctype ist streng genommen kein HTML Element. Er ist rein als Information für den Web-Browser anzusehen. [W3sa]

3.2.2 Die Semantik in HTML Dokumenten

Die einzige Möglichkeit ein HTML Dokument zu strukturieren bzw. Layout-Bereiche zu definieren war unter HTML 4.01 bzw. XHTML 1.0 der Einsatz von * * Blöcken. Ein * * Element besitzt grundsätzlich keine Semantik, die beschreibt, was der Inhalt des Elementes ist. Eine Möglichkeit, Semantik in dieses Element zu integrieren, ist die Angabe eines *id* oder *class* Attributes. Semantik wird hier nur durch das

eingesetzte *id* oder *class* Attribut gewährleistet und nicht durch das eigentliche <*div*> Hauptelement. Das Listing 3 zeigt ein solches Beispiel. [Sta]

Listing 3: Semantik von <*div*> Elementen

```
1  <div id="newseintrag">Text</div>
```

Zwar ist es für Menschen relativ leicht zu erkennen, welchen semantischen Hintergrund das <*div*> Element mit gesetztem *id* Attribut besitzt, ein Computer ist aber nicht in der Lange zu unterscheiden, ob es sich bei diesem <*div*> Block um einen Newseintrag, oder um das Impressum einer Website handelt. Ein weiteres Problem bei den Vorgängerversionen von HTML5 war, dass es ebenfalls wenige semantische Elemente auf der Textebene gab. [W3se]

Der Unterschied zwischen *id* und *class* Attribut ist, dass ein *id* Attribut ein eindeutiger Bezeichner ist und somit nur einmal innerhalb eines HTML Dokumentes eingesetzt werden kann. Ein *class* Attribut kann mehrere Male in einem HTML Dokument eingesetzt werden. Es ist auch möglich, einem HTML Element mehrere *class* Attribute zuzuweisen. [W3sc; W3si]

Mit HTML5 wurden einige neue Elemente eingeführt, die selbst Semantik besitzen und somit keine Attribute zur semantischen Beschreibung benötigen.

3.2.3 Strukturelemente

Die HTML5 Spezifikation definiert einige neue Elemente, die es erlauben, semantische Auszeichnungen in Bezug auf die Website-Struktur einzusetzen. [Sta]

Tabelle 2 zeigt eine Übersicht der neuen Strukturelemente in HTML5.

Im Abschnitt 3.2.2 wurde die Problematik mit den universellen <*div*> Blöcken erläutert. Das Listing 4 zeigt ein kleines Beispiel, wie die Struktur eines Weblogs mit <*div*> Blöcken aussehen könnte.

Wie man sieht, ist die einzige Unterscheidung der einzelnen <*div*> Blöcke, das *id* bzw. *class* Attribut. Ein Problem, dass sich daraus ergibt ist, dass das <*div*> Element mit dem gesetzten *class* Attribut „blogentry“ von einem anderen Webentwickler auch einfach nur mit einem Attributwert „entry“ versehen werden könnte. Daraus folgt, dass es zwar gewisse Konventionen bei der Benennung von *id* oder *class* Attributen geben kann, es aber für den Computer grundsätzlich nicht möglich ist, die einzelnen Elemente aufgrund des Attributs semantisch zu unterscheiden.

Tabelle 2: Übersicht der neuen Strukturelemente

Element	Kurzbeschreibung
<section>	Ein Abschnitt innerhalb eines Dokuments. Es ist dem * * Element sehr ähnlich, besitzt aber dennoch eine andere Semantik.
<nav>	Hauptnavigation auf einer Website.
<article>	Ein Abschnitt innerhalb eines Dokuments, der in sich abgeschlossen ist.
<aside>	Der Inhalt weist einen bedingten Zusammenhang zum Hauptthema der Website auf (Beispiel: Seitenleiste mit Widgets).
<hgroup>	Gruppierung von Überschriften.
<header>	Kopfbereich einer Website, oder eines Abschnittes. Dieses Element kann auch innerhalb eines anderen Elementes eingesetzt werden. (Beispiel: Innerhalb eines *<article>* Elementes, kennzeichnet es den Kopfbereich des jeweiligen Artikels).
<footer>	Fußbereich einer Website, oder eines Abschnittes. Das *<footer>* Element bietet eine ähnliche Funktionalität wie das *<header>* Element.

Listing 4: Seitenaufbau mit * * Elementen

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/
       TR/html4/loose.dtd">
2  <html>
3    <head><title>Blog</title></head>
4    <body>
5      <div id="header"><h1>Blog</h1></div>
6      <div id="navigation">
7        <ul>
8          <li><a href="page1.html">Seite 1</a></li>
9          <li><a href="page1.html">Seite 2</a></li>
10       </ul>
11     </div>
12     <div id="main">
13       <div class="blogentry">
14         <h2>Blogeintrag 1</h2>
15         <p>Das ist der Teaser-Text von Blogeintrag 1</p>
16         <p>Und das ist der Haupt-Text</p>
17       </div>
18       <div class="blogentry">
19         <h2>Blogeintrag 2</h2>
20         <p>Das ist der Teaser-Text von Blogeintrag 2</p>
21         <p>Und das ist der Haupt-Text</p>
22       </div>
23     </div>
24     <div id="sidebar">Nicht zu durchsuchender Inhalt</div>
25     <div id="footer">&copy; Johannes Varga</div>
26   </body>
27 </html>
```

Das Listing 5 zeigt die Umsetzung des gleichen Weblogs mittels der neuen HTML5 Elemente.

Listing 5: Seitenaufbau mit den neuen HTML5 Elementen

```
1  <!DOCTYPE html>
2  <html>
3    <head><title>Blog</title></head>
4    <body>
5      <header><h1>Blog</h1></header>
6      <nav>
7        <ul>
8          <li><a href="page1.html">Seite 1</a></li>
9          <li><a href="page1.html">Seite 2</a></li>
10       </ul>
11     </nav>
12     <section>
13       <article>
14         <h2>Blogeintrag 1</h2>
15         <p>Das ist der Teaser-Text von Blogeintrag 1</p>
16         <p>Und das ist der Haupt-Text</p>
17       </article>
18       <article>
19         <h2>Blogeintrag 2</h2>
20         <p>Das ist der Teaser-Text von Blogeintrag 2</p>
21         <p>Und das ist der Haupt-Text</p>
22       </article>
23     </section>
24     <aside>Nicht zu durchsuchender Inhalt</aside>
25     <footer>&copy; Johannes Varga</footer>
26   </body>
27 </html>
```

Auf den ersten Blick scheint es so, als ob es zwischen dem *<section>* Element und dem *<article>* Element keine semantischen Unterschiede gibt. Das ist aber nicht der Fall. Der Inhalt des *<article>* Elements repräsentiert eine in sich abgeschlossene Einheit von Elementen. Es könnte aus der Dokumentenstruktur herausgelöst werden und dabei keine Semantik verlieren. Das *<section>* Element fasst Elemente zusammen, die in Bezug zueinander stehen und teilt das HTML Dokument in sinngemäße Abschnitte. Eine gegenseitige Verschachtelung der beiden Elemente ist auch möglich. [Sta]

Beispielsweise wäre es in der Umsetzung mit * * Blöcken (Listing 4) nicht möglich, oder nur durch Mehraufwand, die Seitenleiste (Zeile 28) von Durchläufen einer Suchroutine auszuschließen. Durch das neue Element *<aside>* ist es dem Computer möglich, diesen Bereich gezielt anzusprechen und nicht zu durchsuchen.

Obwohl es einige neue semantische Elemente in HTML5 gibt, haben * * Elemente immer noch ihre Berechtigung. Sie werden hauptsächlich für das Layout eingesetzt, da hier keine Semantik notwendig ist.

3.2.4 Textelemente

Nicht nur im Strukturbereich von HTML Dokumenten gibt es neue Elemente, sondern auch im Textbereich. Die Tabelle 3 zeigt neue Auszeichnungen, die in erster Linie für Fließtext gedacht sind.

Tabelle 3: Übersicht der neuen Textelemente

Element	Kurzbeschreibung
<data>	Ergänzt dessen Inhalt um einen maschinen-lesbaren Code.
<time>	Auszeichnung von Datum und Zeit.
<mark>	Textmarkierungen, die mit einem Textmarker vergleichbar sind.
<ruby>	Ruby-Auszeichnungen, die vor allem für ostasiatische Tyopgrafie eingesetzt werden.
<bdi>	Bidirektionale Formatierung. (Beispiel: Arabischer Text innerhalb von deutschsprachigem Inhalt.)
<wbr>	Gelegenheit für einen Zeilenumbruch.

Die beiden Elemente *<time>* und *<mark>* sind von den in der Tabelle 3 genannten Elementen besonders hervorzuheben. Das Listing 6 zeigt ein kleines Anwendungsbeispiel, das den Einsatz von den beiden Elementen veranschaulichen soll.

Listing 6: Beispiel mit <time> und <mark>

```
1  <p>
2    Sie haben nach dem Begriff <mark>Webbrowser</mark> gesucht.
3    Die letzte Suche wurde am
4    <time datetime="2013-04-23 12:15:04">23. April 2013 um 12:15 Uhr</time>
5    durchgeführt.
6  </p>
```

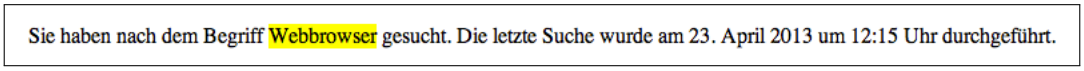

Abbildung 2: Ausgabe im Web-Browser: Beispiel mit <time> und <mark>

Wie man an der Abbildung 2 sieht, wird durch das Auszeichnungselement *<mark>* der Text wie mit einem Textmarker gekennzeichnet[5]. Das Element *<time>* wird zwar in der Ausgabe nicht besonders gekennzeichnet, hat aber eine hohe Bedeutung für automatisierte Anwendungen.

[5]Je nach Implementierung der Web-Browser Hersteller unterscheidet sich die grafische Ausgabe.

Für einen Menschen ist klar, dass im Listing 2 in der Zeile 4 eine Datums- bzw. Zeitangabe steht. Für einen Computer sind es jedoch einzelne Zeichen, die keine Bedeutung haben. Erst durch die Auszeichnung durch das *<time>* Element wird dem Computer bewusst, dass es sich um eine Datums- bzw. Zeitangabe handelt. Weiters ist durch das *datetime* Attribut eine genaue Zeitangabe möglich, auch wenn diese nicht explizit als Text vorhanden ist. Somit können auch ungenaue Zeitangaben wie „heute Vormittag“ mit genauen Zeitpunkten versehen werden. [W3sk; W3sn]

3.2.5 Eingebettete Inhalte

Tabelle 4: Übersicht der neuen Elemente für eingebettete Inhalte

Element	Kurzbeschreibung
<figure>	Einbindung von Illustrationen jeglicher Art.
<figcaption>	Arbeitet mit dem *<figure>* Element zusammen und stellt die Funktion einer Beschriftung bereit.
<embed>	Einbindungspunkt für externe Ressourcen. (Beispiel: Plug-ins)
<video>	Einbinden von Video Dateien.
<audio>	Einbinden von Audio Dateien.
<source>	Alternative Medienressourcen für Video oder Audio Dateien. (Beispiel: Unterschiedliche Video- bzw. Audio-Formate)
<track>	Zusätzliche Medienspuren in Video oder Audio Dateien (Beispiel: Untertitel)
<canvas>	Bitmap-Bereich, der mittels Skriptsprachen dynamisch gefüllt werden kann.
<svg>	Eingebundene Vektorgrafik.
<math>	Eingebundene mathematische Formel.

Die Einbindung von Bildern in HTML Dokumenten ist schon seit 1993 möglich. Mit der Zeit änderten sich jedoch die Anforderungen, die Benutzer an Webseiten stellen. Vor allem die Verwendung von multimedialen Inhalten innerhalb von Webseiten wurde schnell populär. Nicht zuletzt durch die Veröffentlichung der Website *YouTube*, veränderte sich das *World Wide Web* von einem vorwiegend textbasierten Informationspool, hin zu einem Unterhaltungsmedium.

In den Spezifikationen vor HTML5 war die Einbindung und Steuerung von Medieninhalten schwer möglich. Es musste eine einfache Lösung gefunden werden, die es ermöglicht, Video- oder Audio-Dateien einzubinden und ebenfalls über den Web-Browser abzuspielen. Lange Jahre galten proprietäre Plug-ins als die einzige Möglichkeit, Medien-

inhalte im Web-Browser einzubinden. Populäre Plug-ins sind beispielsweise *Adobe Flash Player* oder *Microsoft Silverlight*.

Mit der HTML5 Spezifikation wurden einige Neuerungen im Bereich der eingebetteten Inhalte eingeführt, die es ermöglichen, ohne die oben genannten Plug-ins, Medieninhalte einzubinden und zu steuern.

Listing 7: Beispiel mit <figure> und <figcaption>

```
1 <figure>
2   <img src="tux.png" alt="Tux">
3   <figcaption>Linux-Maskottchen (Name: Tux)</figcaption>
4 </figure>
```

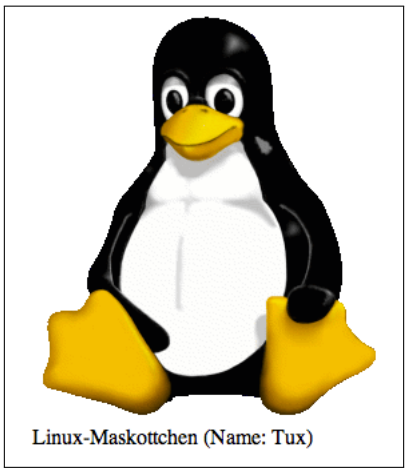

Abbildung 3: Ausgabe im Web-Browser: Beispiel mit <figure> und <figcaption>

Das *<figure>* bzw. *<figcaption>* Element ermöglicht die Einbindung eines Bildes mit einer dazugehörigen Bildunterschrift. Vor HTML5 war es zwar auch möglich, ein Bild mittels verschachtelter * * Elemente zu beschriften, jedoch wohnt dieser Methode die gleiche Problematik inne, die im Abschnitt 3.2.2 beschrieben wurde. [W3sf]

Das Listing 7 bzw. die Abbildung 3 zeigen den Einsatz der neuen Elemente *<figure>* und *<figcaption>*.

Zwei weitere Elemente zur Einbindung von Medieninhalten sind das *<video>* und das *<audio>* Element. Sie zielen vor allem darauf ab, die proprietären Web-Browser Plug-ins zu ersetzen. Mit diesen beiden Elementen ist es möglich, wie es der Name schon vermuten lässt, eine Audio- bzw. Video-Datei in eine Website einzubinden und zu steuern. [W3sg; W3sh]

Listing 8: Beispiel von <audio>

```
1 <audio src="beispielmusik.mp3" controls>
2   <source src="horse.ogg" type="audio/ogg">
3   <source src="beispielmusik.mp3" type="audio/mpeg">
4   Ihr Browser unterstützt das Audio-Tag nicht.
5 </audio>
```

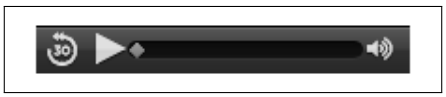

Abbildung 4: Ausgabe im Web-Browser: Beispiel mit <audio>

Das Listing 8 bzw. die Abbildung 4 zeigen, wie eine Audio-Datei eingebunden werden kann. Mittels des *controls* Attributs, ist es möglich, Elemente anzuzeigen, die die Steuerung der Audio-Datei ermöglichen. Innerhalb des *<audio>* Elements können, durch die Angabe des *<source>* Elements, mehrere Audio-Dateien angegeben werden. Vorteil dieser Auszeichnung ist es, dass unterschiedliche Formate angegeben werden können. Je nachdem welcher Audio-Codec beim Endbenutzer verfügbar ist, wird die dazugehörige Audio-Datei geladen.

Das *<video>* Element ist dem *<audio>* Element sehr ähnlich. Als Beispielanwendung des *<video>* Elements dient das Listing 9 bzw. die dazugehörige Abbildung 5. Auch hier ist die Angabe von verschiedenen Video-Dateien durch das *<source>* Element möglich.

Wird das *<audio>*, oder das *<video>* Element vom Web-Browser nicht unterstützt, wird nur der Text, der innerhalb des Tags angegeben ist, angezeigt.

Listing 9: Beispiel mit <video>

```
1 <video width="320" height="240" controls>
2   <source src="beispielvideo.mp4" type="video/mp4">
3   <source src="beispielvideo.ogg" type="video/ogg">
4   Ihr Browser unterstützt das Video-Tag nicht.
5 </video>
```

Ein weiteres, im Bereich von Browserspielen weit verbreitetes Element, ist das *<canvas>* Element. Es definiert einen Bereich, der mittels einer Skriptsprache wie *JavaScript* dynamisch mit Grafiken gefüllt werden kann.

Abbildung 5: Ausgabe im Web-Browser: Beispiel mit <video>

Es ist anzumerken, dass allein mit dem *<canvas>* Element noch keine Grafiken generiert werden können [W3ss]. In der HTML5 Spezifikation ist zwar eine API definiert, die eine Schnittstelle zum *<canvas>* Element darstellt, der Inhalt des Elements muss jedoch erst mit *JavaScript* befüllt werden.

Das Listing 10 und die dazugehörige Abbildung 6 zeigen ein kleines Beispiel, wie das *<canvas>* Element eingesetzt werden könnte.

Listing 10: Beispiel mit <canvas>

```
1  <html>
2    <head>
3      <style type="text/css">
4        #canvaselement{
5          border: 5px solid #AAAAAA;
6        }
7      </style>
8      <script>
9        function draw() {
10         var canvas = document.getElementById("canvaselement");
11         var context = canvas.getContext("2d");
12         context.fillStyle = "rgb(200,0,0)";
13         context.fillRect(10, 10, 50, 50);
14         context.fillStyle = "rgb(0,200,0)";
15         context.fillRect(60, 60, 50, 50);
16         context.fillStyle = "rgb(0,0,200)";
17         context.fillRect(110, 110, 50, 50);
18       }
19     </script>
20   </head>
21   <body onload="draw();">
22     <canvas id="canvaselement" width="170" height="170"></canvas>
23   </body>
24 </html>
```

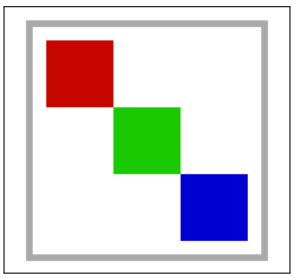

Abbildung 6: Ausgabe im Web-Browser: Beispiel mit <canvas>

3.2.6 Formularelemente

In den 2010er Jahren wurde das *Buzzword* (deutsch: „Schlagwort“) *Web 2.0* sehr populär. Vor der Jahrtausendwende galt das *World Wide Web* als Informationspool, der vorwiegend zum Auffinden textbasierter Informationen genutzt wurde. Mittlerweile rückt die Interaktion zwischen dem Endbenutzer und der Website immer mehr in den Mittelpunkt. Um den gesteigerten Benutzeranforderungen gerecht zu werden, muss die Funktionalität der Web-Formulare weiter ausgebaut werden. [Sie13]

In der HTML5 Spezifikation finden sich einige Änderungen bzw. Neuerungen, die sich mit der Auszeichnung von Webformularen beschäftigen. Die Tabelle 5 zeigt eine kleine Übersicht der neuen Elemente im Bereich der Web-Formulare.

Tabelle 5: Übersicht der neuen Formularelemente

Element	**Kurzbeschreibung**
<datalist>	Enthält eine Liste von *<option>* Elementen.
<keygen>	Erleichtert die Erzeugung von *public-keys* bzw. *private-keys* in Zusammenhang mit kryptographischen Verfahren.
<output>	Ergebnis einer Berechnung.
<progress>	Eine Fortschrittsanzeige.
<meter>	Messskala von diversen Daten.

Ein interessantes Element, das jedoch nicht von allen Web-Browsern unterstützt wird, ist das *<datalist>* Element. Es wird grundsätzlich in Verbindung mit dem *<input>* Element verwendet, um eine Art Auswahlliste aus mehreren vorgegebenen Werten anzuzeigen. Durch die *<option>* Elemente innerhalb des *<datalist>* Elements werden die einzelnen, zur Verfügung stehenden Listeneinträge definiert.

Das Listing 11 bzw. die Abbildung 7 zeigen eine Beispielanwendung einer Kombination aus *<datalist>* und *<option>* Elementen. [W3sd]

Listing 11: Beispiel mit <datalist>

```
<input list="colors">

<datalist id="colors">
  <option value="Red">
  <option value="Green">
  <option value="Blue">
  <option value="Yellow">
  <option value="Black">
</datalist>
```

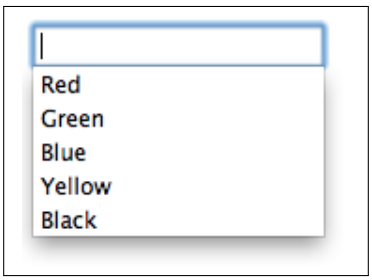

Abbildung 7: Ausgabe im Web-Browser: Beispiel mit <datalist>

Das *<progress>* Element hat eine nicht zu verbergende Verwandtschaft mit dem *<meter>* Element. Beide Elemente kennzeichnen eine Art Fortschrittsanzeige, decken aber unterschiedliche Einsatzgebiete ab. Während das *<progress>* Element vorwiegend für Fortschrittsanzeigen bei Installationen oder Downloads eingesetzt wird, dient das *<meter>* Element der grafischen Darstellung eines Wertes innerhalb eines abgegrenzten Wertebereiches. Das Listing 12 (Abbildung 8) und das Listing 13 (Abbildung 9) zeigen den Einsatz dieser beiden Elemente.

Listing 12: Beispiel mit <progress>

```
<progress value="66" max="100"></progress>
```

Die Konfigurationsmöglichkeiten beim *<meter>* Element sind umfangreicher als jene beim *<progress>* Element. Beim *<meter>* Element ist es möglich, eine obere (Attribut: *max*) und eine untere (Attribut: *low*) Grenze anzugeben. Mit diesen Attributen wird dem Web-Browser signalisiert, dass es sich oberhalb bzw. unterhalb dieser Grenze um einen hohen bzw. niedrigen Wert handelt.

Zwar können die beiden Elemente *<meter>* und *<progress>* ohne weiteres statisch mit Werten gefüllt werden, ihre Stärke spielen sie aber erst in Kombination mit Skriptsprachen wie *JavaScript* aus. [W3sl; W3sm]

Abbildung 8: Ausgabe im Web-Browser: Beispiel mit <progress>

Listing 13: Beispiel mit <meter>

```
1  <p>Ein normaler Wert (65)</p>
2  <meter value="65" min="0" max="100" low="20" high="80"></meter>
3
4  <p>Ein hoher Wert (85)</p>
5  <meter value="85" min="0" max="100" low="20" high="80"></meter>
6
7  <p>Ein niedriger Wert (15)</p>
8  <meter value="15" min="0" max="100" low="20" high="80"></meter>
```

In Zusammenhang mit dem *<datalist>* Element wurde das *<input>* Element erwähnt. Das *<input>* Element kennzeichnet ein Formularfeld, das vom Endbenutzer mit Daten gefüllt werden kann. Das *<input>* Element wird sehr häufig innerhalb von Web-Formularen eingesetzt und deshalb wurden diesem Element im Rahmen der HTML5 Spezifikation einige neue Attribute spendiert. Die Tabelle 6 zeigt alle neuen Attribute, die mit dem *<input>* Element verwendet werden können. Einige der Funktionen, die diese neuen Attribute bieten, waren vor HTML5 nur mittels *JavaScript* möglich. [W3sr]

Doch nicht nur neue Attribute, sondern auch neue Attributwerte für das *type* Attribut wurden eingeführt. Beispielsweise kann angegeben werden, ob es sich bei der Eingabe in das *<input>* Element um eine E-Mail Adresse, oder eine Telefonnummer handelt. Vor der Einführung dieser neuen Attributwerte mussten Webentwickler auf *JavaScript* zurückgreifen, um die Eingaben clientseitig[6] zu prüfen. Durch die neuen Attributwerte wird nun von den Web-Browsern selbst eine einfache Prüfung der Eingabe durchgeführt. Eine umfangreiche serverseitige[7] Überprüfung ersetzt das allerdings nicht.

Im Beispiel-Listing 14 wird ein kleines Formular erstellt, das die neuen Attributwerte verwendet. Wie man in der Ausgabe im Web-Browser sieht, werden einige *<input>* Elemente, die normalerweise nur reine Text-Eingaben erlauben, durch andere Steuerelemente ersetzt. Die Prüfung des E-Mail *<input>* Elements wird erst durchgeführt wenn das Formular abgesendet wird.

[6] Mit *clientseitig* wird jener Endpunkt einer Client-Server Verbindung bezeichnet, von der aus der Endbenutzer eine Website anfordert.
[7] Der Server, von dem ein Endbenutzer eine Website anfordert.

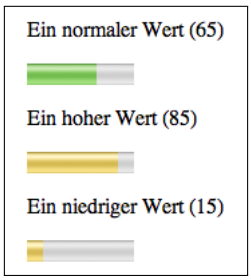

Abbildung 9: Ausgabe im Web-Browser: Beispiel mit <meter>

Tabelle 6: Übersicht der neuen Attribute des <input> Elements

Element	**Kurzbeschreibung**
autocomplete	Der Web-Browser vervollständigt die Eingabe, basierend auf den zuvor getätigten Eingaben. Dieses Attribut kann ebenfalls für das ganze Formular (*<form>* Element) gesetzt werden.
autofocus	Sobald die Website vollständig geladen wurde, wird der Focus auf dieses Element gesetzt.
form	Mit diesem Attribut können mehrere Formulare angegeben werden, zu denen das Element gehört.
formaction	Angabe, welche Datei aufgerufen werden soll, wenn das Formular abgesendet wird.
formenctype	Angabe der Codierung.
formmethod	Angabe, welche HTTP Methode verwendet werden soll.
formnovalidate	Dieses *input* Element soll nicht validiert werden.
formtarget	Spezifiziert, wo die im *formaction* angegebene Datei angezeigt werden soll.
height	Die Höhe des *<input>* Elements.
list	Referenziert eine vordefinierte *Datalist*.
max	Definiert einen maximal möglichen Wert. Wird in Kombination mit den Type-Attributwerten *number* und *date* verwendet.
min	Definiert einen minimal möglichen Wert.
multiple	Kennzeichnet, dass der Endbenutzer mehrere Werte in das *<input>* Element eintragen kann.
pattern	Spezifiziert eine Zeichen-Vorlage, die zur Prüfung der eingegebenen Werte herangezogen wird.
placeholder	Mit diesem Attribut kann ein Platzhalter angezeigt werden.
required	Kennzeichnet ein *<input>* Element, dass vom Endbenutzer ausgefüllt werden muss.
step	Legt in Kombination mit dem Type-Attributwert *number* die möglichen Intervalle fest, die eingegeben werden können.
width	Die Breite des *<input>* Elements.

Tabelle 7: Übersicht der neuen Werte des type Attributs

Element	Kurzbeschreibung
color	Auszeichnung als Farbe. In manchen Web-Browsern wird ein Farbwähler angezeigt.
datetime, date, time, month, week	Auszeichnung diverser Datumsangaben.
email	Auszeichnung als E-Mail.
number	Auszeichnung als Nummer.
pattern	Es kann eine gewisse Zeichen-Vorlage angegeben werden.
range	Auszeichnung als Wertebereich. Manche Web-Browser zeigen einen Schieberegler an.
search	Auszeichnung als Suchfeld.
tel	Auszeichnung als Telefonnummer. Es setzt jedoch keine bestimmte Syntax voraus.
url	Auszeichnung als *URL* (*Uniform Resource Locator*).

Listing 14: Beispiel mit den neuen Werten des type Attributs

```
1 <form action="index.html" method="post">
2   <p>Color: <input type="color"></p>
3   <p>URL: <input type="url" name="homepage"></p>
4   <p>Number: <input type="number"></p>
5   <p>Range: <input type="range" min="1" max="100"></p>
6   <p>E-Mail: <input type="email"></p><br><br>
7   <p><input type="submit" value="send"></p>
8 </form>
```

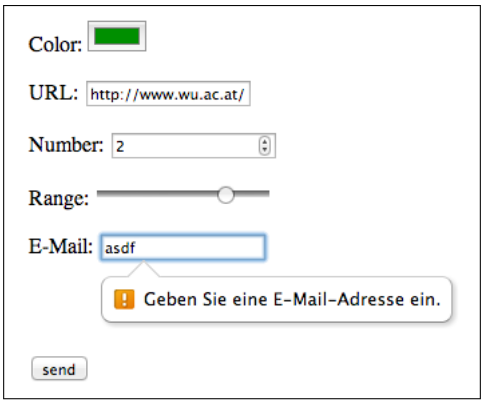

Abbildung 10: Ausgabe im Web-Browser: Beispiel mit den neuen Werten des type Attributs

3.3 APIs

3.3.1 Was sind APIs?

Unter *APIs* (*Application Programming Interface*) versteht man Programmierschnittstellen eines Softwaresystems, die durch andere Software genutzt werden kann, um sich an das jeweilige Softwaresystem anzubinden. Bei Web APIs gibt es eine grundsätzliche Unterscheidung in *client-side* und *server-side* APIs. Eine server-side API bietet eine Schnittstelle an, die es ermöglicht, in die Kommunikation von Client und Server einzugreifen. Client-side APIs hingegen sind in lokalen Web-Browsern angesiedelt und bieten unterschiedliche Möglichkeiten, mittels Skriptsprachen, dynamisches Verhalten zu programmieren. Sie definieren also eine Schnittstelle von HTML, die mit Skriptsprachen angesprochen werden kann. [Wikh]

In den folgenden Abschnitten werden ausgewählte APIs im Bereich der Client-side APIs vorgestellt.

3.3.2 APIs in der W3C Spezifikation

Die **Media API** ist die Programmierschnittstelle, die mit den Medienelementen *<audio>* und *<video>* zusammenarbeitet. Aufgrund dieser API ist es möglich, die Steuerung der Medienelemente dynamisch zu gestalten. Die API bietet Funktionen wie *play()* und *pause()* an, die auch dazu verwendet werden können eigene Steuerelemente zu programmieren. [Ber+12c]

Eine enge Verwandtschaft zu der Media API weist die **Text Track API** auf. Diese API ermöglicht es, Textspuren mit den *<audio>* bzw. *<video>* Elementen zu verbinden. Interessante Anwendungsgebiete erschließen sich in Kombination mit dem *WebVVT* Format (*Web Video Text Tracks*). Das WebVVT Format ist zwar nicht in der W3C Spezifikation enthalten, arbeitet aber mit der Text Track API zusammen. Durch dieses spezielle Format ist es beispielsweise möglich, genaue Angaben zu machen, wann und wie lange ein gewisser Untertitel bei einem eingebundenen Video angezeigt werden soll. [Ber+12e; SP13]

Durch die **Drag and Drop API** ist es möglich, wie der Name schon vermuten lässt, Drag and Drop Funktionen in Web-Browsern zu implementieren. Es ist nicht nur Drag and Drop innerhalb des Web-Browser Fensters möglich, sondern auch die Interaktion mit lokal gespeicherten Dateien. Somit können beispielsweise Dateien, die am Desktop gespeichert sind, einfach auf das Web-Browser Fenster „gezogen“ und mittels einer Skriptsprache weiterverarbeitet werden. [Ber+12a; W3sq]

Die Drag and Drop API lässt somit die Grenzen zwischen dem Web-Browser und dem Betriebssystem verschwimmen. Eine weitere Funktionalität namens **Application Cache** führt diesen Gedanken noch ein Stück weiter. Sie erlaubt, durch die Angabe einer Manifest-Datei[8], das Aufrufen einer Website ohne einer direkten Internetverbindung. Wird eine Website, bei der das Attribut *manifest* innerhalb des *<html>* Elements gesetzt ist, aufgerufen, werden die in der Manifest-Datei angegebenen Dateien offline gesichert. Sollte einmal keine Internetverbindung zur Verfügung stehen, werden beim Aufruf der Website die auf dem Client-System befindlichen Dateien geladen. [Ber+12d; W3sp]

Die gerade vorgestellten APIs zeigen recht deutlich in welche Richtung sich HTML entwickelt. Durch diese APIs sind umfangreiche Web-Applikationen möglich, die auch offline zur Verfügung stehen und sich nahezu wie native Programme steuern lassen. Vor allem im Bereich der Plattformunabhängigkeit bieten diese Technologien gewisse Vorteile. Beispielsweise muss somit nur noch eine Web-Applikation entwickelt werden, die auf allen Geräten, vom PC bis zum Smartphone, funktioniert.

3.3.3 APIs in der WHATWG Spezifikation

Im Abschnitt 3.2.5 wurde schon ein kleines Beispiel gezeigt, wie das *<canvas>* Element mit einem *Canvas 2D Context* eingesetzt wird. Das *<canvas>* Element bietet derzeit die Möglichkeit mit 3 Kontexten zusammenzuarbeiten. [Hic13c]

- **2D**
  Der Kontext *2D* bietet die Möglichkeit 2D Grafiken zu erstellen, oder vorhandene Bilddateien in das *<canvas>* Element zu laden und zu bearbeiten.

- **WebGL**
  WebGL ist die Umsetzung von OpenGL[9] im Browser. Durch diesen Kontext sind hardwarebeschleunigte 3D-Grafiken möglich. Theoretisch können damit umfangreiche Spiele in *canvas* Elementen programmiert werden. In der Praxis wird dieser Kontext derzeit jedoch nur auf experimenteller Ebene eingesetzt. [Wike]

- **Herstellerspezifisch**
  Web-Browser Hersteller können hiermit ihre eigenen Erweiterungen für das *<canvas>* Element einbinden.

[8] Eine einfache Text-Datei, die alle benötigten Dateien der Website auflistet.
[9] OpenGL ist eine plattformunabhängige Grafikbibliothek, die zur Entwicklung von 2D und 3D Grafiken verwendet wird.

Aus Sicherheitsgründen können HTML Dokumente aus unterschiedlichen Domänen nicht aufeinander zugreifen. Dadurch ist es beispielsweise nicht möglich von einem Web-Browserfenster auf ein anderes zuzugreifen. Durch die neu eingeführte Funktion **Cross-document messaging** kann eine Kommunikation zwischen HTML Dokumenten aufgebaut werden, ohne dabei ein Sicherheitsrisiko darzustellen. [Hic13b]

Werden umfangreiche Berechnungen mittels JavaScript ausgeführt, kann das, abhängig vom Client-System, relativ viel Zeit in Anspruch nehmen. Eine neu eingeführte API namens **Web Workers** ermöglicht die Ausführung einer Berechnung, ohne das Browser-Interface zu blockieren. [Hic13e]

HTTP ist grundsätzlich ein zustandsloses Protokoll und jede Anfrage an den Web-Server wird als unabhängig betrachtet. Diese Eigenschaft ist für Web-Applikationen eher hinderlich, da oft über mehrere Seiten hinweg der Besucher identifiziert werden muss. Beispielsweise muss ein bestimmter Besucher in einem Online-Shop von der Artikelauswahl bis zur Bezahlung identifiziert werden. Um das zu bewerkstelligen, wurden vor HTML5 vorwiegend *Cookies* benutzt. Ein Cookie ist eine in Klartext gespeicherte Datei mit Schlüssel-Wert Paaren (englisch: „key-value pairs“), die beim ersten Besuch einer Website vom Web-Server am Client-System gespeichert wird. Sie beinhaltet oft eine client-spezifische Session-ID, die bei jeder Anfrage an den Web-Server mitgesendet wird. Somit ist eine Identifizierung über mehrere Anfragen hinweg möglich. Es gibt jedoch einige Probleme im Zusammenhang mit Cookies, die die Sicherheit, oder auch die mögliche Speichergröße betreffen.

Mit HTML5 wurde eine neue Art von Speicher eingeführt, der Cookies ablösen soll. Unter dem Namen **Web Storage** verbirgt sich eine API, die es ermöglicht, clientseitig Daten zu speichern. Web Storage kann auf zwei Arten eingesetzt werden. Die Daten können entweder persistent (*local storage*), oder nur für die jeweilige Sitzung (*session storage*) gespeichert werden. [Hic13d]

3.3.4 Weitere APIs im HTML5 Universum

Eine viel umworbene Funktion von HTML5 ist die **Geolocation API**. Obwohl die *Geolocation API* in keiner der beiden Hauptspezifikationen (*W3C* und *WHATWG*) von HTML5 enthalten ist, wird sie oft als Beispiel für eine HTML5 Anwendung angeführt.

Geolocation bietet die Möglichkeit, Standortdaten mittels *JavaScript* vom Web-Browser auszulesen und für diverse Web-Applikationen zu verwenden. Vor allem die Anwendung in Zusammenhang mit mobilen Endgeräten bietet eine Vielzahl an Einsatzmöglichkeiten. Unterstützt das Endgerät *GPS* (*Global Positioning System*), ist es mittels dieser API

möglich, sehr genaue Angaben über den Standort einer Person zu manchen. Obwohl diese Funktion durchaus einen Mehrwert für Endbenutzer darstellen kann, sollte die Problematik bezüglich Datenschutz nicht vernachlässigt werden. [W3so]

Mit der **Indexed DB API** ist es möglich, eine clientseitige, umfangreiche Datenbanklösung anzulegen. Zwar können mittels *Web Storage* ebenfalls Daten clientseitig gespeichert werden, jedoch bietet *Indexed DB* einen größeren Umfang, was die mögliche Speichergröße und die Funktionsvielfalt betrifft. [Moza]

Damit Web-Applikationen konkurrenzfähig zu nativen Applikationen sind, ist vor allem der Umgang mit lokal gespeicherten Dateien unumgänglich. Die **File API** bietet Möglichkeiten um lokale Dateien einzulesen und zu verarbeiten. In Verbindung mit der *Drag and Drop API* ist es somit möglich, funktionsreiche Web-Applikationen zu entwickeln. [RS12]

4 HTML5 in der Praxis

4.1 Browserkompatibilität

Im Großen und Ganzen ermöglicht HTML5 die Entwicklung von Web-Applikationen, die schon recht nahe an native Anwendungen heranreichen, aber noch nicht die Performance bieten um mit nativen Anwendungen mitzuhalten. Die diversen neuen APIs, die im Abschnitt 3.3 beschrieben wurden, sind zwar nicht eine vollständige Auflistung aller neuen Funktionen von HTML5, bieten aber einen guten Überblick über die Möglichkeiten, die HTML5 heute (2013) schon bietet.

Grundsätzlich wurde HTML5 unter der Voraussetzung der Abwärtskompatibilität definiert. Aus diesem Grund werden alle Elemente, die in HTML 4.01 bzw. XHTML 1.0 definiert wurden ebenfalls unterstützt. Somit ist eine Migration bzw. Erweiterung von älteren Websites relativ einfach möglich.

Viele Teile von HTML5 sind in den heute aktuellen Browsern (2013) schon implementiert. Eine vollständige Implementierung des gesamten HTML5 Standards[10] ist jedoch in keinem Browser zu finden. Derzeit ist es eher so, dass die unterschiedlichen Web-Browser, unterschiedliche Unterstützungsgrade für HTML5 aufweisen. Zwar sind derzeit noch nicht alle Elemente in allen Browsern verfügbar, setzt man jedoch einzelne Teile von HTML5 ein, sind wesentliche Produktivitätsverbesserungen möglich.

[10] W3C Spezifikation, WHATWG Spezifikation und alle HTML5 Elemente, die dem allgemeinen Sprachgebrauch zuzuordnen sind.

Alle im Anschluss angeführten Web-Browser bzw. ihre Nachfolger bieten eine ausreichende Unterstützung für HTML5. [Pet, S. 43]

- **Firefox 3.0** - Erscheinung: 17. Juni 2008
- **Safari 3.0** - Erscheinung: 11. Juni 2007
- **Opera 10.0** - Erscheinung: 1. September 2009
- **Chrome 3.0** - Erscheinung: 15. September 2009
- **Internet Explorer 9** - Erscheinung: 15. März 2011

Für Webentwickler drängt sich die Frage auf: Was kann zum derzeitigen Zeitpunkt verwendet werden und was nicht? Das Internet bietet eine Fülle an Webseiten, die Übersichten der derzeit unterstützten HTML5 Elemente in aktuellen Web-Browsern anbieten.

Eine erste Anlaufstelle sind die einzelnen Spezifikation selbst. Die WHATWG Spezifikation[11] bietet eine auf den ersten Blick etwas versteckte Informationsquelle zu den einzelnen Web-Browser Implementierungen. Es ist nämlich möglich, innerhalb eines Pop-up Fensters, den aktuellen Implementierungsstand des jeweiligen Elements anzeigen zu lassen.

Abbildung 11 zeigt das Pop-up Fenster des Attributwertes „email“, des *type* Attributs.

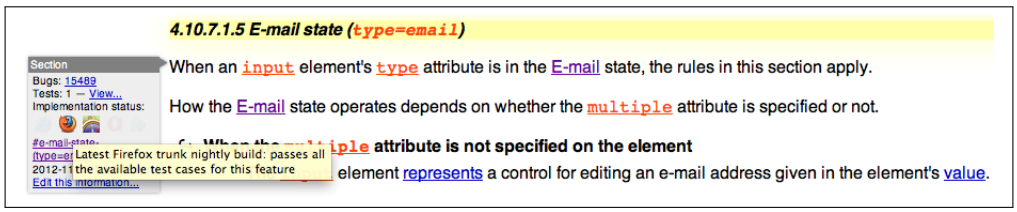

Abbildung 11: WHATWG - Browsersupport

Da in den Spezifikationen relativ umfangreiche Informationen stehen, die oft für Webentwickler nur von geringer Bedeutung sind, bieten einige Websites übersichtlichere Tabellen, die auf einen Blick erkennen lassen, in welchen Web-Browsern das jeweilige Element unterstützt wird. Eine Beispielwebsite wäre *http://caniuse.com*. Die Abbildung 12 zeigt eine Übersicht der Elemente *<progress>* und *<meter>* auf dieser Website.

[11] http://www.whatwg.org/specs/web-apps/current-work/multipage/

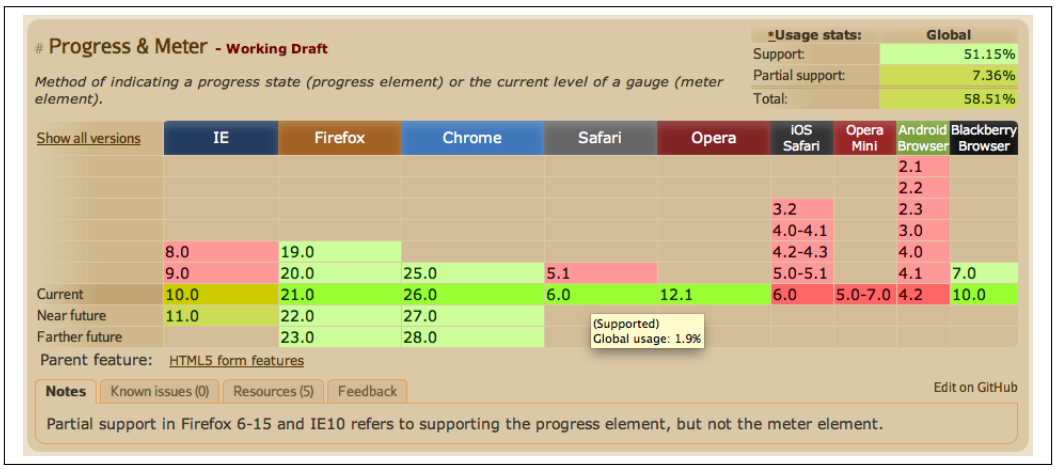

Abbildung 12: http://caniuse.com - Browsersupport (<*progress*> und <*meter*>)

Es ist ebenfalls möglich, die HTML5-Unterstützung des derzeit genutzten Browsers zu ermitteln. Auf der Website *http://html5test.com* ist es möglich, die HTML5-Unterstützung für den aufrufenden Web-Browser aufzulisten. Abbildung 13 zeigt die Ausgabe mit dem Google Chrome Web-Browser (Version: 26) unter Mac OS X 10.8.3.

4.2 HTML5 in Unternehmen

Bei dem Thema HTML5 sind Unternehmen derzeit geteilter Meinung. Nach dem anfänglichen Enthusiasmus für HTML5, äußern sich manche Unternehmen derzeit etwas skeptisch in Hinblick auf den neuen Standard. *Facebook Gründer Mark Zuckerberg* verkündete, dass Facebook für ihre App-Entwicklung nicht mehr HTML5 einsetzen wird, sondern wieder zu der nativen App-Entwicklung zurückkehrt. Der Grund dafür sei die derzeit noch fehlende Marktreife von HTML5.

Ganz anders sieht man dieses Thema bei Mozilla. Die Mozilla Organisation legt ihre ganze Strategie auf HTML5 aus. Nicht nur der Web-Browser Firefox unterstützt einen Großteil der HTML5 Technologien, sondern das neueste Produkt der Softwareschmiede treibt den Gedanken von HTML5 als Applikationsplattform an die Spitze. *Firefox OS* ist ein Smartphone Betriebssystem, das auf Linux und der Gecko Technologie[12] basiert. Besonders hervorzuheben ist, dass die gesamte Benutzeroberfläche auf HTML5 Technologien basiert. Die einzelnen *Apps*[13] basieren ebenfalls auf den HTML5 Web-Technologien, wodurch die Anwendungsentwicklung vor allem für Webentwickler verein-

[12]Gecko ist eine freie HTML-Rendering-Engine, die das HTML Dokument und alle verlinkten Ressourcen einliest und im Web-Browser darstellt.
[13]Kurzbezeichnung für Applikationen.

Abbildung 13: http://html5test.com - Browsersupport

facht wurde.[Mozb; Osl12]

Firefox OS ist ein extremes Beispiel, wie HTML5 eingesetzt werden kann. Die grundsätzliche Frage für ein Unternehmen, das Business-to-Employee-Applikationen erstellt ist, ob die Vorteile der Entwicklung mittels HTML5, die Vorteile von einer nativen Anwendungsentwicklung überwiegen, oder eben nicht. Die Vorteile von nativen Anwendungen sind die perfekte Integration in das Betriebssystem, die einfachere Anbindung von Gerätekomponenten und vor allem die hohe Performance. Der größte Nachteil von nativen Applikationen ist der Aufwand der betrieben werden muss, wenn man die Anwendung für mehrere Betriebssysteme bzw. mehrere Endgeräte entwickeln muss. Genau dieser Nachteil ist zudem der größte Vorteil von HTML5 Applikationen. Die jeweilige Applikation muss nur einmal entwickelt werden und ist anschließend unter diversen Betriebssystemen lauffähig. Der Nachteil von HTML5 Applikationen ist die derzeit noch schlechte Performance.

Als Lösungsansatz könnten *hybride Applikationen* dienen, die die Vorteile beider Welten vereinen. Somit könnten die Applikationen das volle Potential der Geräte ausnutzen und zugleich effizient entwickelt werden. [Bux13]

Das Marktforschungsinstitut *Gartner* prophezeit, dass bis 2015 HTML5 die am weit

verbreitetste Programmiersprache im mobilen Bereich[14] sein wird. Im unternehmerischen Umfeld werden sich vor allem hybride Applikationen durchsetzen. Auch heute schon setzten Unternehmen bei der Integration von ERP-Software im mobilen Bereich auf HTML5 unterstützte Applikationen. [Com]

4.3 Kritik an HTML5

Seit dem Anbeginn von HTML waren Web-Browser Hersteller stark in die Entwicklung des Standards involviert. Wie im Abschnitt 2.1 kurz angeschnitten, haben einzelne Web-Browser Hersteller die Intention, proprietäre Elemente in den jeweiligen Web-Browsern einzubauen, um sich von der Konkurrenz abzugrenzen.

Nachdem sich die Web-Browser Hersteller einen relativ großen Einfluss auf die Entwicklung von HTML erkämpft hatten, hinkte die Spezifikationserstellung des W3C immer hinter den Entwicklungen der Web-Browser Hersteller hinterher. Die Neuerungen werden erst nachdem sie als proprietäre Erweiterungen in den Web-Browsern Einzug gehalten haben in die Spezifikation übernommen. Damit hat diese Art der Spezifikationserstellung einen eher reaktiven Charakter angenommen. Überspitzt ausgedrückt wird die HTML5 Entwicklung von Web-Browser Herstellern diktiert und deshalb liegt die Zukunft des HTML Standards vor allem in deren Händen. Der *Living Standard* der WHATWG hingegen wird ständig aktualisiert und passt demnach besser zu der schnellen Web-Browser Entwicklung. Die durch die Web-Browser Hersteller eingeführten Erweiterungen können somit schneller in den Standard übernommen werden und müssen nicht wie beim W3C relativ lang auf die Verabschiedung warten.

Es gibt jedoch Kritikpunkte an der Vorgehensweise der WHATWG. Die Erstellung der WHATWG Spezifikation liegt in den Händen eines Mannes. *Ian Hickson*, ein Angestellter beim Unternehmen Google, ist hauptsächlich für die WHATWG Spezifikation verantwortlich und steht nur in Kontakt mit einem kleinen Komitee. Dieses Komitee besteht vorwiegend aus Web-Browser Herstellern. Diese Konstellation gewährleistet, in gewisser Art und Weise, die „Alleinherrschaft“ über den HTML5 Standard. Wenn an den Stellschrauben des Webs gedreht wird, betrifft das alle Menschen, die das World Wide Web nutzen und nicht nur ein paar Unternehmen. Deshalb wohnt dieser Vorgehensweise eine gewisse Problematik inne, die nur schwer zu lösen ist. [Pet, S. 452-453]

[14] Vor allem bei Smartphones und Tablets.

5 Zusammenfassung und Ausblick

Der Grundgedanke von *Tim Berners-Lee* war eine globale einheitliche Kommunikationsform zu schaffen, die von allen Menschen genutzt werden kann. Das W3C sollte die Schirmherrschaft über das World Wide Web übernehmen, um einen gemeinsamen Standard zu schaffen, auf dem alle Entwicklungen aufbauen.

Schon seit den Anfangsjahren des World Wide Web hatten die Web-Browser Hersteller einen hohen Einfluss auf den HTML Standard, denn sie sind diejenigen, die die Erneuerungen implementieren. Web-Browser werden oft von Unternehmen entwickelt, die einen wirtschaftlichen Hintergedanken verfolgen. Deshalb wurde in der Vergangenheit immer wieder versucht, proprietäre Erweiterungen jenseits des Standards in den eigenen Web-Browser zu implementieren, um sich von der Konkurrenz abzugrenzen.

Zudem kam der Umstand, dass sich das W3C in der Weiterentwicklung von HTML in eine Richtung bewegte, die von vielen Entwicklern nicht befürwortet wurde und somit kam es zur Abspaltung der WHATWG vom W3C. Ab diesem Zeitpunkt lag die Weiterentwicklung von HTML nicht mehr allein in den Händen des W3C. Derzeit (2013) wird von den beiden Gruppierungen an unterschiedlichen HTML5 Spezifikationen gearbeitet.

Es entstand ein regelrechter Wildwuchs um den neuen Standard HTML5. Nicht nur in der Entwicklergemeinde, sondern vor allem in den diversen Marketingabteilungen werden Techniken oft als HTML5 bezeichnet, die nur entfernt etwas mit der Auszeichnungssprache HTML5 zu tun haben.

Nichts desto trotz ist die Kombination von HTML5, CSS3 und JavaScript derzeit unumgänglich, wenn es um das Thema der Web-Applikationsentwicklung geht. Umfangreiche Web-Applikationen, die nativen Programmen schon recht nahe kommen, werden oft mit diesen drei Technologien umgesetzt.

Das umfangreiche Verständnis von HTML5 und die rasante Entwicklung im World Wide Web erschweren die traditionelle Spezifikationserstellung zunehmend. Das W3C geriet mit ihrer Vorgehensweise ins Hintertreffen und hinkt oft bei Neuerungen hinterher. Der dynamische Prozess, den die WHATWG verfolgt, kann sich wesentlich schneller an Trends im World Wide Web anpassen. Der Gedanke des *Living Standard* geht sogar soweit, dass der Standard von „HTML5“ in „HTML“ umbenannt wurde. Die Frage die sich stellt ist natürlich: Wird es je ein HTML6 geben? Die Zukunft wird es zeigen.

Literatur

[And12] Harvey Anderson. *Windows on ARM Users Need Browser Choice Too*. Mozilla. 9. Mai 2012. URL: http://blog.mozilla.org/blog/2012/05/09/windows-on-arm-users-need-browser-choice-too/ (besucht am 26. 03. 2013).

[Ant12] Gary Anthes. „HTML5 Leads a Web Revolution“. In: *Communications of the acm* 55 (2012).

[Ber+12a] Robin Berjon et al. *W3C Candidate Recommendation - Drag and Drop*. W3C. 17. Dez. 2012. URL: http://www.w3.org/TR/2012/CR-html5-20121217/editing.html#dnd (besucht am 11. 05. 2013).

[Ber+12b] Robin Berjon et al. *W3C Candidate Recommendation - History*. W3C. 17. Dez. 2012. URL: http://www.w3.org/TR/2012/CR-html5-20121217/introduction.html#history-0 (besucht am 17. 04. 2013).

[Ber+12c] Robin Berjon et al. *W3C Candidate Recommendation - Media Elements*. W3C. 17. Dez. 2012. URL: http://www.w3.org/TR/2012/CR-html5-20121217/embedded-content-0.html#media-elements (besucht am 09. 05. 2013).

[Ber+12d] Robin Berjon et al. *W3C Candidate Recommendation - Offline Web Applications*. W3C. 17. Dez. 2012. URL: http://www.w3.org/TR/2012/CR-html5-20121217/browsers.html#offline (besucht am 11. 05. 2013).

[Ber+12e] Robin Berjon et al. *W3C Candidate Recommendation - Text Track API*. W3C. 17. Dez. 2012. URL: http://www.w3.org/TR/html5/embedded-content-0.html#text-track-api (besucht am 09. 05. 2013).

[Ber+13] Robin Berjon et al. *HTML5: Edition for Web Authors*. W3C. 17. Apr. 2013. URL: http://www.w3.org/TR/html5-author/ (besucht am 17. 04. 2013).

[Bux13] Ima Buxton. *Gartner: Hybride Apps setzen sich durch*. 23. Apr. 2013. URL: http://www.computerwoche.de/a/gartner-hybride-apps-setzen-sich-durch,2536657 (besucht am 21. 05. 2013).

[Com] *HTML5 – Die einzige Lösung für eine wirtschaftliche Mobility Strategie in Unternehmen?* commsult Aktiengesellschaft. 18. Juli 2012. URL: http://www.commsult.de/aktuell/html5-apps (besucht am 21. 05. 2013).

[Hic11] Ian Hickson. *HTML is the new HTML5*. WHATWG. 19. Jan. 2011. URL: http://blog.whatwg.org/html-is-the-new-html5 (besucht am 28. 03. 2013).

[Hic13a] Ian Hickson. *HTML: The Living Standard - A technical specification for Web developers*. WHATWG. 17. Apr. 2013. URL: http://developers.whatwg.org/ (besucht am 17. 04. 2013).

[Hic13b] Ian Hickson. *WHATWG - Cross-document messaging*. WHATWG. 5. Mai 2013. URL: http://www.whatwg.org/specs/web-apps/current-work/multipage/web-messaging.html#crossDocumentMessages (besucht am 11. 05. 2013).

[Hic13c] Ian Hickson. *WHATWG - The canvas element*. WHATWG. 5. Mai 2013. URL: http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#the-canvas-element (besucht am 11. 05. 2013).

[Hic13d] Ian Hickson. *WHATWG - Web storage*. WHATWG. 5. Mai 2013. URL: http://www.whatwg.org/specs/web-apps/current-work/multipage/webstorage.html (besucht am 11. 05. 2013).

[Hic13e] Ian Hickson. *WHATWG - Web workers*. 5. Mai 2013. URL: http://www.whatwg.org/specs/web-apps/current-work/multipage/workers.html (besucht am 11. 05. 2013).

[Hic13f] Ian Hickson. *WHATWG Living Standard - Introduction*. WHATWG. 28. März 2013. URL: http://www.whatwg.org/specs/web-apps/current-work/multipage/introduction.html (besucht am 28. 03. 2013).

[Kö11] Peter Körner. *HTML5 im Überblick*. 9. März 2011. URL: http://blogs.technet.com/b/sieben/archive/2011/03/09/gastartikel-html5-im-220-berblick.aspx (besucht am 17. 04. 2013).

[Kö13] Peter Körner. *Die Karte des HTML5-Universums*. 17. Apr. 2013. URL: http://www.peterkroener.de/die-karte-des-html5-universums (besucht am 17. 04. 2013).

[Moza] *Indexed DB - Basic concepts*. Mozilla. 22. Jan. 2013. URL: https://developer.mozilla.org/en-US/docs/IndexedDB/Basic_Concepts_Behind_IndexedDB (besucht am 11. 05. 2013).

[Mozb] *Introduction to Firefox OS*. Mozilla. 15. März 2013. URL: https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS/Introduction (besucht am 21. 05. 2013).

[Osl12] Peter Oslak. *HTML5 spaltet die Netzgemeinschaft - Mark Zuckerberg kritisiert Umstieg*. 31. Okt. 2012. URL: http://www.ictk.ch/content/html5-spaltet-die-netzgemeinschaft-mark-zuckerberg-kritisiert-umstieg (besucht am 21. 05. 2013).

[Pag] *XML, HTML und SGML - eine erste Abgrenzung*. pagina - Publikationstechnologien. 1. Jan. 2004. URL: http://www.pagina-online.de/xml-hintergruende/pagina-das-kompendium/themenkomplex-ii-xml/xml-die-standardisierte-datenfreiheit/xml-html-und-sgml-eine-erste-abgrenzung/ (besucht am 17. 04. 2013).

[Pak08] Ingo Pakalski. *15 Jahre WWW: Die Browserkriege*. 1. Mai 2008. URL: http://www.golem.de/0805/59377.html (besucht am 26. 03. 2013).

[Pet] *HTML5. Websiten innovativ und zukunftssicher*. 2. Aufl. 2011.

[RS12] Arun Ranganathan und Jonas Sicking. *W3C Working Draft - File API*. W3C. 25. Okt. 2012. URL: http://www.w3.org/TR/FileAPI/#introduction (besucht am 11. 05. 2013).

[SP13] Ian Hickson Silvia Pfeiffer. *WebVTT: The Web Video Text Tracks Format*. 15. Apr. 2013. URL: http://dev.w3.org/html5/webvtt/ (besucht am 09. 05. 2013).

[Sela] *Bewegung im W3C*. 30. Okt. 2006. URL: http://blog.selfhtml.org/2006/10/30/bewegung-im-w3c/ (besucht am 26. 03. 2013).

[Selb] *Entstehung des World Wide Web*. 26. März 2013. URL: http://de.selfhtml.org/intro/internet/www.htm (besucht am 26. 03. 2013).

[Selc] *HTML-Varianten*. 24. Apr. 2013. URL: http://de.selfhtml.org/html/referenz/varianten.htm (besucht am 24. 04. 2013).

[Seld] *Unterschiede zwischen XHTML und HTML*. 26. März 2013. URL: http://de.selfhtml.org/html/xhtml/unterschiede.htm (besucht am 26. 03. 2013).

[Sha12] Stephen Shankland. *Google agrees with Mozilla's Windows RT browser concerns*. 10. Mai 2012. URL: http://news.cnet.com/8301-1001_3-57431475-92/google-agrees-with-mozillas-windows-rt-browser-concerns/ (besucht am 26. 03. 2013).

[Sie13] Markus Siepermann. *Web 2.0*. 24. Apr. 2013. URL: http://wirtschaftslexikon.gabler.de/Definition/web-2-0.html (besucht am 24. 04. 2013).

[Sta] *Neue HTML5 Tags und Elemente mit Semantik*. 21. Nov. 2011. URL: http://staticfloat.com/csshtmlxml/neue-html5-tags-und-elemente-mit-semantik/ (besucht am 24. 04. 2013).

[W3c] *XHTML 1.0 The Extensible HyperText Markup Language (Second Edition)*. W3C. 24. Apr. 2013. URL: http://www.w3.org/TR/xhtml1/#h-4.1 (besucht am 24. 04. 2013).

[W3oa] *Facts about W3C*. W3C. 26. März 2013. URL: http://www.w3.org/Consortium/facts (besucht am 26. 03. 2013).

[W3ob] *People of the W3C*. W3C. 26. März 2013. URL: http://www.w3.org/People/ (besucht am 26. 03. 2013).

[W3oc] *W3C Current Members*. W3C. 26. März 2013. URL: http://www.w3.org/Consortium/Member/List (besucht am 26. 03. 2013).

[W3od] *W3C Mission*. W3C. 26. März 2013. URL: http://www.w3.org/Consortium/mission (besucht am 26. 03. 2013).

[W3sa] *HTML Doctype Declaration*. 24. Apr. 2013. URL: http://www.w3schools.com/tags/tag_doctype.asp (besucht am 24. 04. 2013).

[W3sb] *HTML Elements*. 24. Apr. 2013. URL: http://www.w3schools.com/html/html_elements.asp (besucht am 24. 04. 2013).

[W3sc] *HTML class Attribute*. 24. Apr. 2013. URL: http://www.w3schools.com/tags/att_global_class.asp (besucht am 24. 04. 2013).

[W3sd] *HTML datalist Tag*. 24. Apr. 2013. URL: http://www.w3schools.com/tags/tag_datalist.asp (besucht am 24. 04. 2013).

[W3se] *HTML div Tag*. 24. Apr. 2013. URL: http://www.w3schools.com/tags/tag_div.asp (besucht am 24. 04. 2013).

[W3sf] *HTML figure Tag*. 24. Apr. 2013. URL: http://www.w3schools.com/tags/tag_figure.asp (besucht am 24. 04. 2013).

[W3sg] *HTML figure Tag*. 24. Apr. 2013. URL: http://www.w3schools.com/html/html5_video.asp (besucht am 24. 04. 2013).

[W3sh] *HTML figure Tag*. 24. Apr. 2013. URL: http://www.w3schools.com/html/html5_audio.asp (besucht am 24. 04. 2013).

[W3si] *HTML id Attribute*. 24. Apr. 2013. URL: http://www.w3schools.com/tags/att_global_id.asp (besucht am 24. 04. 2013).

[W3sj] *HTML img Tag*. 24. Apr. 2013. URL: http://www.w3schools.com/tags/tag_img.asp (besucht am 24. 04. 2013).

[W3sk] *HTML mark Tag*. 24. Apr. 2013. URL: http://www.w3schools.com/tags/tag_mark.asp (besucht am 24. 04. 2013).

[W3sl] *HTML meter Tag*. 24. Apr. 2013. URL: http://www.w3schools.com/tags/tag_meter.asp (besucht am 24. 04. 2013).

[W3sm] *HTML progress Tag*. 24. Apr. 2013. URL: http://www.w3schools.com/tags/tag_progress.asp (besucht am 24. 04. 2013).

[W3sn] *HTML time Tag*. 24. Apr. 2013. URL: http://www.w3schools.com/tags/tag_time.asp (besucht am 24. 04. 2013).

[W3so] *HTML5 - Geolocation*. 11. Mai 2013. URL: http://www.w3schools.com/html/html5_geolocation.asp (besucht am 11. 05. 2013).

[W3sp] *HTML5 Application Cache*. 11. Mai 2013. URL: http://www.w3schools.com/html/html5_app_cache.asp (besucht am 11. 05. 2013).

[W3sq] *HTML5 Drag and Drop*. 11. Mai 2013. URL: http://www.w3schools.com/html/html5_draganddrop.asp (besucht am 11. 05. 2013).

[W3sr] *HTML5 Form Attributes*. 24. Apr. 2013. URL: http://www.w3schools.com/html/html5_form_attributes.asp (besucht am 24. 04. 2013).

[W3ss] *HTML5 canvas Tag*. 24. Apr. 2013. URL: http://www.w3schools.com/tags/tag_canvas.asp (besucht am 24. 04. 2013).

[Web] *Web-Barometer*. 6. März 2013. URL: http://www.webhits.de/deutsch/index.shtml?webstats.html (besucht am 26. 03. 2013).

[Wika] *Browserkrieg*. 6. März 2013. URL: http://de.wikipedia.org/w/index.php?title=Browserkrieg&oldid=115021773 (besucht am 26. 03. 2013).

[Wikb] *Chrome*. 26. März 2013. URL: http://de.wikipedia.org/w/index.php?title=Google_Chrome&oldid=115919056 (besucht am 26. 03. 2013).

[Wikc] *Extensible Hypertext Markup Language*. 14. Jan. 2013. URL: http://de.wikipedia.org/w/index.php?title=Extensible_Hypertext_Markup_Language&oldid=112941360 (besucht am 26. 03. 2013).

[Wikd] *Hypertext Markup Language*. 16. März 2013. URL: http://de.wikipedia.org/w/index.php?title=Hypertext_Markup_Language&oldid=115459477 (besucht am 18. 03. 2013).

[Wike] *OpenGL*. 25. Apr. 2013. URL: http://de.wikipedia.org/w/index.php?title=OpenGL&oldid=117868782 (besucht am 11. 05. 2013).

[Wikf] *Opera*. 25. März 2013. URL: http://de.wikipedia.org/w/index.php?title=Opera&oldid=115821173 (besucht am 26. 03. 2013).

[Wikg] *Tim Berners-Lee*. 22. März 2013. URL: http://de.wikipedia.org/w/index.php?title=Tim_Berners-Lee&oldid=115711107 (besucht am 26. 03. 2013).

[Wikh] *Web API*. 8. Mai 2013. URL: http://en.wikipedia.org/w/index.php?title=Web_API&oldid=554122592 (besucht am 09. 05. 2013).

[Wiki] *World Wide Web - Browser*. 17. Dez. 2012. URL: http://de.wikipedia.org/w/index.php?title=WorldWideWeb&oldid=111788145 (besucht am 26. 03. 2013).

[Wikj] *World Wide Web Consortium*. 21. Feb. 2013. URL: http://de.wikipedia.org/w/index.php?title=World_Wide_Web_Consortium&oldid=114485874 (besucht am 26. 03. 2013).