MASTER THESIS

zur Erlangung des akademischen Grades
„Master of Science in Engineering“
im Studiengang Multimedia und Softwareentwicklung

HTML5 Canvas vs. Adobe Flash

Ausgeführt von: Peter Kerschner, BSc.
Personenkennzeichen: 1210299029

1. BegutachterIn: Dipl.-Ing. (FH) Arthur Michael Zaczek
2. BegutachterIn: Dipl.-Ing. Mag. Dr. Michael Tesar

Wien, 7. Dezember 2013

FACHHOCHSCHULE
TECHNIKUM WIEN

Eidesstattliche Erklärung

„Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbständig angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher weder in gleicher noch in ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht. Ich versichere, dass die abgegebene Version jener im Uploadtool entspricht."

Ort, Datum

Unterschrift

Kurzfassung

Das HTML5 Canvas-Element und die Skriptsprache JavaScript ermöglichen es, den Browser für Multimedia-Anwendungen und Spiele zu nutzen und die Internet-Entwicklung unabhängig von Flash zu machen. In Zukunft könnte das bedeuten, dass Flash-Anwendungen obsolet werden könnten. Im Zuge der Master Thesis sollen Faktoren wie unter anderem die Möglichkeiten, Performance, Cross-Plattformkompatibilität und mobile Verwendbarkeit von HTML5 mit Flash verglichen werden. Die Analysen sollen anhand praktischer Umsetzungen im Bereich Multimedia- und Spiele-Entwicklung durchgeführt werden. Als Ergebnis soll ersichtlich sein ob das Canvas-Element und JavaScript die Flash-Entwicklung ersetzen kann.

Schlagwörter: Webapplikation, Web-Entwicklung, Web Standards, HTML5, Adobe Flash

Abstract

——-Kurzfassung in Englisch!!——-

Keywords: Webapplication, Web-Development, Web Standards, HTML5, Adobe Flash

Danksagung

———DANKSAGUNG———-

Inhaltsverzeichnis

1. Einleitung

1.1. Problemstellung

Die Entwickler von Flash (früher Macromedia, jetzt Adobe) haben schon früh das Potential von ihrem Flash Player erkannt, um plattformübergreifend Video- und Audioinhalte im Web bereitzustellen. Ein großer Vorteil war, dass Inhalte für Flash lediglich einmal kodiert werden mussten. Tausende von Seiten die sich für Flash als Streamingplattform von Multimediainhalten entschieden haben, bestätigten, bis heute, das Potential.

Mit dem Erscheinen von Apples iPhone und iPod touch im Jahr 2007 und der folgenden Entscheidung Flash auf diesen Geräten nicht zu unterstützen, mussten Webseitenbetreiber darauf reagieren. Viele boten Video/Audio-Streams an, die direkt im mobilen Safari Browser wiedergegeben werden konnten. Durch die Verwendung der H.264 Kodierung konnten die Inhalte auch über den Flash-Player (sofern vom verwendeten Gerät unterstützt) abgespielt werden. Dadurch mussten Inhalte weiterhin nur einmal kodiert werden um mit möglichst vielen Plattformen kompatibel zu sein.

Die Entwickler der HTML5 Spezifikation (und u.a. Apple) sind der Meinung, dass Browser Audio und Video nativ unterstützen sollten ohne dabei auf etwaige Plugins zurückgreifen zu müssen. Dies hat den Vorteil, dass es keine herstellerspezifischen Einschränkungen (wie es bei Flash der Fall ist) für den Entwickler gibt, was er mit den Inhalten anstellt, nachdem sie in die Website eingebettet wurden. Mittels CSS und JavaScript ist es jederzeit direkt möglich dargestellte HTML Elemente zu manipulieren.

Um ein Bild für eine Webanwendung zu erstellen wird üblicherweise eine Grafiksoftware genutzt und anschließend in die Anwendung eingebettet. Für Animationen wird vorwiegend Flash verwendet. Mit dem in HTML5 verfügbaren Canvas Element können Entwickler Bilder und Animationen direkt im Browser mittels JavaScript generieren. Mit Canvas ist es möglich einfache bis komplexe Formen, Graphen und auch Diagramme zu erstellen, ohne auf diverse Bibliotheken, Flash oder ein anderes Plugin zurückgreifen zu müssen.

Mit der Veröffentlichung und der Implementierung von HTML5 Features in die neuesten Brower eröffneten sich neue Möglichkeiten um Animationen, Spiele und Entertainment-Applikationen für das Web zu erstellen. Aufgrund vieler auftretender Fragen ist sich kein Entwickler wirklich klar ob HTML5 den Platzhirschen Flash verdrängen kann:

- Wo liegen die Vorteile und Nachteile beider Technologien?
- Ist HTML5 die Zukunft und löst Flash ab?
- Welche Technologie ist einfacher zu erlernen?
- Was kann HTML5 was Flash nicht kann (und visa-versa)?

- Lohnt es sich noch Flash zu erlernen?
- Gibt es Entwicklungsumgebungen die das Programmieren erleichtern?

1.2. Motivation

———MOTIVATION———-

1.3. Zielsetzung

Das Ziel dieser Arbeit ist es herauszufinden, ob die kommenden Webstandards HTML5 und CSS3, im speziellen das Canvas Element, in Zukunft das proprietäre Webformat Flash samt ActionScript ablösen können. Mittels der Ergebnisse sollen Richtlinien erstellt werden, anhand derer Web-Entwickler leicht ermitteln können, in welchen Fällen die eine Technologie der anderen vorzuziehen ist und aus welchen Gründen.

1.4. Aufbau der Arbeit

Kapitel 2 bietet einen Überblick und eine Einführung in die verwendeten Technologien, die für das Verständnis der Inhalte dieser Arbeit notwendig sind.

Kapitel 3

Kapitel 4

2. Überblick

2.1. Die Geschichte von HTML5

HTML ist die allgegenwärtige Auszeichnungssprache des World Wide Webs. Durch den geschickten Einsatz der paar Elemente (fortan: Tags) die die Sprache unterstützt, konnten erstaunlich viele unterschiedliche Netzwerke von verlinkten Dokumente erstellt werden. Von bekannten Seiten wie Amazon, Ebay und Wikipedia bis hin zu personalisierten Blogs und Webseiten die sich auf Katzenbilder spezialisiert haben. HTML5 ist die aktuellste Version dieser Auszeichnungssprache. Obwohl diese Version die bisher umfangreichsten Änderungen mit sich bringt, ist es nicht die erste Aktualisierung von HTML. Sir Tim Berners-Lee zeichnet sich für die Entwicklung von HTML und damit dem Beginn des Internet verantwortlich. 1991 veröffentlichte er ein Dokument mit dem Titel "HTML Tags", in dem er weniger als zwei Dutzend Elemente, die für das Schreiben von Webseiten genutzt werden konnten, vorschlug. Die Verwendung von Tags (Wörter umgeben von eckigen Klammern: z.B. <html>) war nicht Berners-Lees eigene Errungenschaft. Tags wurden bereits in der SGML (Standard Generalized Markup Language) verwendet. Anstatt einen komplett neuen Standard zu erfinden, erkannte Berners-Lee die Vorteile, bereits existierende Standards weiterzuentwickeln - ein Trend der auch in der Entwicklung der neuen HTML5 Spezifikation erkennbar ist.

2.1.1. Von der IETF zur W3C: Der Weg zu HTML4

Die erste offizielle Spezifikation war HTML 2.0, veröffentlicht durch die IETF, die "Internet Engineering Task Force". Viele der neuen Features dieser Spezifikation basierten dabei auf bereits veröffentlichte Implementierungen. So bot der im Jahr 1994 marktführende Web Browser Mosaic Autoren von Webseiten bereits die Möglichkeit Bilder in Dokumente mittels eines <img>-Tags einzubinden. Das <img>-Tag wurde in die HTML 2.0 Spezifikation inkludiert. Die IETF wurde allmählich durch die W3C, dem World Wide Web Consortium, ersetzt. Folgende Aktualisierungen des HTML Standards wurden auf `http://www.w3.org` veröffentlicht. In der zweiten Hälfte der Neunziger wurde der Standard mehrmals überarbeitet bis 1999 HTML 4.01 veröffentlicht wurde.

2.1.2. XHTML 1: Die Vermischung von HTML und XML

Das nächste Update von HTML 4.01 trug den Namen XHTML 1.0. Das X steht dabei für "eXtensible". Die Spezifikation von XHTML 1.0 war ident zu der von HTML 4.01, somit wurden keine neuen Elemente und Attribute eingeführt. Die Spezifikationen unterschieden sich nur durch die zu verwendende Syntax zum Schreiben von Dokumenten. Während Autoren von HTML Dokumenten kaum Einschränkungen im Schreibstil von Elementen und Attributen hatten, setzte XHTML es voraus, dass die Regeln von XML, einer strikteren Auszeichnungssprache auf der viele Technologien des W3C basierten, eingehalten werden. Aufgrund der strikteren Regeln einigten sich Autoren auf einen gängigen Schreibstil, der auch unter dem HTML 4.01 Standard Verwendung fand. Während Tags früher in Kleinbuchstaben, Großbuchstaben oder einer Mischung aus beidem geschrieben

werden konnten, verlangte ein valides XHTML Dokument, dass alle Elemente und Attribute klein geschrieben werden. Während XHTML 1.0 noch auf HTML basierte und lediglich die strikteren Regeln von XML verwendete, war die XHTML 1.1 Spezifikation reines XML. Dieser Umstand führte zu schwerwiegenden Problemen. XHTML 1.1 Dokumente konnten nicht mehr unter dem Mime-Type *text/html* definiert werden. Der bis dato bekannteste Webbrowser Internet Explorer konnte jedoch Dokumente die mit einem XML Mime-Type publiziert wurden, nicht darstellen.

2.1.3. XHTML 2

Das W3C war mit der vierten Version von HTML der Meinung, dass der HTML basierte Ansatz seinen Zenith erreicht hat und setzten für die zukünftigen Version vollständig auf XML. Trotz der fast identen Namen von XHTML 1 und XHTML 2, konnten die Unterschiede zwischen den beiden Spezifikationen nicht größer sein. Anders als XHTML 1, war XHTML 2 nicht abwärtskompatibel mit bereits existierenden Webinhalten oder vorangegangenen HTML Versionen. Es sollte ein vollkommen neuer Standard werden und es zeigte sich, dass dieser Weg nicht mit Erfolg gekrönt sein würde.

2.1.4. Die Spaltung: WHATWG

Innerhalb des W3C bildete sich eine Gruppierung die gegen die Ansätze rebellierten. Namenhafte Vertreter von Opera, Apple und Mozilla waren mit der eingeschlagenen Entwicklungrichtung des W3Cs nicht zufrieden. Ihnen war es wichtiger mehr Aufmerksamkeit auf Formate zur Entwicklung von Webapplikationen zu richten. Bei einem Workshop in 2004 schlug Ian Hickson, der zu der Zeit bei Opera arbeitete, eine Weiterentwicklung von HTML vor, mit der es möglich sein soll Anwendungen zu entwickeln. Der Vorschlag wurde durch die W3C abgelehnt. Unzufrieden mit der Entscheidung bildeten die Rebellen eine eigene Gruppe: Die Web Hypertext Application Technology Working Group, oder kurz WHATWG.

2.1.5. Der Weg von Web Apps 1.0 zu HTML5

Von Beginn an operierte die WHATWG anders als das W3C. Während das W3C einen kosensorientierten Ansatz anwendet, Themen werden vorgetragen, es wird diskutiert und abgestimmt, wird auch bei der WHATWG diskutiert und abgestimmt, allerdings liegt die finale Entscheidung, was in die Spezifikation komm und was nicht, beim Editor. Der Editor ist Ian Hickson. Der W3C Ansatz klingt demokratisch und fair, allerdings führt dieser auch dazu, dass der Prozess stark verlangsamt wird. Bei der WHATWG hat jeder die Möglichkeit mitzuwirken, da die letzte Entscheidung aber beim Editor liegt, entwickelt sich alles schneller. Schon zu Beginn der Gründung von WHATWG wurden die Projekte in zwei große Spezifikationen aufgeteilt: Web Forms 2.0 und Web Apps 1.0. Beide Spezifikationen sollen die bisherige HTML Spezifikation erweitern. Mit fortschreitender Entwicklung wurden alle Spezifikationen in eine einzige implementiert und in HTML5 umbenannt.

2.1.6. Die Wiedervereinigung

Während HTML5 von der WHATWG weiter entwickelt wurde, setzte das W3C die Entwicklung der XHTML 2 Spezifikation fort. Erst im Oktober 2006, schrieb Sir Tim Berners-Lee einen Blog-Post in dem er zugab, dass der Versuch, das Internet von HTML auf XML zu übertragen, nicht funktionieren wird. Nur wenige Monate später entschied sich die W3C eine neue HTML Working Group zu bilden. Anstatt komplett von Anfang an zu beginnen, nutzen sie glücklicherweise die

bisherige Arbeit der WHATWG als Basis für zukünftige HTML Spezifikationen. Dieses hin- und her führte zu einer schwer überschaubaren Situation. Das W3C arbeitete parallel an den beiden unterschiedlichen, nicht kompatiblen Spezifikationen: XHTML 2 und HTML 5 (mit Leerzeichen). Währenddessen arbeitete die WHATWG, an der Spezifikation für HTML5 (ohne Leerzeichen) auf der die Arbeit der W3C aufbaut.

2.1.7. HTML5: 2012 und 2022

Heute ist der aktuelle Stand der HTML5 Spezifikation nicht mehr so undurchsichtig wie er früher war, allerdings gibt es noch immer offene Fragen.

Es gibt noch immer zwei Gruppen die an HTML5 arbeiten. Die wohl wichtigste Frage für Webentwickler ist „Wann können wir es nutzen?“. In einem Interview gab Ian Hickson an, dass HTML5 frühestens 2022 den “proposed recommendation” Status erreichen wird. Klingt nach einer langen Wartezeit, allerdings bedeutet “proposed recommendation”, dass die HTML5 Spezifikation zwei mal komplett implementiert werden muss. Als Vergleich: HTML 4 existiert nun seit über einem Jahrzehnt und hat noch nicht die gesetzten Features erreicht. Wenn man den Umfang der Spezifikation betrachtet, klingt dieses Datum hoch gesteckt. Browserhersteller sind nach wie vor nicht dafür bekannt, dass existierende Standards so schnell wie möglich implementiert werden. Der Internet Explorer benötigte mehr als ein Jahrzehnt um das *abbr*-Element richtig darstellen zu können.

Für Webentwickler war das Jahr 2012 wesentlich wichtiger. 2012 erreichte die HTML5 Spezifikation den “candidate recommendation” Status, der gleichbedeutend ist mit „fertig und abgeschlossen“. Allerdings reicht das alleine leider auch nicht aus. Wirklich entscheident ist es wann Webbrowser den neuen Standard unterstützen. Schon die Veröffentlichung des CSS 2.1 Standards zeigte, dass man nicht auf die Fertigstellung der Spezifikationen warten sollte, sondern, wenn möglich, die Features nutzen sollte sobald es möglich ist. Das selbe gilt auch für HTML5. Sobald Webbrowser bestimmte Features der Spezifikation unterstützen können diese auch jederzeit verwendet werden. Man darf nicht vergessen, dass HTML5 keine komplett neu entwickelte Sprache ist. Im Sinne der HTML Spezifikation ist es eher eine Evolution als eine Revolution. Da HTML5 auf den früheren Versionen aufsetzt und irgendeine Version des HTML Standards zur Erstellung von Webseiten genutzt wird, wird bereits HTML5 genutzt.

2.2. Rich Media

2.2.1. Audio

MP3 stellt das allgegenwärtige Format zur Kodierung von Audio-Dateien dar. Um sich diese Audio-Dateien auch anhören zu können waren propritäre Technologien notwendig. Damit wurde der Flash Player allgegenwärtig.

Mit HTML5 bietet sich eine neue Technologie an die versucht den Platzhirschen Flash zu verdrängen. Das Einbinden einer Audio-Datei in ein HTML5-Dokument ist äußerst simpel:

```
<audio src="sample.mp3">
</audio>
```

Das Audio Element bietet mehrer Attribute zum Steuern der Audioausgabe an. Das Attribut “autoplay” sorgt dafür, dass die Audio-Datei nach vollständigem Laden sofort gestartet wird. “loop” hingegen, kümmert sich darum, dass das die Audio-Datei in einer Endlosschleife abgespielt wird.

```
<audio src="sample.mp3" autoplay loop>
</audio>
```

Eine Besonderheit dieser Attribute ist, dass sie im Gegensatz zu den herkömmlichen HTML-Attributen, keinen Wert zugewiesen bekommen. Das liegt daran, dass es sich bei diesen Attributen bereits um "Boolean"-Attribute handelt. Das zuweisen eines Wertes, z.B. autoplay="no", wird an der Ausführung des Attributes nichts ändern - entweder die Attribute werden angegeben und ausgeführt oder eben nicht.

Audio API

Das Boolean-Attribut controls sorgt dafür, dass der Browser die nativen Kontrollmöglichkeiten anzeigt und den Usern zu Verfügung stellt. Somit ist es sehr schnell möglich die Audiowiedergabe zu Starten/Stoppen, die Position und die Lautstärke zu verändern. Über JavaScript ist es möglich die Audio API zuzugreifen und somit Methoden für Play, Pause und Eigenschaften wie Lautstärke zu nutzen. Ein einfaches Beispiel mit Button Elementen und Inline Event Handler:

```
<audio id="player" src="sample.mp3">
</audio>
<div>
<button onclick="document.getElementById('player').play()">
Play
</button>
<button onclick="document.getElementById('player').pause()">
Pause
</button>
<button onclick="document.getElementById('player').volume += 0.1">
Volume Up
</button>
<button onclick="document.getElementById('player').volume -= 0.1">
Volume Down
</button>
</div>
```

Audioformate

Obwohl das Audio Element einen sehr guten Eindruck macht gibt es einen Wehrmutstropfen, und dieser liegt nicht in der Spezifikation des Elements. Das Problem ist die breite Fragmentierung von Audiocodecs. Leider gibt es bezüglich der zu verwendeten Codecs unterschiedliche Meinungen unter den Browserherstellern. Während heutzutage das MP3 Format allgegenwärtig ist, ist es nach wie vor kein offenes Format. Die Folge ist, dass MP3 Dateien von Applikationen nur dann dekodiert werden können, wenn für die entsprechenden Patentrechte bezahlt wird. Für Giganten wie Apple oder Adobe stellt das kein gröberes Problem dar, allerdings erschwert es die Arbeit von kleineren Unternehmen und Open-Source Organisationen. Infolgedessen gibt Apples Safari ohne Probleme MP3 Datei wieder während Mozillas Firefox daran scheitert.

Natürlich existieren weitere Audioformate. Der Vobis Codec - auch bekannt unter der Datei-Endung .ogg - ist zum Beispiel nicht patentiert. Firefox unterstützt den Codec aber Safari

nicht.

Glücklicherweise ist es nicht notwendig eine fixe Entscheidung bei der Auswahl des Codecs zu treffen. Anstatt das src Attribut im sich öffnenden <audio> Tag zu nutzen, können mehrere Dateiformate über source Elemente festgelegt werden:

```
<audio controls>
<source src="sample.ogg">
<source src="sample.mp3">
</audio>
```

Ein Browser der Ogg Vorbis Datei wiedergeben kann wird sich für die weiteren source Elemente nicht mehr interessieren. Ein Browser der MP3 wiedergeben kann aber Ogg Vorbis nicht wird einfach das erste Source Element überspringen und die Datei im zweiten wiedergeben. Zusätzliche Hilfe bei der Entscheidung bietet die Deklarierung des entsprechenden Mime Types der Audio Datei:

```
<audio controls>
<source src="sample.ogg" type="audio/ogg">
<source src="sample.mp3" type="audio/mpeg">
</audio>
```

Um die Möglichkeiten vom Audio Element vollständig ausnutzen zu können wird angeraten MP3 und Ogg Vorbis als Codecs zu verwenden.

Fallback Lösungen

Auch wenn das festlegen von mehreren source Elementen sehr nützlich ist, muss bedacht werden, dass Browser existieren die das Audio Element nicht unterstützen. Internet Explorer und Konsorten müssen auf die altmodische Art auf ein Flashumsetzung zurückgreifen. Glücklicherweise unterstützt das Audie Element die Nutzung von Flash. Alles was zwischen den sich öffnenden und schließenden <audio> Tags befindet und kein source Element ist wird nur dem Browser, der das audio Element nicht unterstützt, angezeigt:

```
<audio controls>
<source src="sample.ogg" type="audio/ogg">
<source src="sample.mp3" type="audio/mpeg">
<object type="application/x-shockwave-flash"
<param name="movie" value="player.swf?soundFile=sample.mp3">
</object>
</audio>
```

Das object Element bietet zusätzlich eine Möglichkeit an um Fallback Inhalte anzubieten. So kann zum Beispiel im schlimmsten Fall ein gewöhnlicher Downloadlink angezeigt werden.

```
<audio controls>
<source src="sample.ogg" type="audio/ogg">
<source src="sample.mp3" type="audio/mpeg">
<object type="application/x-shockwave-flash"
<param name="movie" value="player.swf?soundFile=sample.mp3">
```

```
<a href="sample.mp3">Audio Datei herunterladen</a>
</object>
</audio>
```

Mit diesem Code werden bereits vier Fallback Ebenen angeboten:

- Der Browser unterstützt das Audio Element und den Ogg Vorbis Codec.
- Der Browser unterstützt das Audio Element und den MP3 Codec.
- Der Browser unterstützt das Audio Element nicht, hat aber das Flash Plug-in installiert.
- Der Browser unterstützt das Audio Element nicht und hat kein Flash Plug-in installiert.

2.2.2. Video

Mit dem anstieg der verfügbaren Bandbreite stieg auch das Interesse an Video Inhalten an. Das Flash-Plugin ist derzeit noch die erste Wahl wenn Videos im Web angeboten werden sollen. Mit HTML5 könnte sich das ändern.

Das Video-Element funktioniert genauso wie das Audio-Element. Es unterstützt die selben optionalen "autoplay", "loop" und "preload" Attribute. Der Speicherort des Videos kann entweder mittels "src" Attribut im Video-Element oder mittels "source" Elementen, die sich verschachtelt zwischen den sich öffnenden und schließenden <video> Tags befinden, festgelegt werden. Zur Darstellung eines geeigneten User Interfaces kann entweder mittels "controls" Attribut dem der Browser die Darstellung übernehmen lassen oder es wird mit entsprechenden HTML-Elementen, CSS Befehlen und JavaScript ein benutzerdefiniertes erstellt.

Einer der wesentlichen Unterschiede zwischen dem Audio und Video Element ist, dass Videos natürlicherweise einen fixen Bereich der Webseite einnehmen werden. Um diesen Bereich festzulegen müssen im Video Element die entsprechenden Dimensionen definiert werden:

```
<video src="sample.mp4" controls width="360" height="240">
</video
```

Um beim Laden des Videos kann mittels "poster" Attribute dem Browser mitgeteilt werden, währenddessen ein representatives Bild anzuzeigen:

```
<video src="sample.mp4" controls width="360" height="240"
poster="sampleimage.jpg">
</video
```

Der Kampf der rivalisierenden Videoformate stellt sich noch extremer als bei den Audio Formaten dar. Die am stärksten vertretensten Formate sind das patentierte MP4 und das freie Theora Video. Wie beim Audio Element muss das Video in mehreren Kodierungen verfügbar und Notfalls eine Fallback Lösung vorhanden sein.

```
<video controls width="360" height="240"
poster="sampleimage.jpg">
<source src="sample.ogv" type="video/ogg">
<source src="sample.mp4" type="video/mp4">
```

```
<object type="application/x-shockwave-flash" width="360" height="240"
data="player.swf?file=movie.mp4">
<a href="movie.mp4">Video herunterladen</a>
</object>
</video
```

Die Autoren der HTML5 Spezifikation hofften auf eine Standardisierung des Videoformates. Allerdings konnten sich die Browserhersteller, bis heute, nicht auf ein einziges Format einigen.

Native Unterstützung von Videos

Die Fähigkeit Videos nativ in Webseiten einzubinden stellt womöglich eine der aufregensten Erweiterung von HTML dar, seit der Einführung des "img" Elements. Giganten wie Google zögern nicht lange und zeigen bereits ihren Enthusiasmus mit einer auf HTML5 basierenden YouTube-Version: `http://youtube.com/HTML5` Eines der Probleme von Plug-ins zur Darstellung von Webinhalten ist, dass der Inhalt des Plug-ins von dem restlichen Inhalt der Webseite geschützt ist ("sandboxed"). Nativen Rich Media Elementen in HTML haben zur Folge, dass sie ohne Probleme mit den anderen Web-Technologien, CSS und JavaScript, zusammen arbeitet.

Das Video Element ist somit nicht nur programmierbar sondern auch stylebar. Ein Plug-in bietet derartige Möglichkeiten nicht an.

2.2.3. Canvas

Das Canvas Element ist eine Umgebung zur Erstellung von dynamischen Bildern. Das Element ist genauso einfach wie das Audio oder Video Element zu verwenden. Als Attribute werden lediglich Breiten- und Höhenangaben des Canvas angeboten:

```
<canvas id="canvas" width="360" height="240">
</canvas>
```

Alles was sich zwischen den sich öffnenden und schließenden <canvas> Tags befindet, wird nur Browsern angezeigt, die das canvas Element nicht unterstützen.

```
<canvas id="canvas" width="360" height="240">
<p>Canvas Element wird von ihrem Browser nicht unterstuetzt.</p>
</canvas>
```

JavaScript wird verwendet um das Canvas produktiv nutzen zu können. Um mit dem Canvas arbeite zu können muss immer das entsprechende Element über ihre ID ausgewählt und der Kontext festgelegt werden. Kontext bedeutet in diesem Fall welche API genutzt werden soll:

```
var canvas = document.getElementById('canvas');
var context = canvas.getContext('2d');
```

Aktuell stehen nur der 2D und WebGL Kontexts zur Verfügung. Mit der Auswahl des Kontextes ist das Canvas Element bereit um für das Zeichnen von Bildern genutzt zu werden. Die 2D API verfügt über so ziemlich alle Tools die man auch bei einem Grafik Programm wie Illustrator erwartet: Es können Linien, Konturen, gefüllte Flächen, Verläufe, Schatten, Formen und Bézier Kurven gezeichnet werden. Der wesentliche Unterschied besteht allerdings darin, dass kein grafischen User Interface genutzt wird, sondern alles mit JavaScript definiert werden muss.

Mit Code zeichnen

Um die Farbe einer Kontur bzw. einer Linie zu definieren ist folgender Code notwendig:

```
context.strokeStyle = "\#990000";
```

Alles was nun auf dem Canvas gezeichnet wird, hat eine rote Kontur.

Die Syntax für die strokeRect Methode sieht dabei folgendermaßen aus:

```
strokeRect(left, top, width, height);
```

Um nun ein rotes Rechteck zeichnen, das 20 Pixel vom linken Rand und 30 Pixel vom oberen Rand des Canvas entfernt ist und 100 Pixel breit und 50 Pixel hoch ist, ist folgender Code notwendig:

```
context.strokeRect(20, 30, 100, 50);
```

Dabei handelt es sich noch um ein sehr einfaches Beispiel. Die 2D API bietet eine sehr umfangreiche Auswahl an Methoden wie fillStyle, fillRect, lineWidth, shadowColor und viele mehr.

2.3. Verfügbarkeit

Seit der offiziellen Präsentation der ersten Entwürfe der erneuerten Webtechnologie, ist die Begeisterung in der IT Branche groß. Diese Begeisterung ist besonders an dem Enthusiasmus der Web-Giganten Apple, Google und Mozilla und der stetigen Implementierung von HTML5 und CSS3 in ihre Browser erkennbar.

2.3.1. HTML5, CSS3 und JavaScript: Das Web von morgen

Graph

2.3.2. Verfügbarkeit auf Computer

Ob und wie weit die Implementierung der neuen Webfeatures in Browsern fortgeschritten ist, hängt vollständig von den jeweiligen Browserherstellern ab. Um die Verfügbarkeit auf Computern festzustellen, muss der Marktanteil der unterschiedlichen Browser mit einbezogen werden.

IMAGE Browser Jahr

Es ist notwendig die kommenden Trends zu bedenken, da die weitere Implementierung von HTML5 stark von der Popularität des jeweiligen Web-Browsers abhängt. Unbekanntere Browserherstellller werden naturgemäß mehr Zeit für die zur Verfügungstellung von HTML5-Features benötigen als bekannte und beliebte Größen wie Google oder Mozilla.

IMAGE Browser Statistik Monat zu Monat

Die angeführten Statistiken zeigen auf, dass nicht nur der gegenwärtige Prozentsatz an Nutzern sondern auch der Trendverlauf darauf schließen lässt, dass der Internet Explorer tatsächlich nicht mehr zu den marktführenden Browsern zählt, wodurch auch dessen Bedeutung in der Web-Entwicklung schwindet. Aufgrund der sich reduzierenden Nutzeranzahl und dem Umstand, dass der Internet Explorer der Browser mit der am geringsten fortgeschrittensten Implementierung des HTML5 Spezifikation am Markt ist, ermöglicht vielen Web-Entwicklern, schon vor der Fertigstellung der Spezifikation, die Nutzung einiger neuen Features.

Microsofts Marktstrategie ihr eigenes Betriebssystem (Windows) ausschließlich mit dem eigenen Browser auszuliefern ändert kaum etwas an der Statistik. Viele Benutzer installieren daher zum Beispiel Mozilla Firefox oder Google Chrome selbstständig.

2.3.3. Verfügbarkeit auf mobilen Geräten

2.4. Möglichkeiten

Folglich werden einige neue Elemente der HTML5 Spezifikation genauer vorgestellt. Diese wurden auch in der Umsetzung der praktischen Ausarbeitungen verwendet.

2.4.1. Spiele

2.4.2. Animationen

Mit HTML5 Canvas und den CSS3 Neuerungen gibt es für Entwickler neue APIs um Grafiken im Web zu zeichnen und diese auch zu animieren. Die APIs bieten einfache anzuwendente Funktionen an um vordefinierte Formen zu zeichnen, Bilder zu importieren, die Darstellung von bereits gezeichneten Bildern zu verändern oder diese auch zu animieren. Im Gegensatz dazu bietet Flash eine komplette IDE um schnell komplexe Formen zu erstellen, diese anzuzeigen, deren Verhalten zu steuern und zu animieren.

Ein Charakter lässt sich mit einer Zeichenapplikation um ein vielfaches einfacher gestalten als mit mehreren Zeilen Code. Flash bietet eine vollständiges Tool zur Erstellung und Animierung komplexer Grafiken und ist damit wesentlich effizienter zu nutzen, solange es kein equivalentes Tool für HTML5 und CSS3 gibt.

Dieses Kapitel soll aufzeigen, ob HTML5 und CSS3 effizient für die Erstellung von Animationen verwendet werden kann. Anhand bestehender Animationen ... Anschließend werden die Fragen behandelt wie eine Animation mittels Code erstellt wird und ob sich der Zeitaufwand für Entwickler auszahlt. Abschließend wird analysiert ob Webbrowser die Animationen flüssig wiedergeben können.

CSS3 Spiderman

Ein kurzer Animationsfilm der mittels HTML5, CSS3 und jQuery erstellt wurde. Die Animationen sind komplex. So bewegt sich der Hintergrund, es gibt mehrere Szenen, unterschiedliche Betrachtungswinkel, Bewegungen und Gesichtsausdrücke. Der Film beinhaltet auch Musik mittels dem HTML5 Audio Element.

Bisher gibt es viele Experimente die mittels CSS3 umgesetzt wurden. Im Vergleich gibt es nur wenige die auf das Canvas Element und JavaScript setzen.

CSS3 bietet einfache Transformationen (Rotation, Translation, Skalierung) und reichen für simple Animationen. Flash bietet zusätzlich Formtransformationen, als Beispiel die allmähliche Veränderung eines Kreises in ein Quadrat. Bisher ist das mit CSS3 nicht möglich.

CSS3 zeigt im Vergleich zu HTML5 ein größeres Potential um Animationen zu erstellen. Allerdings ist es schwierig mittels Code komplexere Animationen zu erstellen. Solange es kein Tool für HTML5 und CSS3 Animationen existiert wird Flash weiterhin die führende Software in diesem Bereich bleiben.

Performancetechnisch wurde diese Animation ohne jegliche Probleme flüssig in jedem modernen Browser angezeigt.

2.4.3. Entertainment

Audio & Video

Flash ist bis heut noch das bekannteste propritäte Plugin um Audio- und Videoinhalte im Internet zur Verfügung zu stellen. Jedoch bietet nun HTML5 eigene Features die diesem Bereich gewidmet und sind ernst zu nehmende Alternativen zu Flash. Es werden die Audio und Video Spezifikation von HTML5 genauer betrachtet.

Audio

Das Audio Element steht bisher in allen aktuellen Webbrowsern zur Verfügung und ist ähnlich einfach zu nutzen wie das Image Element. Folgender Code zeigt wie ein Audiofile in eine Webseite eingebunden werden kann:

```
<audio src="audio.ogg" controls>
<p>Your browser does not support the audio element.</p>
</audio>
```

Leider gibt es bezüglich der zu verwendeten Codecs unterschiedliche Meinungen unter den Browserherstellern. Um die Möglichkeiten vom Audio Element vollständig ausnutzen zu können wird angeraten MP3 und Ogg Vorbis als Codecs zu verwenden.

Video

Ein Video kann mittels folgendem Code auf einer Webseite angezeigt werden:

```
<video src="movie.ogg" width="640" height="360" controls>
<p>Your browser does not support the video element.</p>
</video>
```

Auch beim Video Element ist die Frage nach dem Codec nicht einfach zu beantworten. Das Video Element kann mit allen modernen Webbrowsern genutzt werden, allerdings gibt es andere Probleme

- Apple Geräte können den Ogg Theora Codec aufgrund von Hardware Problemen nicht wiedergeben
- Opera und Firefox unterstützen den H.264 Codec aufgrund von Lizenz Problemen nicht.

Erst vor kurzem brachte Google eine mögliche Lösung ist Spiel: Die Nutzung von WebM. Dabei handelt es sich um einen lizenzfreie Videocodec. Firefox, Opera, Chrome und auch IE haben bereits bestätigt, dass dieser Codec in Zukunft unterstützt wird. Einige der bekanntesten Video Streaming Anbieter sind bereits dabei HTML5 für Videoinhalte zu nutzen, darunter unter anderem YouTube und Vimeo.

In Kombination mit einem Canvas Element können die Inhalte des Video Elements auf verschiedenste Art manipuliert werden. Unter anderem könnte das Bild in mehere Teile zerschnitten, Explosionen eingefügt und Filter angewendet werden. Allerdings, benötigen komplexe Effekte und Manipulationen eine Hardwarebeschleunigung. Flash nutzt zur effizienten Darstellung von Animationen und Videos die Grafikkarte des Anwenders. Die Möglichkeiten von HTML5 sind noch nicht soweit entwickelt und bietet keine native Möglichkeit um die Hardware des Anwenders zu nutzen.

Um auch die Video Element effizient nutzen zu können, sollten die Quelldateien mit mindestens zwei verschiedene Codecs zur Verfügung gestellt werden.

2.5. Vor- und Nachteile

Jeffrey Zeldman war schon im Jahr 2010 der Meinung, dass HTML, CSS und JavaScript in Zukunft die Entwicklung von Rich Media Applikationen vorantreiben wird. Einige der Vorteile werden hier aufgezählt.

2.5.1. Plattformübergreifende Web-Technologien

Web-Entwickler mussten vor der Umsetzung einer Webanwendung oder Webseite entscheiden für welche Webbrowser entwickelt werden soll. Dieser Umstand entstand aufgrund der ungeeigneten Konzeption früherer HTML und CSS Versionen. Mit den neuen Webstandards (HTML5, CSS3 und JavaScript) ist es nun möglich plattformübergreifende Anwendungen zu erstellen, die nicht nur auf allen Desktopbrowsern sonder auch auf mobilen Browsern genutzt werden können.

2.5.2. Optimiertes Mobile-Web

Bei der Entwicklung von Anwendungen für mobile Geräte spielt die begrenzte Bandbreite des Internets eine entscheidente Rolle. Mittels CSS3 können viele Bilder durch Gradienten, Schatten und andere Effekte ersetzt werden. Überdimensionierte Hintergrund- und Füllbilder werden somit überflüssig und führen zu einem geringerem Verbrauch der zur Verfügung stehenden Bandbreite. Einige HTML5 Features wie Local oder Session Storage erlauben es Daten clientseitig zu speichern und Applikationen auch ohne Internet verfügbar zu machen. Zusätzlich müssen Daten die bereits runtergeladen wurden nicht noch einmal gealden werden.

2.5.3. Rich Media ohne Plugins

Die Audio, Video und Canvas APIs von HTML5 erlauben es Rich Media ohne Verwendung von Plugins im Internet zu nutzen. Propritäte Technologien die in diesem Bereich hauptsächlich verwendet werden haben mit HTML5 einen ernst zu nehmenden Konkurrenten bekommen.

2.5.4. Rich Media Applikationen

Der Großteil der neuen HTML5 Features sind speziell für die Erstellung von Rich Media Applikationen vorgesehen.

2.6. Die Syntax von HTML5

2.7. Rich Media

2.8. Web Forms 2.0

2.9. Semantik

2.10. HTML5 schon heute

3. Mobile Geräte

3.1. Die Entwicklung des mobilen Internets

Die Optimierung von Applikationen und Webseiten für mobile Endgeräte gewinnt immer mehr an Wichtigkeit. Wollte ein Nutzer vor der aktuellen Smartphonegeneration (iPhone und Android) unkompliziert mobil im Internet surfen oder E-Mails schreiben wollte, kam er nicht um ein Blackberry von RIM herum. RIM legte bei ihren Telefon den Hauptfokus auf die E-Mail Funktion. Das Surfen auf einem mobilen Telefon war meist durch zu kleine Bildschirme, langsame Verbindungsgeschwindigkeiten, umständlicher Bedienung, zu hohen Kosten und Einschränkungen in der Darstellung und Nutzung von Webinhalten stark eingeschränkt. Die Einführung der GSM-Erweiterungen GPRS und EDGE sowie die noch schnelleren, auf dem Mobilfunkstandard basierenden UMTS-Datenübertragungsverfahren HSDPA und HSUPA sorgte, insbesondere mit dem Erscheinen des iPhone von Apple, für einen langsamen Umschwung in der mobilen Internetnutzung. Das Bankhaus Morgan Stanley hat im April 2010 eine 87-seitige Präsentation zum Thema „Internet Trends“ veröffentlicht, welche zum einen dem mobilen Internet ein schnelleres Wachstum gegenüber dem desktop-basierten Internet verspricht, und zum anderen, dass im Jahr 2014 dieses in der Nutzung überholen wird.

3.2. iOS von Apple

3.3. Android von Google

4. Adobe Flash

4.1. Was ist Flash?

Während die Popularität von HTML5 immer weiter steigt, werden immer mehr Vergleiche mit „Flash“ durchgeführt. Hierbei handelt es sich um ein Produkt der Firma Adobe, die unter anderem auch für Software wie Photoshop oder Illustrator verantwortlich ist. Allerdings bezieht sich das Wort Flash in den meisten Fällen nicht auf die gleichnamige Entwicklungsumgebung der Firma, sondern der Flash Player, der das Abspielen von proprietären SWF-Dateien im Browser ermöglicht. Anfänglich war Flash ein Programm um Illustrationen und einfache Animationen zu erstellen und wurde 1997 vonder Firma Macromedia, welche im Jahr 2005 von Adobe aufgekauft wurde, entwickelt.

4.2. Die Geschichte von Flash und ActionScript

Seit der Veröffentlichung von Flash in 1996 wurde Flash und ActionScript gemeinsam weiter entwickelt. Mittlerweile stellt die Kombination aus Design- und Animations-Tools in Flash und den interaktiven Möglichkeiten von ActionScript eine der mächtigsten, vielseitigsten und beliebtesten Entwicklungsumgebung dar. Allerdings war ActionScript zu beginn eine sehr bescheidene Programmiersprache.

Die ersten drei Versionen von Flash boten noch keine Programmiertools an und interaktive Abläufe wurden durch Drag-and-Drop Optionen als Actions festgelegt. Lediglich die Navigation durch die Zeitleiste und das Erstellen von Links war über diese Actions möglich.

Flash 4 war die erste Version, die die Möglichkeit bot Code mittels einer einfachen Skriptsprache zu schreiben, welche inoffiziell bereits ActionScript genannt wurde. Mit Flash 5 entwickelte sich ActionScript weiter und wurde zur offiziellen Skriptsprache. Mit jeder folgenden Version von Flash wurden auch die Möglichkeiten von ActionScript erweitert. ActionScript ermöglichte die interaktive Kontrolle von Text, Animation, Sound, Video, Data und mehr. 2003 wurde ActionScript 2.0 vorgestellt, und dessen Möglichkeiten waren schon vergleichbar mit objekt-orientierten Programmiersprachen wie Java oder C#.

Professionelle Programmierer interessierten sich immer mehr für ActionScript als Entwicklungstool, jedoch zeigte sich, dass trotz der vergleichbaren Möglichkeiten ActionScript in der Performance noch nicht mit seiner Konkurrenz mithalten konnte. Der Grund dafür war, dass jede Version von ActionScript auf der vorherigen aufbaute. Der Flash Player war ursprünglich nicht für aufwändige Anwendungen und komplexe Spiele konzipiert, dennoch begannen Entwickler den Flash Player dafür zu nutzen. Damit wurde es klar, dass eine neue Version von ActionScript von Beginn an neu entwickelt werden musste um den Erwartungen gerecht zu werden.

2006 wurde von Adobe ActionScript 3.0 vorgestellt, welches bedeutend mehr neue Funktionalitäten und eine starke Performancesteigerung mit sich brachte. Flash CS3 war die erste Version von Flash die ActionScript 3.0 unterstützte. Flash CS4 fügte neue Funktionen zu ActionScript 3.0 hinzu, unter anderem neue 3D Möglichkeiten, Kontrollfunktionen für Animationen und ActionScript Klassen für Adobe AIR. Flash CS5 und Flash CS6 erweiterten die Funktionalitäten unter anderem mit der Unterstützung von Controller und anderen Geräten, inklusive Multitouch-Funktionen und Touch-Screen Geräten.

4.3. Flash Verfügbarkeit auf Computer

Trotz propritärer Technologien hat Adobe es geschafft, ihren Flash Player auf 99% Prozent aller Systeme unterzubringen. Gründe dafür waren die Fähigkeiten von Flash mit denen Ziele erreicht werden konnten, die mit Hilfe von offenen Standards nicht möglich waren. In den Anfangstagen von Flash wurde es vor allem für Introanimationen, welche auf Startseiten eingesetzt wurden, interaktive Navigationen, die auf die Aktionen des Users mit z.B. Animationen reagierten, oder für die allseits bekannten Werbebanner. Im Gegensatz zu Flash war die Erstellung ähnlicher Funktionen mit standardkonformen Technologien wie HTML 4.01, JavaScript oder CSS nur schwer oder überhaupt nicht realisierbar. Selbst heute werden für komplette Webseiten, Spiele und vor allem Werbebanner mittels Flash realisiert. Ein weiterer Vorteil von Flash ist, dass es so einfach zu erlernen und bedienen ist, das auch Neueinsteiger sehr schnell zu ansehnlichen Ergebnissen kommen.

4.4. Flash Verfügbarkeit auf mobilen Geräten

4.5. Möglichkeiten

4.5.1. Spiele

4.5.2. Animationen

4.5.3. Entertainment

4.6. Die Einflüsse von Flash auf das Web

4.6.1. Die Wurzeln und Entwicklung von Flash

4.6.2. Stärken von Flash

4.6.3. Wichtiger Beiträger für die Entwicklung des Webs

4.6.4. Apples Argumente gegen Flash

Ein oft verwendete Aussage und immer wiederkehrendes Streitthema vieler Diskussionen ist sicherlich, ob HTML5 langfristig Flash als Webentwicklungswerkzeug ablöst. Apple spielt hierbei eine große Rolle. Das Unternehmen verzichtete seit der Einführung ihrer iDevices (iPhone, iPod touch, iPad) auf die Nutzung von Flash. Vor allem die fehlende Möglichkeit Flash auf dem iPad zu verwenden führte vermehrt zu Kritik. Der User sei eingeschränkt und ohne Flash handelt es sich nicht um das „echte Internet“, welches Apple mit ihren Produkten anpreist. Der verstorbene Steve Jobs, CEO von Apple, hat daraufhin einen offenen Brief verfasst, der den Nutzern erklärt,

warum die Firma auf Flash verzichtet hat und auch weiterhin darauf verzichten wird. Steve Jobs Hauptargumente waren und gelten noch bis heute:

1 Flash ist ein 100 Prozent proprietäres System von Adobe, welches trotz der allgemeinen Verfügbarkeit des Flash Players die Zügel in der Hand hat. Adobe könnte ganz alleine über die Richtung entscheiden, in die Flash geht oder die Preisgestaltung ändern. Apple möchte das Internetstandards offen sein sollen, selbst wenn es auf die eigenen Geräte zutrifft.

3 Flashprodukte sind laut einer Analyse des Softwarehauses Symantec mit am anfälligsten für Sicherheitsprobleme. Sie seien einer der Hauptgründe für Abstürze auf dem Mac, auch die Performance auf Mobiltelefonen würde zu wünschen übrig lassen.

4 Flash beansprucht die Akkulaufzeit auf mobilen Geräten, da das Dekodieren von Videomaterial über die Software laufen muss. H.264 würde über die Hardware dekodiert werden.

5 Flash wurde nicht für Touchscreens geschaffen. Viele Webseiten müssten ihre Seite komplett neu konzipieren und entwickeln. Entwickler sollten aber dann doch auf modernere Technologien wie HTML5, CSS3 und Javascript zurückgreifen.

Ein kommerzieller Hersteller wirf nun einem anderen kommerziellen Hersteller vor zu proprietär zu sein - das mag durchaus merkwürdig erscheinen. Vor allem weil die Kunden von Apple auch im eigenen System eingezäunt und der Kontakt zu anderen Systemen über Schnittstellen so gering wie möglich gehalten wird un bei Adobe nur von der theoretischen, aber unwahrscheinlichen Möglichkeite ausgegangen wird, dass diese die Vormachtstellung ihres Browserplugins missbrauchen könnten.

Auch die Performanceprobleme scheinen bei Googles Smartphone Betriebssystem Android seit der Version 2.2 gelöst worden zu sein. Denn seitdem läuft Flash stabil und schnell, ohne den Akku zu sehr zu beanspruchen.

Die Behauptung das Flash nicht für Touchscreen ausgelegt wirden ist, ist an und für sich wahr, das gleiche gilt allerdings auch für HTML. Auch hier gibt es eben so viele mausbasierende Aktionen, wie z.B. Rollover Animationen.

5. HTML5 Canvas und Adobe Flash: Vergleich und mögliche Auswirkungen

5.1. Vergleich von HTML5 Canvas und Adobe Flash

Proprietär vs Standard

5.1.1. Verfügbarkeit

5.1.2. Audio und Video

5.1.3. Animation

5.1.4. Spiele

5.1.5. Werbung

5.1.6. Web-Applikationen

5.2. Mögliche Auswirkungen

5.2.1. Der Wandel von dem allgegenwärtigen Flash zu den neuen Webstandards

5.2.2. Die Zukunft von HTML5 und Flash

5.2.3. Unternehmen und ihre Einstellung zu neuen Ideen

6. Auswirkungen

6.1. Browser

6.1.1. Chrome und Safari (WebKit)

Chrome von Google und Safari von Apple bauen beide auf der HTML-Rendering-Engine WebKit zur Darstellung von Webinhalten auf. Google Chrome konnte zunehmend mit dem bis dato meist genutzen Webbrowser Internet Explorer konkurrieren, bis es nach den Angaben des globalen Statistiksunternehmens StatCounter erstmals im Mai 2012, weltweit die Spitzenposition einnehmen konnte. Mit einem durchschnittlichen Anteil von 35% für Google Chrome und 7% für Apple Safari stellen diese die beliebtesten Webbrowser dar. Im Vergleich decken beide Browser den größten Teil der HTML5 Unterstützung ab.

Die WebKit Engine wurde auf der Grundlage des KDE-Projekts KHTML als Open Source Projekt von Apple entwickelt. Apple ist sehr daran interessiert, dass HTML5 so schnell wie möglich auf allen Geräten verfügbar ist, um den Benutzern ein uneingeschränktes Internet anbieten zu können. Auch die Mobile Safari Variante auf den iDivices basieren auf der WebKit-Engine.

6.1.2. Firefox (Gecko)

Der Open Source Webbrowser Firefox von Mozilla stellt im deutschsprachigen Raum den meist genutzten Webbrowser dar. Weltweit betrachtet befindet sich Firefox mit einem durchschnittlichen Anteil von 23% hinter dem Internet Explorer von Microsoft und Google Chrome. Mozilla zeichnete sich schon früh als Unterstützer des HTML5 Standards ab. So war es schon mit sehr frühen Versionen von Firefox möglich, das Video-Element zu nutzen.

6.1.3. Internet Explorer (Trident)

Der wohl bekannteste Webbrowser Internet Explorer von Microsoft ist bereits in der zehnten Version veröffentlicht worden. Schon mit dem Internet Explorer 9 war Microsoft darauf bedacht die Geschwindigkeit ihres Browsers zu verbessern und es wurde darauf verzichtet, eigene Standards durchzusetzen. Erstmals orientierte man sich gänzlich an den Spezifikationen und unterstütze bereits einige HTML5 Features. So wurde das Audio/Video-Element unterstützt, wobei nur auf proprietäre Codecs wie H.264 und MP3/AAC zurückgegriffen wurde. Seit dem Internet Explorer 10 wird auch der freie Codec WebM unterstützt.

Microsoft hatte sich aus seinem größten Manko einen Vorteil verschafft: Dadurch, das der Internet Explorer nur auf Windows Geräten läuft, kann er die ganze Plattform und Hardware nutzen. Im Gegensatz zur Browsersoftware der Konkurrenz, die auf den kleinsten gemeinsamen Nenner zurückgreifen müssen. So kann Internet Explorers JavaScript Engine Chakra, die Leistung von Multicore Prozessoren ausnutzen.

Mit dem Internet Explorer 10 hat Microsoft einiges aus der Vergangenheit aufgeholt, allerdings sollten diese neuen Features nicht überbewertet werden, denn die Einsatzfähigkeit von HTML5 zeichnet sich weniger durch die neuen Browser, sonder viel mehr durch die Anzahl an veralteten

Browser aus. Solange die alten Versionen des Internet Explorer im Gebrauch sind können viele Neuheiten von HTML5 nicht sinnvoll und ohne Fallbacklösung verwendet werden.

7. Zukunftsaussicht

7.1. Wird HTML5 Flash ersetzen?

Steve Jobs deutet in seinem offenen Brief immer wieder an, dass Flash veraltet und HTML5 die Zunkunft ist. Dies ist insofern falsch, da HTML5 mit seiner bisherigen Browserunterstützung noch zu weit entfernt ist, um mit dem Flash Player mitzuhalten. Unrecht hat Steve Jobs mit der Aussage der fehlenden Offenheit allerdings nicht. Flash soll eingesetzt werden, wenn das Ziel mit einem offenen Standard nicht erreicht werden kann. Mit Flash kann qualitativ hochwertige Leistung erbracht werden. Aufgrund seiner Verbreitung ist es ein inoffizieller Standard.

HTML5 steckt noch in den Kinderschuhen und hat es schwer, mit Flash zu konkurrieren. Denn auch wenn Flash geschlossen ist, ist die Verbreitung des Flashplayer immer noch höher als die Verbreitung von HTML5-fähigen Browsern. Außerdem hat es ein Unternehmen wie Adobe wesentlich einfacher, ein neues Feature in ihren Player einzuführen, als Konsortien wie die W3C oder WHATWG, welche jedes Feature auf ihre Art demokratisch einführen und dann auf Browsersupport hoffen.

Im Videbereich kann sich HTML5 zwar schon präsentieren, doch auch auf YouTube wird der Flash Player weiterhin eine wichtige Rolle spielen, da es noch an wichtigen Funktionen in HTML5 Video fehlt, wie z.B. an einem nativen Vollbild, einem Schutz vor Download zur Wahrung der Urheberrechte oder der Kommunikation mit dem Nutzer. YouTube bietet dem Nutzer die Möglichkeit, direkt über die Webcam ein Video hochzuladen, was ohne Flash nicht möglich wäre.

Entscheidend dafür, was demnächst in HTML5 und was weiterhin in Flash umgesetzt wird, werden sicher die Interessen der Webentwickler sein. Während der Anwender nämlich nur die Technik nutzt und es ihm egak sein wird wie sein Video oder seine Webseite läuft, liegt es an dem Entwickler, ob er überhaupt in relativ funktionsarmen Javascript statt des sehr umfangreichen ActionScript 3 programmiert. Große Unternehmen wie Yahoo, Facebook oder Google könnten ihre Vorteile wiederum aus Webstandards ziehen, da sie selbst eigene Anforderungen an eine neue Spezifikation einbringen können, ohne dass sie von einem kommerziellen Anbieter wie Adobe abhängig sind.

Die Frage, ob HTML5 den Flash Player in Zukunft ablösen wird, bleibt also offen. Flash wird uns sicherlich noch eine Weile erhalten bleiben, da mit HTML5 in seiner jetztigen Form und Verbreitung noch nicht diesselbe Masse erreicht werden kann. Sicher ist auch, dass keines der beiden Programme das andere vollständig ersetzen kann, weshalb man auch nicht von Konkurrenz sprechen sollte, sondern von gegenseitiger Ergänzung.

So eignet sich Flash in Zukunft vo allem für Elemente, die eine hohe Performance benötigen und auf eine umfrangreiche Scriptbibliothek (ActionScript 3) zugreifen können, wie z.B. Spiele, Rich Internet Applications, komplexe 3D-Animationen, Audio/Videoplayer, Simulationen oder umfangreiche Präsentationen mit hohem Audio- und Video-Anteil. Die HTML5/JavaScript

Funktionalitäten eigenen sich hingegen für interaktive Webseitenelemente (Accordions, Tooltips, Dropdown-Menüs, Tabs, uvm), Formularvalidierung, Chats oder einfache Präsentationen.

7.2. HTML6

Die Unterstützung von HTML5 ist noch nicht gewährleistet - hier und da wird an allen Ecken noch gebastelt. Aber trotzdem hat im Januar der Google-Mitarbeiter und das WHATWG-Mitglied Mark Pilgrim im WHATWG-Blog bereits einen Einblick in HTML6 gewährt. Unter anderem wurde ein neues Element mit dem Namen *<device> vorgestellt, welches z.B. Webcam-Konferenzen ohne den Umweg über Flash ermöglichen würde. Aber ob diese Spezifikation auch wirklich „HTML6" heißen wird, ist unklar. Weiter schreibt Pilgrim nämlich:*

> *The next version of HTML doesn't have a name yet. In fact, it may never have a name, because the working group is switching to an unversioned development model. Various parts of the specification will be at varying degrees of stability, as noted in each section. But if all goes according to plan, there will never be One Big Cutoff that is frozen in time and dubbed „HTML6". HTML is an unbroken line stretching back almost two decades, and version numbers are a vestige of an older development model for standards that never really matched reality very well anyway. HTML5 is so last week. Let's talk about what's next.*

Pilgrim sagt also, dass die Vergabe von Versionsnummern veraltet ist und besonders für den Veröffentlichungsprozess von HTML-Standards nicht funktioniert. Ebenfalls interessant ist, dass der HTML5 WHATWG-Entwurf seit diesem Eintrag seinen Namen immer wieder von „WHATWG HTML (including HTML5)" zu „HTML5 (including next generation addition still in development)" ändert.

All dies könnte darauf hindeuten, dass HTML5 wirklich die letzte versionierte HTML-Spezifikation ist. Die Zukunft heißt also nicht „HTML6", sondern einfach nur „HTML" oder „HTML5 mit zusätzlichen Elementen". Das würde die Entwicklung vorantreiben. Es muss nicht mehr ein ganzes Paket an Neuerungen herausgebracht werden, auch einzelne Elemente können eingeführt werden. Schließlich sind es die Browserhersteller, die im Endeffekt HTML5 entwerfen. Dieses könnte eventuell die Existenz der W3C in Zukunft überfküssig machen. Die beiden Spezifikationen sind beinahe identisch und alles, was sich in der W3C-Version findet, steht auch in der WHATWG-Version. Iam Hickson hat allerdings in einer E-Mail kurz erwähnt, dass geplant ist, die WHATWG-Spezifikation in Zukunft ohne engere Absprachen mit dem W3C zu erweitern. Das neue <device> Element ist also nur ein Anfang.

> I've given up trying to keep a WHATWG copy of the HTML5 spec that matches what the W3C publish [...]

Auch Buchautor Peter Kröner prophezeit, dass es eventuell nicht einmal zu einer W3C-Version von HTML5 kommt.

> Die spannende Frage ist, ob es den „One Big Cutoff", sprich den Recommendation-Status für HTML5 zu einem Zeitpunkt, an dem es noch jemanden interessiert, zumindest für HTML5 geben wird. Ich glaube nicht.

Dies begründet Kröner damit, dass die Ausarbeitung von HTML5 seit dem Beginn der Entwicklung im Jahr 2004 bis heute (2013) neun Jahre gedauert hat und inzwischen zum großen Teil implementiert und benutzbar ist. Trotzdem wird aber von einem Recommendation-Status gesprochen, der erst im Jahr 2022 angesetzt wird. Wenn im Jahr 2013 nun aber bereits von einem HTML6 *<device> Element geredet wird, könnte bereits im Jahr 2016 ein Standard namens HTML6 entwickelt und großteilig in den Browsern implementiert sein und infolgedessen den Stand von HTML5 im Jahr 2013 haben. Also ein Datum, was neun Jahre vor dem Recommendation-Status von HTML5 liegt und im Jahr 2022 wahrscheinlich niemanden mehr interessieren wird. Auch aus dieser Sicht scheint ein versionsloses HTML-Modell mehr Sinn zu machen und ist laut Kröner bereits auf dem Weg:*

> *Wenn die WHATWG obendrein plant, mit HTML6 ein neues, versionsloses Entwicklungsmodell einzuführen, betrifft dies bereits HTML5. Die Spezifikationen der WHATWG sind schon jetzt ein Mischmasch aus alten HTML-Features, Neuerungen, die sich auf einen breiten Konsens stützen hoch kontroversen Ideen wie HTML5-Microformats und dem* <device> Element. Der unversionierte Ausbau ist also bereits im Gange."

Literaturverzeichnis

[1] *D. Geary*, Core HTML5 Canvas - Graphics, Animation and Game Development, *1st ed. Prentice Hall, May 2012.*

[2] *S. Fulton and J. Fulton*, HTML5 Canvas, *1st ed. O'Reilly Media, May 2011.*

[3] *R. Hawkes*, Foundation HTML5 Canvas: For Games and Entertainment, *1st ed. friendsofED, April 2011.*

[4] *B. Lamberta and K. Peters*, Foundation HTML5 Animation with JavaScript, *1st ed. friendsofED, November 2011.*

[5] *Makzan*, HTML5 Games Development by Example: Beginner Guide, *1st ed. Packt Publishing, August 2011.*

[6] *S. Koch*, JavaScript: Einführung, Programmierung und Referenz, *6th ed. Dpunkt Verlag, August 2011.*

[7] *T. Hauser, A. Kappler, and C. Wenz*, Das Praxisbuch ActionScript 3: Aktuell zu Adobe Flash CS5, *1st ed. Galileo Design, September 2010.*

[8] *D. Winnie*, Fundamentals of ActionScript 3.0: Develop and Design, *1st ed. Peachpit Press, Juli 2011.*

[9] *E. Feronato*, Flash Game Development by Example, *1st ed. Packt Publishing, Mar 2011.*

[10] *K. Peters*, Foundation ActionScript 3.0 Animation: Making Things Move!, *1st ed. friendsofED, April 2007.*

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

3G	*3rd Generation Mobile Telecommunications*
Ajax	*Asynchronous JavaScript and XML*
AS3	*ActionScript3*
API	*Application Programming Interface (Programmierschnittstelle)*
CSS	*Cascading Style Sheet*
CSS3	*Cascading Style Sheet Level 3*
DOM	*Document Object Model*
HTML	*HyperText Markup Language*
HTML5	*HyperText Markup Language Version 5*
JS	*JavaScript*
JSON	*JavaScript Object Notation*
RIA	*Rich Media Application*
SGML	*Standard Generalized Markup Language*
SQL	*Structured Query Language*
SVG	*Scaleable Vector Graphics*
UA	*User Agent*
URL	*Uniform Resource Locator*
W3C	*World Wide Web Consortium*

A. Überschrift des ersten Anhangs

Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text ...

B. Überschrift des zweiten Anhangs

Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text …