

Seminararbeit

# HTML5

## Überblick und Vergleich

Verfasser:  
Dario Züger  
06-915-025  
Strählgasse 1  
8840 Einsiedeln  
[dario.zueger@unifr.ch](mailto:dario.zueger@unifr.ch)

Betreuung durch:  
Darius Zumstein  
Information System Research Group  
Wirtschafts- und Sozialwissenschaftliche Fakultät  
Universität Fribourg

Fribourg, 17.06.2010

# 1 Abstract

Vor 20 Jahren installierte Tim-Berners Lee am CERN in Genf den allerersten Webserver und entwickelte eine Auszeichnungssprache, die leicht verständlich war und hauptsächlich aus Textelementen und Hyperlink bestand. Sein Ziel war es, dass verschiedene Forschungsinstitute und Universitäten einfach und schnell rund um den Globus Dokumente austauschen konnten. Somit waren das World Wide Web und die Auszeichnungssprache HyperText Markup Language kurz HTML geboren. Berners-Lee gründete daraufhin das World Wide Web Consortium (W3C), welches sich fortan für die Weiterentwicklung und Standardisierung von HTML verantwortlich zeichnete.

20 Jahre später wird das World Wide Web längst nicht mehr nur von Universitäten genutzt, sondern mehrere Millionen Menschen erstellen und konsumieren täglich Informationen zu Hause, am Arbeitsplatz oder mobil unterwegs. Fotos, Multimediainhalte und komplexere Anwendungen wie Spiele, Büroapplikationen oder soziale Kontaktmöglichkeiten ergänzen normale Textelemente und Hyperlinks. Als vor mehr als 10 Jahren der letzte Standard HTML 4.01 und danach XHTML 1.0 verabschiedet wurde, dachte noch niemand an diese Elemente. Das Internet war zumindest für den Heimgebrauch zu langsam, zu teuer und nicht mobil. Die Voraussetzungen änderten sich schnell, nicht jedoch die Auszeichnungssprache für die Webseiten.

Aus diesem Grund gründeten die grossen Browserhersteller – namentlich Microsoft, Mozilla, Opera, Google – und weitere Interessenten wie bspw. Adobe die Web Hypertext Applications Technology Working Group auch WHATWG genannt. Dessen Ziel war es, unabhängig vom manchmal schwerfällig handelnden W3C HTML weiterzuentwickeln und den aktuellen Bedürfnissen anzupassen. HTML5, wie der neue Standard genannt wird, soll dabei vollständig rückwärtskompatibel zu HTML 4.01 und XHTML 1.0 sein und neue Elemente erhalten.

Der Website-Programmierer hat dabei die Wahl, ob er die ursprüngliche HTML- oder die wohlgeformte und strukturierte XHTML-Syntax einsetzen möchte. Beide Sprachen werden vollumfänglich unterstützt. Dies ist ein Kriterium, das die Migration auf HTML5 massgebend erleichtert und höchstwahrscheinlich ebenso fördert.

Damit Webseiten besser strukturiert und für sogenannte Screenreader verständlicher aufgebaut werden können, wurden zusätzliche semantische Tags eingeführt. Ab sofort müssen nicht mehr mehrere DIV-Container nebeneinander oder ineinander geschachtelt definiert werden, sondern man kann beispielsweise ein `<section>` einführen, das einen eigenen Abschnitt auf einer Seite definiert und bei Bedarf einen eigenen Kopfbereich besitzt. Mit dem Tag `<nav>` wird die Hauptnavigation der Website gekennzeichnet. `<article>` kann für Blogposts, Newsartikel oder Kommentare verwendet werden.

Neue Tags für Filme, Audio, Grafiken und kleinere Applikationen sollen weitverbreitete Plugins wie Adobe Flash oder Microsoft Silverlight ersetzen. Die Browserhersteller integrieren benötigte Codecs für die Wiedergabe direkt in ihre Software und machen die Installation von Drittanbieterprogrammen, die unter anderem oft wegen Sicherheitslücken dem grossen Ressourcenverbrauch in die Schlagzeilen geraten, überflüssig.

Verschiedenste Formularfunktionen, wie beispielsweise eine Client-seitige Validierung und neue Textfelder vereinfachen die Programmierung für den Websitehersteller. Bis anhin musste viel Zeit in aufwändige JavaScripts investiert werden, die nun direkt im Browser integriert und durch einfache Tags ersetzt werden.

HTML5 erlaubt viele neue Möglichkeiten und erleichtert die Websiteprogrammierung. Die wichtigsten Neuerungen werden in dieser Arbeit vorgestellt.

## **Stichworte zur Arbeit:**

HTML (HyperText Markup Language), HTML5, XHTML (Extensible HyperText Markup Language), Auszeichnungssprache, W3C (World Wide Web Consortium), WHATWG (Web Hypertext Applications Technology Working Group)

## Inhaltsverzeichnis

1	Abstract .....	1
2	Einleitung.....	6
2.1	Ausgangslage und Motivation .....	6
2.2	Problemstellung und Vorgehensweise .....	6
2.3	Zielsetzung und Output.....	6
3	Neuerungen in HTML5.....	7
3.1	Trennung von Inhalt und Design .....	7
3.2	HTML5 versus XHTML5.....	8
3.3	Das Grundgerüst.....	9
3.4	Eine Übersicht der neuen Tags und Attribute .....	10
4	Neue semantische Tags in HTML5 .....	12
4.1	Einleitung .....	12
4.2	Header und HGroup.....	13
4.3	Footer .....	15
4.4	Navigation .....	15
4.5	Section.....	16
4.6	Article.....	18
4.7	Aside.....	19
4.8	Figure .....	19
4.9	Time.....	20
5	Formulare .....	22
5.1	E-Mail, URL und Suchfelder.....	22
5.2	Zahlenfelder .....	23
5.3	Datum und Uhrzeit .....	24
5.4	Datalist.....	25
5.5	Farbauswahl .....	26
6	Multimediaelemente.....	27
6.1	Canvas.....	27
6.2	Video .....	29
6.3	Audio .....	32
7	Schlussbemerkungen .....	33
8	Literaturverzeichnis.....	35

## Abbildungsverzeichnis

Abbildung 1: Beispiel für den Header einer Seite. Mit CSS formatiert.....	14
Abbildung 2: Beispiel eines Footers für einen Artikel .....	15
Abbildung 3: Beispiel eines Navigations-Blocks.....	16
Abbildung 4: Beispielseite mit div und section-Blöcke .....	17
Abbildung 5: Beispiel zu Aside .....	19
Abbildung 6: Poster mit Überschrift .....	20
Abbildung 7: Beispiele für Fehlermeldungen im Browser Opera .....	22
Abbildung 8: Number und Range mit Output in Opera.....	24
Abbildung 9: Das neue Datumfeld im Browser Opera.....	24
Abbildung 10: Datalist in Opera .....	25
Abbildung 11: Canvas-Grafik.....	28
Abbildung 12: Abspielen eines Filmes in Opera.....	30
Abbildung 13: Abspielen einer Audio-Datei in Opera .....	32

## Tabellen Verzeichnis

Tabelle 1: Neue HTML5 Tags mit den wichtigsten Attributen.....	10
Tabelle 2: Unterstützung der Attribute <code>placeholder</code> , <code>required</code> und <code>autofocus</code> .....	22
Tabelle 3: Unterstützung von E-Mail und URL-Feldern.....	23
Tabelle 4: Unterstützung von Suchfeldern.....	23
Tabelle 5: Unterstützung der neuen Zahlenfelder .....	24
Tabelle 6: Unterstützung des Datumsfeld.....	25
Tabelle 7: Unterstützung der Datalist.....	25
Tabelle 8: Unterstützung der Farbauswahl .....	26
Tabelle 9: Unterstützung von Canvas.....	29
Tabelle 10: Welcher Browser unterstützt welche Video-Codecs .....	31
Tabelle 11: Welcher Browser unterstützt welche Audio-Codecs .....	32

## Listing Verzeichnis

Listing 1: Styling innerhalb und eines ganzen Textabsatzes .....	8
Listing 2: Grundgerüst von XHTML5.....	9
Listing 3: XHTML 1.0 und HTML 4.01 - Doctype.....	9
Listing 4: HTML5-Doctype .....	9
Listing 5: Grundlegende DIV-Container .....	12
Listing 6: Content-Block.....	12
Listing 7: Grundlegende Blöcke in HTML5.....	12
Listing 8: Blockelemente definieren in CSS .....	13
Listing 9: JavaScript-Code für neue Elemente - hier Header - im Internet Explorer .....	13
Listing 10: Beispiel für den Header einer Seite .....	14
Listing 11: Header, HGroup, H1 in Article .....	14
Listing 12: Beispiel eines Footers für einen Artikel.....	15
Listing 13: Navigation .....	16
Listing 14: Beispiel zu section.....	17
Listing 15: Beispiel zu Article .....	18
Listing 16: Beispiel für eine Sidebar.....	19
Listing 17: Poster mit Überschrift.....	20
Listing 18: Publikationsdatum ohne und mit Zeit.....	20
Listing 19: Publikationsdatum mit pubdate.....	20
Listing 20: Textfelder für E-Mail und URL .....	23
Listing 21: Suchfeld .....	23
Listing 22: Zahlenfelder .....	23
Listing 23: Beispiel für Output.....	24
Listing 24: Datumsfeld .....	25
Listing 25: Beispiel Datalist.....	25
Listing 26: Farbauswahl.....	26
Listing 27: Block für Canvas-Grafiken.....	27
Listing 28: JavaScript Beispielcode für Canvas-Block.....	27
Listing 29: Zusätzlicher JavaScript-Code für den Internet Explorer.....	28
Listing 30: Film in Webseite einbinden .....	30
Listing 31: Gleicher Film mit verschiedenen Codecs .....	30
Listing 32: Zusätzlich mit Type-Angaben .....	31
Listing 33: Fallback-Lösung.....	31
Listing 34: Einbetten einer Audio-Datei in die Webseite.....	32
Listing 35: Gleiche Audio-Datei mit verschiedenen Codecs .....	32

## 2 Einleitung

### 2.1 Ausgangslage und Motivation

Im Juni 2004 wurde die WHAT Arbeitsgruppe (Web Hypertext Applications Technology Working Group) bestehend aus Vertretern der Mozilla Foundation, Opera Software, Google, Apple, Microsoft und weiteren Interessenten gebildet. Deren Ziel ist es, die bestehenden HTML Standards der 90er-Jahre weiterzuentwickeln und die Arbeit der Webdesigner zu vereinfachen. Dabei soll die Sprache neue Elemente und Funktionen erhalten, aber auch abwärtskompatibel sein, damit eine einfache Migration für Websiteentwickler gewährleistet ist. Das Resultat der Arbeitsgruppe soll anschließend an das W3C (World Wide Web Consortium), der internationalen Webstandard-Organisation, zur Annahme übermittelt werden.

Im Januar 2008 konnte durch das W3C ein erster Arbeitsentwurf veröffentlicht werden. Der aktuellste Entwurf stammt vom 4. März 2010 und obwohl die Auszeichnungssprache noch nicht finalisiert ist, unterstützen bereits einige Browser viele der neuen Funktionen, während andere noch auf Hilfen durch spezielle JavaScripts angewiesen sind.

Nebst der WHAT Arbeitsgruppe bildete das W3C eine eigene Gruppe, die die Entwicklung von XHTML - der HTML-Standard in XML-Schreibweise - voranbringen sollte. Diese Abteilung wurde jedoch im Oktober 2009 zugunsten von HTML5 aufgelöst.

### 2.2 Problemstellung und Vorgehensweise

Da HTML5 beim W3C noch den Status „Working Draft“ hat und wahrscheinlich noch viel Zeit vergehen, bis es finalisiert und standardisiert wird, wurde bis jetzt noch keine Literatur darüber publiziert.

Aus diesem Grund muss mit vorhandenen Quellen aus dem Internet gearbeitet werden. Die Schwierigkeit ist, vertrauenswürdige und zuverlässige Quellen auszumachen, die ihre Beschreibungen stets dem aktuellen Arbeitsstand der WHAT Gruppe anpassen.

Sehr erwähnenswert ist die Seite Dive into HTML5<sup>1</sup> von Mark Pilgrim. Der Autor beschreibt auf seiner Homepage sehr ausführlich die Funktionen der neuen Tags, die in seinem Buch gedruckt werden.

Als weitere Quelle steht die Website der WHAT Arbeitsgruppe<sup>2</sup> zur Verfügung, auf der täglich neue Spezifikationen und informative Beispiele veröffentlicht werden.

Im ersten Schritt werden die Unterschiede von HTML5 und XHTML5 erklärt, wie der Schnitt von Inhalt und Design vollzogen wurde und wie die Grundstruktur einer Website aussieht.

Der zweite Teil erklärt neue semantische Tags von HTML5. Der dritte Teil konzentriert sich auf Formulare, die viele neue nützliche Funktionen erhalten, während der letzte Teil den Abschluss mit den neuen Multimediafunktionen für Grafiken und Audio/Video macht.

### 2.3 Zielsetzung und Output

Ziel der Arbeit ist es, einen Überblick über die neuen Möglichkeiten und Tags von HTML5 zu erhalten und einen Vergleich zu den früheren Auszeichnungssprachen HTML 4.01 und XHTML 1 zu schaffen.

Dabei soll auch ersichtlich werden, wie HTML5 aktuell von den heutigen Browsern wie dem Internet Explorer 8, Firefox 3.64, Opera 10.54, Google Chrome 5 und Safari 4 unterstützt und welche Massnahmen die Websiteentwickler selber noch treffen müssen bis die Sprache endgültig standardisiert sein wird.

Nebst dieser Arbeit wird eine HTML5-konforme Website mit Beispielen zu den neuen Tags erstellt, mit der die Browserkompatibilitäten überprüft und Umgehungsmassnahmen getestet werden können.

Diese Arbeit stellt keine Einführung in HTML allgemein dar, sondern beschreibt nur Neuerungen im Zusammenhang mit dem neuen Standard. Es wird vorausgesetzt, dass der Leser mit HTML, XHTML, CSS und weiteren Internettechnologien vertraut ist.

---

<sup>1</sup> <http://diveintohtml5.org>

<sup>2</sup> <http://www.whatwg.org>

## 3 Neuerungen in HTML5

### 3.1 Trennung von Inhalt und Design

Die Hypertext Markup Language kurz HTML wurde ursprünglich von Tim Berners-Lee 1990 entwickelt, damit Wissenschaftler und Universitäten auf eine einfache Art und Weise untereinander Entwicklungsfortschritte und Dokumente austauschen konnten. Damals dachte noch niemand an Multimediaelemente wie Filme oder Bilder, sondern die Sprache enthielt nur Elemente für Überschriften, Textabsätze, Listen, Tabellen und natürlich Hyperlinks, um mehrere Seiten untereinander verlinken zu können.

*„HTML is the language for describing the structure of Web pages. [...] With HTML, authors describe the structure of pages using markup. The elements of the language label pieces of content such as “paragraph,” “list,” “table,” and so on.” [W3C 2010]*

Wie die Definition des W3C sagt, ist HTML eine Auszeichnungssprache, die die Grundstruktur einer Webseite beschreibt. Mitte der 90er Jahre implementierten die beiden grossen Browserhersteller Netscape und Microsoft selbstständig neue Elemente und Attribute, die es erlaubten, die ursprünglichen Tags zu formatieren und gestalten, was nicht dem ursprünglichen Sinn von HTML entsprach. Es entstanden Tags wie `<font>`, `<center>`, `<strike>`, `<u>` oder Attribute wie `border`, `align`, `cellpadding`. Das W3C standardisierte die neuen Implementierungen in HTML 3.2 im Jahr 1997, damit sie mit Netscape und dem Internet Explorer Schritt halten konnten. Damit wurde die anfängliche Definition der Auszeichnungssprache ungültig.

Die Websitebesitzer begannen nun innerhalb von HTML, Elemente zu formatieren und Designs für ihre Webseiten zu erstellen, nur interpretierte jeder Browser die Seiten auf seine Weise. Die Websites waren in unterschiedlichen Browsern nicht kompatibel zueinander, was die Arbeit erschwerte. Weiter verzeihen Browser viele Fehler, die in den Code einprogrammiert werden und versuchen ihn trotzdem irgendwie darzustellen.

1999 erlaubte HTML 4.0 als erste Version Cascading Style Sheets (CSS), die verwendet werden, um Webseiten unabhängig von HTML zu formatieren.

*“CSS is the language for describing the presentation of Web pages, including colors, layout, and fonts. [...]The separation of HTML from CSS makes it easier to maintain sites, share style sheets across pages, and tailor pages to different environments. This is referred to as the separation of structure (or: content) from presentation.” [W3C 2010]*

CSS vereinfachte die Gestaltung massgeblich, da man das Aussehen einer ganzen Website durch das Bearbeiten dieser Datei beeinflussen konnte, ohne dass der HTML-Code berührt wurde. Ein grosser Schritt in Richtung der Trennung von Struktur bzw. Inhalt und Design war gemacht, aber noch nicht vollständig vollzogen, weil HTML 4.0 weiterhin die Styling-Elemente, die mit der Version 3.2 eingeführt wurden, erlaubte.

HTML5 macht nun den endgültigen Schnitt. Folgende Tags wurden aus diesem Grund oder weil eventuell noch passendere Elemente bestehen, **nicht** in die Spezifikation übernommen:

- Schriftformatierung: `<font>`
- Grosse Schreibweise: `<big>`
- Ausrichtung des Textes: `<center>`
- Durchgestrichener Text: `<s>`
- Durchgestrichener Text: `<strike>`
- Unterstrichener Text: `<u>`
- Kursiver Text: `<tt>`
- Vorformatierter Text: `<xmp>`

Auch Styling-Attribute wie `border`, `cellpadding`, `cellspacing`, `width` in einer Tabelle sind nicht mehr erlaubt und müssen durch eine CSS-Formatierung ersetzt werden.

Trotz dieser Änderungen können Tags, die es erlauben, Texte kursiv oder fett zu schreiben, weiterhin eingesetzt werden. Es sind dies die Elemente `<i>`, `<em>`, `<b>` und `<strong>`. Sie

dürfen jedoch nur einen Teil und nicht einen ganzen Absatz hervorheben. Sobald ein ganzer Textabsatz z.B. in einem `<p>`-Tag kursiv geschrieben sein soll, muss CSS eingesetzt werden.

Neu wurde der Tag `<mark>` eingeführt, mit dem Textabschnitte oder einzelne Worte wie mit einem Leuchtstift auf einem Blatt Papier hervorgehoben werden können. Dieses Element findet beispielsweise Anwendung in einer Suchmaschine, indem die Suchworte in der Resultatsliste gelb markiert werden.

#### Listing 1: Styling innerhalb und eines ganzen Textabsatzes

```
<p>In diesem Text wird nur <b>dieser Teil</b> fett geschrieben</p>
<p class="bold">Der ganze Textabsatz kann in einem CSS-Style fett formatiert werden.</p>
```

## 3.2 HTML5 versus XHTML5

*„XHTML is a variant of HTML that uses the syntax of XML, the Extensible Markup Language. XHTML has all the same elements (for paragraphs, etc.) as the HTML variant, but the syntax is slightly different. [...]“* [W3C 2010]

1999 veröffentlichte das W3C die neue Markup-Language eXtensible Hypertext Markup Language kurz XHTML 1.0, die eine Reformulierung des 1997 veröffentlichten HTML 4.01 ist und auf XML (eXtensible Markup Language) basiert. Sie wurde nicht mit neuen Elementen ergänzt, sondern ausschliesslich XML-konform umgeschrieben. Es wurden beispielsweise folgende Änderungen beschlossen [Wöhr 2004]:

- Neuer Doctype
- Alle Elemente benötigen nebst dem Start- auch ein End-Tag
- Ein empty-Tag ist sowohl ein Start- als auch End-Tag zugleich. Aus diesem Grund benötigt es zusätzlich einen Leerschlag und einen Schrägstrich am Schluss. Dies betrifft zum Beispiel `<br />`, `<img />` und `<hr />`
- Jeder Attributwert wird von Anführungszeichen umschlossen. Zum Beispiel `id="key1"`
- Jedes Attribut besitzt einen Wert. Sogenannte Schalterattribute sind nicht erlaubt. Zum Beispiel `preload="preload"`
- Alle Tags werden klein geschrieben. Grossschreibweise ist nicht erlaubt
- Elemente dürfen sich nicht überschneiden, sondern müssen sauber geschachtelt werden.

Ziel war es, dass fortan wohlgeformte Seiten geschrieben und in allen Browsern korrekt und gleichermassen dargestellt werden (*siehe auch Kapitel 3.1*).

XHTML hatte stets einen schweren Stand, da alle Browser Fehler immer verzeihen und versuchten, auch nicht konforme Seiten auf eine möglichst korrekte Weise darzustellen. Aus diesem Grund und weil die Entwicklung von HTML5 durch die WHAT Arbeitsgruppe bereits weiter fortgeschritten war, wurden im Oktober 2009 die Arbeiten am Nachfolger XHTML 2.0 eingestellt.

Um die Migration auf HTML5 zu erleichtern, erlaubt die neue Auszeichnungssprache die Schreibweisen von HTML 4.0 und XHTML gleichzeitig. Der Autor kann demnach zum Beispiel den Schrägstrich in einem empty-Tag oder auch die Anführungszeichen für Attributwerte einsetzen oder auslassen. Es wird stets korrekt interpretiert.

Möchte man seinen Code nur in XHTML5 verfassen und diesen auch entsprechend durch die Browser interpretieren lassen, muss der `html`-Tag eine XML-Deklaration und die Seite den Typ `„application/xhtml+xml“` enthalten. Wird der Typ `„text/html“` verwendet, wird die Seite als HTML5 interpretiert. Die XML-Deklaration hat zur Folge, dass eine Webseite bereits bei einem Fehler im Code nicht dargestellt werden kann.

Der Unterschied zwischen HTML5 und XHTML5 ist nur die Schreibweise des Codes. Sie besitzen beide die gleichen Tags und Attribute.

**Listing 2: Grundgerüst von XHTML5**

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/2000/xmlns">
<head>
  <meta content="application/xhtml+xml">
  <title>Titel der Seite</title>
</head>
<body>...</body>
</html>
```

In dieser Arbeit werden fortan alle Code-Beispiele in XHTML5 ohne XML-Deklaration geschrieben mit der Ausnahme, dass Schalterattribute wie `preload`, `autostart`, `novalidate` und weitere erlaubt werden.

### 3.3 Das Grundgerüst

Microsoft führte vor mehr als zehn Jahren den Doctype - zu Deutsch Dokumententyp - für HTML-Seiten ein. Als sie den Internet Explorer 5 entwickelten, fiel ihnen auf, dass ältere Webseiten nicht mehr korrekt angezeigt wurden, weil sie mehr auf Standards achteten als in früheren Versionen. Eigentlich war nicht das Problem, dass sie nicht mehr korrekt angezeigt wurden, sondern die Seiten wurden so falsch dargestellt, wie sie geschrieben waren. Für den Internet Explorer 4 und Netscape war dies kein Problem, da sie nicht auf die Standards des World Wide Web Konsortiums setzten.

Aus diesem Grund wurde vor dem Wurzelement `<html>` der Doctype eingeführt, der dem Browser mitteilte, um welche Auszeichnungssprache es sich handelte und für welchen Browser sie gedacht war. Es gibt unzählig viele verschiedene Versionen des Doctypes. Folgend ein paar Beispiele:

**Listing 3: XHTML 1.0 und HTML 4.01 - Doctype**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

HTML5 verkürzt den Dokumententyp massiv und erlaubt nur noch eine Version:

**Listing 4: HTML5-Doctype**

```
<!DOCTYPE html>
```

Schon kann mit der Programmierung einer HTML5-Webseite begonnen werden.

### 3.4 Eine Übersicht der neuen Tags und Attribute

In der Tabelle 1 werden alle neuen HTML5-Tags mit ihren zugehörigen Attributen aufgelistet und kurz erläutert. Ebenfalls werden Kapitel erwähnt, die das Element weiter ausführen.

**Tabelle 1: Neue HTML5 Tags mit den wichtigsten Attributen**

Tag	Attribute	Beschreibung
<!DOCTYPE HTML>		Deklariert, in welcher HTML Version die Seite geschrieben ist. (siehe Kapitel 3.3)
<article>		Für News-, Blog- oder Forumsbeiträge. Ersetzt <div class="article"> (siehe Kapitel 4.6)
<aside>		Ein Bereich neben einem Artikel mit zusätzlichen Informationen oder weiterführenden Links. Ersetzt <div class="sidebar"> (siehe Kapitel 4.7)
<command>	label type	Definiert einen Radiobutton, Checkbox oder einfach eine Schaltfläche. Muss zwischen dem <menu> Tag sein.
<details>	open	Beschreibt Details eines Dokumentes.
<figcaption>		Überschrift für <figure> (siehe Kapitel 4.8)
<figure>		Gruppiert verschiedene Elemente und gibt ihnen eine Überschrift. Diese Elemente sind elementar für das Dokument und können nicht ausgelassen werden (siehe Kapitel 4.8)
<header>		Kopfbereich einer Seite oder eines Bereiches mit Überschrift, Logo etc. Ersetzt <div id="header"> (siehe Kapitel 4.2)
<hgroup>		Wird verwendet, um mehrere Überschriften zu gruppieren. Kann auch als Kopfbereich verwendet werden, wenn dieser nur Überschriften enthält (siehe Kapitel 4.2)
<mark>		Markierter Text, der speziell hervorgehoben werden soll. Vergleichbar mit einem Leuchtstift (siehe Kapitel 3.1)
<meter>	max, min	Wird für Messangaben verwendet, deren Minimum und Maximum bekannt sind.
<nav>		Hauptnavigationsbereich der Homepage. Ersetzt <div id="navigation"> (siehe Kapitel 4.4)
<progress>	max value	Gibt den Fertigstellungsgrad einer Aufgabe an. Muss mit JavaScript gesteuert werden.
<rp>		Gibt dem Browser an, was er zeigen soll, falls er <ruby> nicht unterstützt.
<rt>		Erklärt die Aussprache eines asiatischen Zeichens, das in <ruby> verwendet wird.
<ruby>		Wird verwendet, um eine Ausspracheanleitung für Asiatische Zeichen zu geben.
<section>		Definiert einen Bereich – meistens mit einem eigenen Header und Footer – einer Seite und enthält verschiedene Elemente. Ersetzt <div id="section"> (siehe Kapitel 4.5)
<summary>		Zusammenfassung von <detail>
<time>	datetime	Wird verwendet, um eine Zeitangabe (z.B. Veröffentlichungsdatum eines Artikels) maschinenlesbar zu machen. Ersetzt <p class="pubdate"> (siehe Kapitel 4.9)
<audio>	autoplay controls preload src	Für Musik, die direkt im Browser ohne Plug-Ins abgespielt werden kann (siehe Kapitel 6.3)
<canvas>	height width id	Container für Grafik, die mit JavaScript programmiert wird (siehe Kapitel 6.1)

<embed>	height src width	Für Nicht-HTML-Elemente wie Adobe Flash, um einen Film abzuspielen. Es kann auch <object> eingesetzt werden, das in diesem Fall die gleiche Bedeutung besitzt ( <i>siehe Kapitel 6.2</i> )
<source>	src type	Definiert Video- und Audioquellen für <video> und <audio>. Wird nur verwendet, wenn mehrere Dateien zur Verfügung stehen ( <i>siehe Kapitel 6.2</i> )
<video>	autoplay controls height preload src width	Für Filme, die direkt im Browser ohne Plug-Ins abgespielt werden ( <i>siehe Kapitel 6.2</i> )
<datalist>	id	Definiert eine Auswahlliste für ein Textfeld, die aber auch Eingaben durch den User erlauben ( <i>siehe Kapitel 5.4</i> )
<keygen>	keytype name	Generiert automatisch einen Schlüssel für das Formular.
<output>	for name	Generiert live ein Resultat aus verschiedenen Formularzellen ( <i>siehe Kapitel 5.2</i> )

## 4 Neue semantische Tags in HTML5

### 4.1 Einleitung

Um mehrere Elemente zu gruppieren oder einen eigenen Bereich wie z.B. einen Header, Footer, Navigation oder Artikel zu erstellen, generiert der Webdesigner `<div>`-Container. Mit den Attributen `class` oder `id` kann er sie eindeutig identifizieren, mit CSS formatieren und ausserhalb des Webseitenverlaufes positionieren. Div hat selber keine spezielle Bedeutung. Aus diesem Grund sollte es nie Text ohne weitere Gruppierungen wie beispielsweise den Tag `<p>` enthalten.

Wird eine Homepage erstellt, besteht sie meistens aus folgenden Bereichen:

#### Listing 5: Grundlegende DIV-Container

```
<div id="header"></div>
<div id="nav"></div>
<div id="content"></div>
<div id="footer"></div>
```

Für die folgenden Unterkapitel soll nun angenommen werden, dass der Container mit der ID "content" zuerst eine Einleitung/Begrüssung und danach mehrere Newsbeiträge – immer mit dem neusten Eintrag an erster Stelle – enthält. Jeder dieser Artikel stellt einen weiteren Container dar. Das würde zu folgendem Code führen:

#### Listing 6: Content-Block

```
<div id="content"></div>
  <div id="introduction"></div>
  <div id="news"></div>
    <div class="article">Artikel 1</div>
    <div class="article">Artikel 2</div>
  </div>
</div>
```

HTML5 definiert neue Sektionen, die diese DIV-Container zum Teil - aber nicht vollständig - ersetzen. Der normale User wird keinen Unterschied feststellen, weil auch sie erst durch CSS-Styles ein Design erhalten. Aber gerade für Personen mit einer Behinderung und Suchmaschinen entstehen dadurch viele Vorteile, auf die jeweils in den einzelnen Unterkapiteln eingegangen wird.

Der Code oben wird in HTML5 beispielsweise neu wie folgt geschrieben:

#### Listing 7: Grundlegende Blöcke in HTML5

```
<header>...</header>
<nav>...</nav>
<div id="content">
  <section id="introduction">...</section>
  <section id="news">
    <article>Artikel 1</article>
    <article>Artikel 2</article>
  </section>
</div>
<footer>...</footer>
```

Obwohl viele Hersteller ihre Browser noch nicht mit den neuen Elementen ausgestattet haben, können sie bereits verwendet werden. Die Tags werden trotzdem erkannt wenn auch nur als Inline-Elemente. Dies kann man umgehen, indem man sie im CSS als Blockelement definiert.

**Listing 8: Blockelemente definieren in CSS**

```
header, nav, section, article, footer {
  display:block;
}
```

Wie bereits erwähnt, werden die Elemente durch die modernen Browsern unterstützt. Der Internet Explorer stellt eine Ausnahme dar. Er besitzt zwar die gleichen Funktionen wie seine Konkurrenten, interpretiert aber bereits seit einem Jahrzehnt den HTML-Code auf eine eigene Art und Weise, die nicht dem Standard des World Wide Web Konsortiums entspricht. Viele Webseiten werden dadurch anders korrekt dargestellt, obwohl ihr Code Fehler aufweist. Dies stellte auch viele Programmierer immer wieder vor Probleme und sie mussten spezielle Lösungen nur für den Internet Explorer einsetzen.

Microsoft verspricht mit der neuen Version 9, die spätestens Ende 2010 veröffentlicht werden soll, alle Websprachen nur noch standardisiert und HTML5 vollumfänglich zu unterstützen. Dadurch tauchen jedoch neue Probleme auf. Was passiert mit den Webseiten, die spezifisch für den Microsoft Browser angepasst wurden oder fehlerhaften Code aufweisen? Solche Seiten werden nicht mehr korrekt angezeigt. Ein zweites Problem ist, dass die User - vor allem Unternehmen - nur sehr träge auf neue Programmversionen migrieren. Laut aktuellen Statistiken von [WebHits 2010] verwenden 63% der Webbenutzer den Internet Explorer, davon 38.7% die betagte Version 6 aus dem Jahr 2001. Erst 7.7% haben auf IE8 gewechselt. Eine sehr geringe Zahl, wenn man bedenkt, wie viele Sicherheitslücken der alte Browser enthält und Standards nicht beachtet. Das dritte und letzte Problem ist, dass der neue Internet Explorer nur noch auf den neuesten Betriebssystemen Windows Vista und 7 installiert werden kann. Windows XP User haben das Nachsehen. Laut [WebHits 2010] ist Windows XP immer noch auf ca. 65% aller Computer weltweit installiert. Die Nachfolger Vista und 7 kommen nur auf je ca. 5%, was sich nun jedoch ändern dürfte.

Damit auch der aktuelle Internet Explorer die neuen Elemente erkennt, muss nebst der CSS-Eigenschaft (*siehe Listing 8*) für jeden Tag ein kurzer JavaScript-Code im `<head>` Bereich erstellt werden (*siehe Listing 9*). Erst dann werden sie korrekt interpretiert und können mit CSS zusätzlich formatiert werden.

Bis jeder User in den Genuss von HTML5 kommt – sprich ein aktuelles Betriebssystem und einen aktuellen Browser einsetzt –, werden wahrscheinlich noch Jahre vergehen.

**Listing 9: JavaScript-Code für neue Elemente - hier Header - im Internet Explorer**

```
<script>
  Document.createElement("header");
</script>
```

## 4.2 Header und HGroup

„The header element represents a group of introductory or navigational aids.“ [WHATWG 2010]

Der Tag `<header>` – nicht zu verwechseln mit `<head>` – kennzeichnet den Kopfbereich einer Webseite oder einer Sektion und enthält normalerweise die Überschriften `<h1>` bis `<h6>` und weitere Elemente wie Logos, Suchfelder für die Website oder selbst die Navigation (*siehe Kapitel 4.4*) kann integriert werden.

Header ersetzt Container wie `<div id="header"></div>` und darf auf einer Seite mehrere Male eingesetzt werden. Es soll angenommen werden, dass eine Seite mit mehreren Newsartikeln (*siehe Kapitel 4.6*) erstellt werden soll. Jeder Artikel hat eine Überschrift und das Veröffentlichungsdatum. Diese beiden Elemente werden in einen Header verpackt.

Analog kann der Tag `<section>` (*siehe Kapitel 4.5*) einen Kopfbereich haben, muss aber nicht.

**Listing 10: Beispiel für den Header einer Seite**

```
<header>
  
  <hgroup>
    <h1>HTML5 - &Uuml;berblick und Vergleich</h1>
    <h2>Seminararbeit von Dario Züger</h2>
  </hgroup>
</header>
```



**Abbildung 1: Beispiel für den Header einer Seite. Mit CSS formatiert.**

„The *hgroup* element represents the heading of a section. The element is used to group a set of *h1*–*h6* elements when the heading has multiple levels, such as subheadings, alternative titles, or taglines.” [WHATWG 2010]

`<hgroup>` kann als Spezialfall eines Headers bezeichnet werden. Das Element dient dazu, die Tags `<h1>` bis `<h6>` zu gruppieren, falls mindestens zwei aufeinander folgend eingesetzt werden. Es darf keine anderen Elemente enthalten. Angenommen der Newsartikel, der weiter oben beschrieben wurde, enthält nur zwei Überschriften aber kein Veröffentlichungsdatum, dann wird ausschliesslich `<hgroup>` als Header verwendet und nicht `<header>`.

Wird der Artikel-Kopfbereich sogar nur auf eine Überschrift `<h1>` reduziert, wird weder `<hgroup>` noch `<header>` eingesetzt.

Listing 11 fasst alle drei Möglichkeiten zusammen:

**Listing 11: Header, HGroup, H1 in Article**

```
<!-- Eine Überschrift und Veröffentlichungsdatum in <header> -->
<article>
  <header>
    <h1>&Uuml;berschrift des Artikels</h1>
    <p><time datetime="2010-05-01">01. Mai 2010</time></p>
  </header>
  ...
</article>

<!-- Zwei Überschriften gruppiert mit <hgroup> und Veröffentlichungsdatum
in <header> -->
<article>
  <header>
    <hgroup>
      <h1>&Uuml;berschrift des Artikels</h1>
      <h2>2. &Uuml;berschrift des Artikels</h2>
    </hgroup>
    <p><time datetime="2010-05-01">01. Mai 2010</time></p>
  </header>
  ...
</article>

<!-- Zwei Überschriften gruppiert mit <hgroup> als Header -->
<article>
  <hgroup>
    <h1>&Uuml;berschrift des Artikels</h1>
    <h2>2. &Uuml;berschrift des Artikels</h2>
  </hgroup>
```

```

...
</article>

<!-- Nur eine Überschrift -->
<article>
  <h1>&Uuml;berschrift des Artikels</h1>
  ...
</article>

```

In HTML5 wird das Problem gelöst, dass bisher zu wenig verschiedene Überschriften zur Verfügung standen. Jede Überschrift der Stufe 1 bis 6 durfte auf einer Webseite nur einmal eingesetzt werden. Neu darf jede Überschrift einmal pro Sektion verwendet werden. Das heisst zum Beispiel, dass der Header der Seite, die Navigation und jeder Artikel `<h1>` enthält, während bis anhin für jedes Element eine andere Stufe benutzt wurde.

### 4.3 Footer

„The footer element represents a footer for its nearest ancestor sectioning content or sectioning root element. A footer typically contains information about its section such as who wrote it, links to related documents, copyright data, and the like.“ [WHATWG 2010]

Wie der Name Footer schon besagt, handelt es sich bei diesem Tag um die Fusszeile für einen Artikel, eine Sektion oder gar für die HTML-Seite. Listing 12 und Abbildung 2 zeigen, dass ein Footer Informationen über den Autor und weiterführende Links enthalten kann. Er ist nicht auf eine Zeile beschränkt. Viele Website-Betreiber fügen meistens neben den Copyright-Informationen ganze Linklisten zu Partner- oder Kontaktseiten ein. `<footer>` ersetzt DIV-Container wie `<div id="footer"></div>`.

#### Listing 12: Beispiel eines Footers für einen Artikel

```

<footer>
  <p class="author">Autor: Hans Muster | 0 Kommentare</p>
  <p class="more"><a href="#">&raquo; Weiterlesen</a></p>
</footer>

```

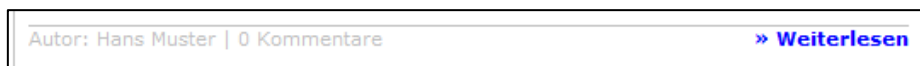


Abbildung 2: Beispiel eines Footers für einen Artikel

### 4.4 Navigation

„The nav element represents a section of a page that links to other pages or to parts within the page: a section with navigation links.“ [WHATWG 2010]

Jede Webseite, die aus mehr als einer Seite besteht, benötigt einen Navigationsblock mit Links zu weiterführenden Inhalten. Dieser befindet sich meistens unterhalb des Headers auf einer Zeile oder z.B. auf der linken Seite mit einem Link pro Zeile. Es ist üblich dafür einen eigenen DIV-Container zu erstellen und z.B. mit der ID „nav“ oder „navigation“ zu kennzeichnen.

HTML5 führt einen neuen Tag `<nav>` ein, der bisherige Blöcke wie `<div id="nav"></div>` ersetzen soll. Syntaktisch gesehen gibt es keine Unterschiede zwischen den beiden Möglichkeiten. Semantisch jedoch schon weil der Tag `<div>` und eine ID keine Bedeutung haben. `<nav>` kennzeichnet ganz klar, dass es sich hiermit um eine Navigation handelt.



Abbildung 3: Beispiel eines Navigations-Blocks

Beide müssen mit CSS formatiert werden und werden im Verlauf der Seite gleich behandelt. Nun stellt sich die Frage, warum man dieses Element verwenden sollte.

Einerseits ist das Tag kürzer und eventuell hebt es auch die Bedeutungen der Box besser hervor als mit einem DIV-Container und andererseits erleichtert es das Surfen im Internet für Personen mit einer Behinderung.

Personen mit Seh Schwierigkeiten sind auf sogenannte „Screenreaders“ angewiesen, die ihnen den Text einer Webseite vorlesen und denen sie sprachliche Befehle geben können. Sobald diese Programme die neuen HTML5 Elemente erkennen, kann der User sagen, dass ihm der Navigationsbereich vorgelesen werden soll. Der Screenreader erkennt nun aufgrund des Tags `<nav>` eindeutig, dass es sich dabei um die Hauptnavigation der Webseite handelt. Dies ist eine grosse Erleichterung für den User.

#### Listing 13: Navigation

```
<nav>
  <ul>
    <li><a href="index.php">Home</a></li>
    <li><a href="tags.php">Neue HTML5-Tags</a></li>
    <li><a href="audio.php">Audio</a></li>
    <li><a href="video.php">Video</a></li>
    <li><a href="canvas.php">Canvas</a></li>
    <li><a href="formular.php">Formulare</a></li>
  </ul>
</nav>
```

Das Navigation Element sollte auf einer Webseite nur einmal und nur für den Hauptnavigation block verwendet werden. Selbst wenn man am Ende der Seite eine Sammlung von weiterführenden Links bezüglich eines Artikels stehen hat, sollte darauf verzichtet werden, damit keine Unklarheiten entstehen.

Was den Footer-Bereich (*siehe Kapitel 4.3*) anbelangt, so enthält dieser meistens wichtige Verknüpfungen zu Kontaktdaten, Impressum oder AGBs, weshalb ein `<nav>` Tag gerechtfertigt wäre. Es wird aber nicht benötigt, da der Fussbereich selber bereits semantisch darauf hinweist, dass er solche Daten enthalten könnte.

## 4.5 Section

*„The section element represents a generic section of a document or application. A section, in this context, is a thematic grouping of content, typically with a heading.“* [WHATWG 2010]

Der HTML5-Tag `<section>` stiftet viel Verwirrung. Denn es kann leicht mit dem bisher verwendeten Blockelement `<div>` verwechselt werden.

Section hat im Gegensatz zu div eine semantische Bedeutung. Das heisst, dass laut [WHATWG 2010] der Inhalt, thematisch einen Zusammenhang aufweist und deshalb nicht aus dem Kontext gerissen werden sollte. Zusätzlich sollte es mindestens eine Überschrift enthalten.

Div hat keine semantische Bedeutung, sondern ist nur ein Container, der mit CSS speziell formatiert wird.

Ein Beispiel: Es wird eine Webseite mit einem Header oben, einem Navigationsbereich links, einem Footer unten und dem eigentlichen Inhaltsbereich rechts erstellt. Der Inhaltscontainer ist ein Container und wird mit `<div id="content"></div>` umschlossen (siehe Abbildung 4). An dieser Stelle wird nicht der Tag `<section>` verwendet, weil es sich um einen allgemeinen Blockelement handelt, das den ganzen Inhalt der Seite enthält.

Der Inhalt setzt sich zusammen aus einer Einführung bzw. Begrüssung und einem Newsbereich mit mehreren Artikeln (siehe Kapitel 4.6). Diese beiden Bereiche werden nun mit `<section>` unterteilt (siehe Abbildung 4 gelbe Rahmen). In Listing 14 folgt der Code:

#### Listing 14: Beispiel zu section

```
<body>
  <header>...</header>
  <nav>...</nav>
  <div id="content">
    <section id="introduction">
      <h1>Willkommen</h1>
      <p>...</p>
    </section>
    <section id="news">
      <h1>News</h1>
      <article>...</article>
      <article>...</article>
    </section>
  </div>
  <footer>...</footer>
</body>
```

The screenshot shows a website with a blue header containing the title "HTML5 - Überblick und Vergleich" and the author "Seminararbeit von Dario Züger". On the left, there is a navigation menu with links for Home, Neue HTML5-Tags, Audio, Video, Canvas, and Formulare. The main content area is divided into two sections: "Neue Tags" and "News". The "News" section contains two articles, each with a title, date, time, and a "Weiterlesen" link. The entire content area is enclosed in a yellow border, and the "News" section is also enclosed in a yellow border.

Abbildung 4: Beispielseite mit div und section-Blöcke

Laut [HTML5-DOCTOR 2010] kann man folgende Regeln anwenden, um zu entscheiden, ob `<section>` eingesetzt werden soll:

- Verwende es nicht, wenn ein Bereich nur speziell formatiert werden soll oder es ein allgemeiner Container ist; das ist ein `<div>`.

- Verwende es nicht, wenn `<article>`, `<nav>`, `<header>` oder `<footer>` passender wären.
- Verwende es nicht, ausser es hat am Anfang eine Überschrift.

## 4.6 Article

„The article element represents a self-contained composition in a document, page, application, or site and that is intended to be independently distributable or reusable, e.g. in syndication. This could be a forum post, a magazine or newspaper article, a blog entry, a user-submitted comment, an interactive widget or gadget, or any other independent item of content.“ [WHATWG 2010]

Dieses Element wurde bis jetzt bereits mehrere Male genannt, aber noch nicht weiter spezifiziert. `<article>` ist wie `<section>` ein Blockelement, geht jedoch bei der semantischen Bedeutung noch einen Schritt weiter. Laut [HTML5-DOCTOR 2010] kann man sich grundsätzlich fragen, ob der Text Sinn macht, wenn man ihn vollständig vom Rest einer Seite isoliert betrachtet. Oder könnte man diesen Text in einem RSS-Feed anbieten? Wenn beide Fragen bejaht werden können, wird `<article>` an Stelle von `<section>` eingesetzt. In Listing 15 wird dieser Tag eingesetzt für einen Newsbeitrag. Weiter vollstellbar wären einzelne Blog-Post oder User-Kommentare zu einem Eintrag wie es [WHATWG 2010] bereits definiert hat.

### Listing 15: Beispiel zu Article

```
<article>
  <header>
    <time datetime="2010-05-10T16:16:00" pubdate>10. Mai 2010, 16:16
    Uhr</time>
    <h1><a href="#" title="Link zum Artikel">Artikel 1</a></h1>
  </header>
  <p>Uatinus. Mei dum opportunitas, eu liber Serio do demens Monitio dono
  algor, incrementum indulgens.</p>
  <footer>
    <p class="author">Autor: Hans Muster | 2 Kommentare</p>
    <p class="more"><a href="#">&raquo; Weiterlesen</a></p>
  </footer>

  <section>
    <h2>Kommentare</h2>
    <article>
      <header>
        <h3>Geschrieben von: User 1</h3>
        <time datetime="2010-05-10T16:30:00" pubdate>10. Mai 2010, 16:30
        Uhr</time>
      </header>
      <p>...</p>
    </article>
    <article>
      <header>
        <h3>Geschrieben von: User 2</h3>
        <time datetime="2010-05-10T16:20:00" pubdate>10. Mai 2010, 16:20
        Uhr</time>
      </header>
      <p>...</p>
    </article>
  </section>
</article>
```

Das Beispiel zeigt einen Newsartikel, der einen Header mit dem Veröffentlichungsdatum und der Überschrift, einem Footer mit Informationen zum Autor und einem weiterführenden Link

enthält. Unterhalb des Footers sind zwei User-Kommentare angehängt, die selber wieder als Artikel interpretiert und zusammen in `<section>` verpackt werden.

## 4.7 Aside

„*The aside element represents a section of a page that consists of content that is tangentially related to the content around the aside element, and which could be considered separate from that content. Such sections are often represented as sidebars in printed typography.*“  
[WHATWG 2010]

Manche Newsseite setzt in Artikeln sogenannte Sidebars ein, in welcher weiterführende Links, Bilder oder ein Glossar eingefügt werden. Die Box kann getrost ausgelassen werden, ohne dass dabei der eigentliche Inhalt der Seite verändert wird.

Aside hat stets einen Bezug zu einem übergeordneten Element. Wird es innerhalb von `<article>` eingesetzt, kann es als Sidebar für den Artikel interpretiert werden. Befindet es sich ausserhalb von `<article>`, enthält es beispielsweise zusätzlichen Inhalt für die Webseite allgemein.

### Listing 16: Beispiel für eine Sidebar

```
<article>
...
<aside>
  
  <h1>Links</h1>
  <ul>
    <li>...</li>
    <li>...</li>
  </ul>
</aside>
</article>
```

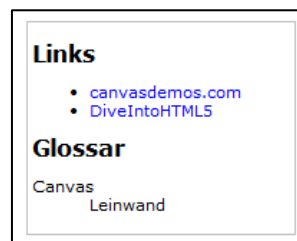


Abbildung 5: Beispiel zu Aside

## 4.8 Figure

„*The figure element represents some flow content, optionally with a caption, that is self-contained and is typically referenced as a single unit from the main flow of the document.*“  
[WHATWG 2010]

Es soll angenommen werden, dass ein langer Text geschrieben wird und ein Bild eingefügt werden soll. Normalerweise enthält ein Bild in einer Zeitschrift, einem Buch oder einer wissenschaftlichen Arbeit eine Überschrift, die es näher umschreibt. In HTML war es nicht möglich, solche Bezeichnungen einzufügen. Aus diesem Grund wurde der Tag `<figure>` eingeführt. Figure ist ein Blockelement, dessen Inhalt elementar für das Dokument ist, jedoch nicht seine Position. Es sollte nicht mit dem Tag `<aside>` (siehe Kapitel 4.7) verwechselt werden. Figure kann zum Beispiel Bilder - auch mehrere - oder Codebeispiele enthalten.

„*The figcaption element represents a caption or legend for the rest of the contents of the figcaption element's parent figure element, if any.*“ [WHATWG 2010]

Wie bereits erwähnt, wird der Tag `<figure>` meistens verwendet, um ein Element mit einer Überschrift zu versehen. Dies wird mit `<figcaption>` erreicht, das entweder ganz am Anfang oder am Schluss innerhalb des Figure-Blocks steht.

Bei Tabellen kann auf Figure verzichtet werden, weil sie ein eigenes Element `<caption>` besitzen.

#### Listing 17: Poster mit Überschrift

```
<figure>
  
  <figcaption>Poster des Filmes Big Buck Bunny</figcaption>
</figure>
```

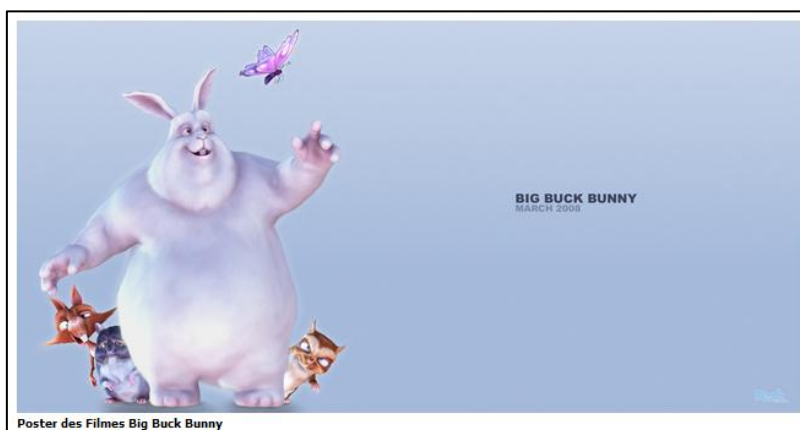


Abbildung 6: Poster mit Überschrift

## 4.9 Time

„The time element represents either a time on a 24 hour clock, or a precise date in the proleptic Gregorian calendar, optionally with a time and a time-zone offset.“ [WHATWG 2010]

Jeder Newsartikel oder Blog-Post enthält das Datum, an welchem er publiziert wurde. Dieses befindet sich entweder im Header oder im Footer von `<article>` und wurde zum Beispiel folgendermassen geschrieben: `<p class="pubdate">10. Mai 2010, 16:16 Uhr</p>`.

Das ist korrekter HTML-Code und daran ist auch nichts auszusetzen, ausser dass es nicht maschinenlesbar ist. Um Google und andere Roboter zu unterstützen, führte die WHAT Arbeitsgruppe das neue Element `<time>` ein. Zwischen die beiden Tags wird das Datum und falls erforderlich die Uhrzeit normal für den Menschen lesbar geschrieben, während das Attribut `datetime` für Maschinen gedacht ist. Dabei muss beachtet werden, dass zwischen Datum und Uhrzeit ein `T` steht.

#### Listing 18: Publikationsdatum ohne und mit Zeit

```
<time datetime="2010-05-10">10. Mai 2010</time>
<time datetime="2010-05-10T16:16:00">10. Mai 2010 2010, 16:16 Uhr</time>
```

Um es noch klarer zu machen, dass es sich um ein Veröffentlichungsdatum handelt, wird `<time>` mit dem Attribut `pubdate` ergänzt. Laut [WHATWG 2010] wird dadurch gekennzeichnet, dass es sich um das Publikationsdatums des Artikels oder, falls es nicht in einem Artikel eingebettet ist, der ganzen Seite handelt.

#### Listing 19: Publikationsdatum mit pubdate

```
<time datetime="2010-05-10" pubdate>10. Mai 2010</time>
```

Das time-Element kann auch eingesetzt werden, wenn ein normales Datum in einem Text eingetragen ist. Ein Beispiel wäre, dass es auf der Website eines Konzertanbieters für die Auftrittdaten der Künstler verwendet wird.

## 5 Formulare

Formulare bestehen meistens aus mehreren Textfeldern, eventuell einer Auswahlliste und einem Button, um die Daten abzuschicken. HTML5 definiert viele neue Typen und nützliche Attribute für Textfelder, die bereits eingesetzt werden können, egal ob sie vom Browser unterstützt werden oder nicht. Kennt ein Browser einen Typ nicht, so interpretiert er ihn als ein normales Textfeld. Man profitiert dann allerdings nicht von den neuen Funktionen, aber die Ausführung ist trotzdem gewährleistet.

Demnach ist es empfehlenswert, bereits jetzt bei den Formularen auf HTML5 zu setzen.

Das Tag `<form>` besitzt zwei neue Funktionen, die besonders erwähnenswert sind.

Falls die Bearbeitung eines Formulars unterbrochen wird, speichert der Browser die Daten lokal auf dem Computer ab und trägt wieder alle Werte in die Felder ein, sobald die Seite geöffnet wird, damit der User die Bearbeitung fortsetzen kann. Das Attribut `autocomplete="off"` unterbindet diese Zwischenspeicherung.

Formulare werden beim Abschicken meistens auf der Seite des Clients mit Hilfe von JavaScripts überprüft und es erscheint eine Meldung, falls ein Feld nicht oder falsch ausgefüllt wurde. Das Problem ist, dass solche Codes kompliziert und umfangreich werden können und einige Benutzer die Ausführung von JavaScripts aus Sicherheitsgründen auf ihrem Computer verhindern. HTML5 schafft Abhilfe, indem es grundsätzliche Überprüfungen übernimmt. Wurden beispielsweise nicht konforme Internet oder E-Mail Adressen eingetragen (siehe Kapitel 5.1) oder eine Zahl in einem Nummernfeld (siehe Kapitel 5.2) entspricht nicht dem vorgegebenen Wertebereich, erhält der User vom Browser eine Fehlermeldung beim entsprechenden Feld. Erst nach der erfolgreichen Überprüfung wird das Formular dem Server übermittelt und dort weiterverarbeitet. Mit dem Attribut `novalidate` wird die Validierung verhindert. Der Browser Opera unterstützt die Überprüfung bis jetzt als einziger Anbieter (siehe Abbildung 7).

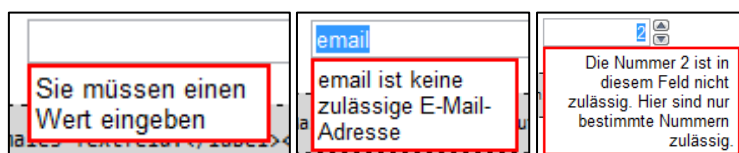


Abbildung 7: Beispiele für Fehlermeldungen im Browser Opera

Textfelder erhalten neben den vielen neuen Typen, die im Folgenden beschrieben werden, ebenfalls drei neue Attribute, die bisherige JavaScripts ersetzen:

- `placeholder=""`: Beim Aufrufen der Seite wird im Formular ein Platzhaltertext als Hilfestellung angezeigt. Dieser verschwindet, sobald das Feld angeklickt wird.
- `required`: Das Feld muss einen Wert enthalten. Wird kein Wert eingetragen, scheitert die Client-seitige Validierung des Formulars.
- `autofocus`: Enthält ein Feld dieses Attribut, wird es beim Laden der Seite ausgewählt und es kann sogleich Text eingetragen werden. In manchen Browsern hat es zugleich einen weiteren Rahmen, der es heraushebt.

Kennt ein Browser die neuen Attribute nicht, ignoriert er sie einfach. Es entstehen keine Fehler beim Laden.

Tabelle 2: Unterstützung der Attribute `placeholder`, `required` und `autofocus`

				
nein	nein	nein	ja	ja

### 5.1 E-Mail, URL und Suchfelder

In HTML 4.0 und XHTML werden für E-Mail und Website Adressen normale Textfelder eingesetzt. HTML5 sieht spezielle Typen für die Adressen und auch für ein Suchfeld vor.

Was ist der Unterschied. Die WHAT Arbeitsgruppe gibt keine genauen Richtlinien für die Implementierung der neuen Felder. Der Browser Opera zeigt aber zum Beispiel in den Feldern kleine passende Icons wie einen Briefumschlag an und führt eine client-seitige Validierung (siehe Kapitel 5) durch. Es wird überprüft, ob im E-Mail Feld ein „@“ und im URL Feld „http://“ enthalten sind. Ein anderes Beispiel bietet das iPhone, das die virtuelle Tastatur anpasst, indem es anstelle einer Leertaste das „@“ Zeichen und „.com“ einblendet.

#### Listing 20: Textfelder für E-Mail und URL

```
<input type="email" name="email" placeholder="@ " />
<input type="url" name="url" value="http://" />
```

Tabelle 3: Unterstützung von E-Mail und URL-Feldern

				
nein	nein	ja	nein	nein

Laut [WHATWG 2010] hat ein Suchfeld nur stylistische Differenzen gegenüber dem Textfeld. Browser können zum Beispiel den Rahmen den Suchfeldern des Betriebssystems anpassen, wie es in Chrome und Safari auf Mac OS X der Fall ist. Dort werden die Ecken abgerundet. Zusätzlich wird rechts im Feld ein Kreuzbutton angezeigt, bei dessen Aktivierung der Wert wieder gelöscht wird.

#### Listing 21: Suchfeld

```
<input type="search" name="field1" placeholder="Wonach suchen Sie?" />
```

Tabelle 4: Unterstützung von Suchfeldern

				
nein	nein	nein	ja	ja

## 5.2 Zahlenfelder

Es soll angenommen werden, dass man von einem Benutzer erfahren möchte, wie gut ihm eine Seite im Intervall von 1 bis 10 gefällt. Den Wert kann er in ein Textfeld eintragen. Wie gross sollen der Bereich - sprich wo ist das Minimum und wo das Maximum - und die Abstände zwischen den Zahlen sein? Um aussagekräftige Antworten zu erhalten, sollten die Wertemöglichkeiten eingeschränkt werden.

Dies wird mit den beiden neuen Typen `number` und `range` und den dazugehörenden Attributen `min`, `max` und `step` als Einschränkungen erreicht.

#### Listing 22: Zahlenfelder

```
<input type="number" name="favorite" min="1" max="10" step="1" />
<input type="range" name="favorite" min="1" max="10" step="1" />
```

Im Beispiel enthält das Attribut `step` den Wert 1. Es werden demnach alle ganzen Zahlen im Intervall von 1 bis 10 erlaubt. Würde es den Wert 2 haben, wären nur die Werte 1,3,5,7,9 zulässig.

Weshalb gibt es zwei Typen: `Number` ist ein Textfeld, in das Zahlen eines vorgegebenen Intervalls eingetragen werden. `Range` ist ein Schieber, dessen Wert mit der Maus ausgewählt wird, indem ein Balken nach links oder rechts gezogen wird. `Range` erleichtert demnach dem Benutzer die Auswahl; er weiss aber nicht genau, welchen Wert er auswählt, weil keine Zahlen angegeben werden. Um dies zu umgehen kann zusätzlich der Tag `<output>` mit dem Event-Handler `onforminput` implementiert werden. Diese Funktion, die unter ande-

rem auch zur Berechnung von Werten in verschiedenen Zahlenfeldern genutzt werden kann, zeigt den aktuellen Wert des Sliders an und wird bei jeder Änderung aktualisiert.

### Listing 23: Beispiel für Output

```
<input type="range" name="favorite" min="1" max="11" value="3" step="2" />
<output onforminput="value=range.value">3</output>
```

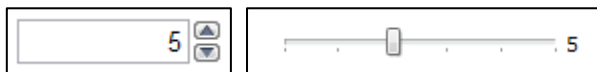


Abbildung 8: Number und Range mit Output in Opera

Es ist dem Browserhersteller überlassen, wie er die beiden Typen implementieren möchte. Opera, Safari und Chrome unterstützen den Slider bereits. Das Zahlenfeld wird aktuell nur in Opera verwendet. Es hat zusätzlich auf der rechten Seite zwei Pfeile, mit denen der User den Wert erhöhen oder verkleinern kann. Sobald ein Minimum oder Maximum erreicht ist, erscheint der Button in einem helleren Grau und kann nicht mehr betätigt werden.

Der Browser des iPhone's unterstützt nur den Typ Number, indem es die virtuelle Tastatur für eine Zahleneingabe anpasst.

Tabelle 5: Unterstützung der neuen Zahlenfelder

				
nein	nein	beide ja	number nein range ja	number nein range ja

## 5.3 Datum und Uhrzeit

Es soll angenommen werden, dass der Benutzer für eine Hotelzimmerreservation im Formular den Anreisetag angeben muss. Um die Eingabe möglichst benutzerfreundlich zu gestalten, soll ein Datum-Picker erscheinen, in welchem das Jahr, der Monat und der Tag übersichtlich und rasch ausgewählt werden können.

Mit HTML5 müssen keine zusätzlichen JavaScript-Bibliotheken mehr eingebunden werden. Gleich fünf neue Typen erleichtern die Arbeit:

- `type="date"`: Datum auswählen
- `type="month"`: Einen Monat auswählen
- `type="week"`: Eine Woche auswählen
- `type="time"`: Eine Uhrzeit auswählen
- `type="datetime"`: Ein Datum und eine Uhrzeit auswählen

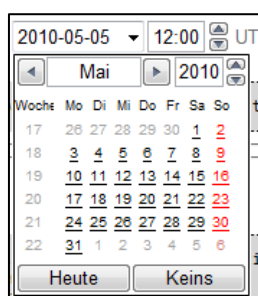


Abbildung 9: Das neue Datumfeld im Browser Opera

**Listing 24: Datumsfeld**

```
<input type="datetime" name="date" />
```

**Tabelle 6: Unterstützung des Datumsfeld**

				
nein	nein	ja	nein	nein

**5.4 Datalist**

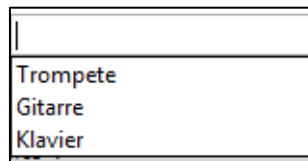
Um dem User eine Liste mit vorgegebenen Werten zur Verfügung zu stellen, aus denen er zutreffendes auswählen kann, gab es bis anhin drei Möglichkeiten:

- Checkboxes über den `<input>`-Tag für Mehrfachauswahl
- Radiobuttons über den `<input>`-Tag für Einfachauswahl
- Den Tag `<select>` mit den Werten als `<option>` für längere Listen geeignet und ebenso als Einfachauswahl.

Beim `<select>`-Tag ist auch eine Gruppierung der Werte möglich, indem zusätzlich `<optgroup>` eingefügt wird.

Bis jetzt fehlt eine Möglichkeit, bei welcher der User eine vorgegebene Auswahl erhält oder bei Bedarf selber einen neuen, nicht vorhandenen Eintrag vornehmen kann.

Eine Lösung wäre, nach den Listen ein leeres Textfeld einzusetzen mit dem Hinweis, dass es verwendet werden soll, falls obige Werte nicht zutreffen. Der gleiche Ansatz wird auf schriftlichen Fragebögen eingesetzt.

**Abbildung 10: Datalist in Opera**

HTML5 führt die sogenannte „Datalist“ für Textfelder ein. Der Tag `<input>` erhält neu das Attribut `list=""`, das die ID des Tags `<datalist>` anspricht. Für die Werte der Datalist wird `<option>` verwendet, jedoch ist der Wert im Attribut `value` und nicht zwischen den Tags wie bei einer Select-Liste gespeichert.

**Listing 25: Beispiel Datalist**

```
<input list="instruments" name="instruments" />
<datalist id="instruments">
  <option value="Trompete"></option>
  <option value="Gitarre"></option>
  <option value="Klavier"></option>
</datalist>
```

Klickt der User in das Textfeld, wird sogleich die Liste mit den drei Vorschlägen Trompete, Gitarre und Klavier angezeigt, von denen er einen auswählen oder selber einen neuen passenden Wert eintragen kann.

**Tabelle 7: Unterstützung der Datalist**

				
nein / nein	nein	ja	nein	ja

## 5.5 Farbauswahl

Jedes Text- oder Grafikprogramm enthält eine Box, in der die wichtigsten Farben für Texte oder Hintergründe auf einen Blick zur Auswahl stehen.

Mit dem Typ `color` soll diese Box auch im Browser ohne weitere JavaScript-Programmierung eingesetzt werden. Verwendung könnte es vor allem in einem Content Management System (CMS) finden, damit man im Live-Betrieb das CSS ändern könnte.

### Listing 26: Farbauswahl

```
<input type="color" name="color" />
```

**Tabelle 8: Unterstützung der Farbauswahl**

				
nein/nein	nein	nein	nein	nein

## 6 Multimediaelemente

### 6.1 Canvas

„The canvas element provides scripts with a resolution-dependent bitmap canvas, which can be used for rendering graphs, game graphics, or other visual images on the fly.“ [WHATWG 2010].

Canvas wird auf Deutsch mit Gemälde oder Leinwand übersetzt. Der Tag `<canvas>` generiert eine rechteckige Fläche, in welcher mit JavaScript Grafiken gezeichnet oder mächtige Applikationen erstellt werden können. In diesem Kapitel wird nur kurz anschaulich gemacht, wie Objekte wie Kreise, Rechtecke und Dreiecke gezeichnet werden können. Natürlich deckt das nur einen sehr kleinen Teil der Möglichkeiten ab, aber weitere Ausführungen würden den Rahmen dieser Arbeit sprengen.

Das Canvas-Element hat ohne den JavaScript Code keine Bedeutung, sondern stellt nur eine weisse Fläche dar, die man mit CSS auf weissem Hintergrund zuerst sichtbar machen muss, indem beispielsweise ein Rahmen um das Objekt herum definiert wird. Es benötigt nur drei Attribute: `width`, `height` und `id`, damit JavaScript das richtige Objekt auswählt.

#### Listing 27: Block für Canvas-Grafiken

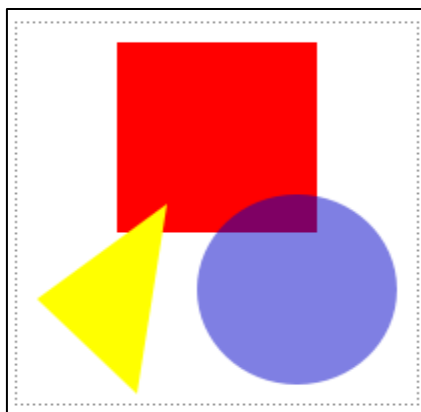
```
<canvas width="200" height="200" id="meincanvas">Leider unterstützt Ihr  
Browser das Canvas-Objekt noch nicht.</canvas>
```

Zwischen den Tags kann Text eingetragen werden, der angezeigt wird, falls ein Browser das Element nicht unterstützt.

Es folgt eine Beispiel-Implementierung, in welcher ein Quadrat, ein transparenter Kreis und ein Dreieck gezeichnet und mit Farben versehen werden. Anschliessend folgt die Erklärung.

#### Listing 28: JavaScript Beispielcode für Canvas-Block

```
<head>  
...  
<script>  
window.onload = function() {  
  var canvas = document.getElementById("meincanvas");  
  var context = canvas.getContext("2d");  
  context.fillStyle = "red";  
  context.fillRect(50, 10, 100, 100);  
  context.fillStyle = "rgba(0, 0, 200, 0.5)";  
  context.beginPath();  
  context.arc(140, 140, 50, 0, 2*Math.PI, true);  
  context.fill();  
  context.fillStyle = "yellow";  
  context.beginPath();  
  context.moveTo(10, 145);  
  context.lineTo(60, 195);  
  context.lineTo(75, 95);  
  context.fill();  
}  
</script>  
</head>
```



**Abbildung 11: Canvas-Grafik**

Der Code wird stets im `<head>`-Bereich der Seite zwischen die `<script>` Tags geschrieben. `window.onload` startet die Funktion beim Laden der Seite; sprich es zeichnet die Objekte. Zuerst wird mit `document.getElementById` der Code mit dem Canvas-Element verknüpft, das eine entsprechende ID besitzt.

Aktuell werden nur zwei-dimensionale Grafiken zugelassen, an einem 3D-Entwurf wird momentan noch gearbeitet. `getContext` erhält demnach den Wert `2d`.

Im nächsten Schritt wird das rote Quadrat erstellt (vgl. *Abbildung 11*). `fillStyle` enthält den Wert `red`, was ebenfalls `rgb(255, 0, 0)` in der RGB-Schreibweise entspricht. `fillRect` wird für Rechtecke verwendet und wird folgendermassen definiert: `fillRect(y-Wert, x-Wert, Breite, Höhe)`. Die `x`- und `y`-Werte kennzeichnen den Startpunkt oben links.

Als zweite Figur soll ein transparent-blauer Kreis gezeichnet werden. Die Transparenz wird erreicht, indem `fillStyle` den Wert `rgba(0, 0, 200, 0.5)` erhält. Der letzte Eintrag `0.5` kennzeichnet die Stärke der Transparenz. Diese liegt zwischen `0` und `1` und nimmt ab, je näher die Zahl bei `1` liegt. `arc(140, 140, 50, 0, 2*Math.PI, true)` definiert einen Kreis, dessen Mittelpunkt sich im Punkt  $(x,y)=(140,140)$  befindet und einen Radius von `50` Pixel hat. Die dritte Figur ist ein gelbes Dreieck. Dieses wird gezeichnet, indem der Cursor auf einem Punkt mit `moveTo(x-Wert, y-Wert)` festgesetzt wird. Anschliessend werden ausgehend von diesem Punkt zwei Linien mit `lineTo(x-Wert, y-Wert)` zu den anderen beiden Ecken gezogen. Der Befehl `fill` füllt das Dreieck mit der gelben Farbe, die vorher mit `fillStyle` definiert wurde.

Der Internet Explorer unterstützt bis und mit den Previews der Version 9 das Canvas-Element nicht. Entwickler stellen auf der Google Code Webseite einen JavaScript Code zur Verfügung, der die Funktionalitäten in den Browser integriert. Der Code ist unter dem Open-Source Apache License 2.0 Programm lizenziert und kann frei verwendet werden. Folgende Zeile muss in den `<head>`-Bereich eingetragen werden:

#### Listing 29: Zusätzlicher JavaScript-Code für den Internet Explorer

```
<!-- [if IE]>
  <script
  src="http://explorercanvas.googlecode.com/svn/trunk/excanvas.js"></script
  >
<![endif]-->
```

Die IF-Klausel in *Listing 29* definiert, dass der folgende Code nur im Internet Explorer kurz IE ausgeführt werden muss und von allen anderen Browsern ignoriert werden kann. Es wird empfohlen, das JavaScript-File zu verlinken und nicht den ganzen Code in die Seite zu integrieren, da es stets weiter entwickelt und verbessert wird.

**Tabelle 9: Unterstützung von Canvas**

				
nein	ja	ja	ja	ja

## 6.2 Video

Als vor fünf Jahren die Videoplattform Youtube gegründet wurde, hätte kaum jemand gedacht, dass es das Web multimedial revolutionieren würde. Einer der wichtigsten Faktoren für die schnelle Verbreitung und grosse Beliebtheit war, dass die Provider die Bandbreiten bei sinkenden Kosten im Heimbereich stark erhöht haben. Jeder kann schnell und einfach einen Film erstellen und ins Netz laden.

Um ein Multimediaobjekt in eine Webseite zu integrieren, müssen mit den Tags `<object>` oder `<embed>` Browsererweiterungen – sogenannte Plug-Ins – wie Adobe Flash oder Microsoft Silverlight gestartet werden, die die Wiedergabe übernehmen. Hat ein Benutzer ein Plug-In nicht installiert, kann das Element nicht abgespielt werden, weil dem Browser selber die Fähigkeit dazu fehlt. Apple verbietet sogar die Ausführung von Flash auf dem iPhone und dem iPad, weil die Software schon mehrmals mit Sicherheitslücken kämpfen musste und zu viele Ressourcen auf mobilen Geräten benötigt. Dadurch bleiben auf den Geräten viele Seiten schwarz.

Die Lösung liefert HTML5, indem es den Tag `<video>` einführt und somit erlaubt, dass Filme ohne Plug-Ins direkt im Browser abgespielt werden. Dazu müssen die Hersteller sogenannte Codecs – Algorithmus, mit dem ein Video kodiert ist – für Filme und Audio in ihren Browser integrieren. Jedem Hersteller ist es jedoch selber überlassen, welche Codecs er integrieren möchte.

Bis jetzt werden die Codecs H.264 und die beiden Open-Source-Codecs Theora Video und VP8 eingesetzt. H.264 ist ein patentierter Algorithmus, der für hochauflösende Filme – vor allem für Blu-ray – entwickelt wurde. Er wird von Adobe Flash und allen voran von Apple unterstützt. Codecs für Filme und Audio werden im Web immer in einem Container zusammengefasst. H.264 wird beispielsweise mit dem Audio-Codec AAC (Advanced Audio Coding) im Container MP4 mit der Dateierweiterung `*.mp4` angeboten.

Dieser Algorithmus wurde von Apple in Safari und von Google in Chrome integriert. Microsoft hat Ende April angekündigt, dass sie den H.264 Codec in den Internet Explorer 9 integrieren werden [Microsoft 2010]. Fraglich ist, wie die Pläne mit ihrer Silverlight-Technologie vereinbar sein werden, weil sie mit dieser Entscheidung höchstwahrscheinlich ihr eigenes Produkt kabbalisieren werden.

Theora Video ist ein Open-Source Codec, der zusammen mit Vorbis Audio im Ogg-Container angeboten wird. Mozilla Firefox und Opera setzen ausschliesslich auf ihn, um ein freies Web zu fördern.

Am 19. Mai 2010 kündigte Google auf seiner Entwicklerkonferenz den neuen Open-Source Codec VP8 im Containerformat WebM an. Ihr Vorhaben war es, einen neuen freien Codec zu entwickeln, der qualitätsmässig mit H.264 mithalten kann, wenig Ressourcen benötigt und von jedem verwendet und eingesetzt werden kann, ohne dass Lizenzgebühren entrichtet werden müssen. Zusammen mit Mozilla Firefox, Opera, Adobe und vielen Grafikkartenherstellern haben sie eine breite Allianz gebildet, damit der neue Algorithmus eine rasche Verbreitung findet. Die Browserhersteller veröffentlichten am gleichen Tag noch Testversionen ihrer Software mit VP8 integriert und die Google-Tochter Youtube begann mit der Rekodierung ihres Filmangebotes [WebM Project 2010]. Kurz darauf machte Microsoft bekannt, dass sie VP8 ebenfalls unterstützen werden [Microsoft 2010].



Abbildung 12: Abspielen eines Filmes in Opera

Der `<video>` Tag und seine Attribute werden wie folgt eingesetzt:

Wie beim `<img>` Tag sollte stets die Breite (`width`) und Höhe (`height`) mitgegeben werden. In `src=""` ist die Quelldatei eingetragen. Wird das Attribut `controls` verwendet, zeigt der Browser Standard-Schaltflächen wie Play/Pause, Laustärke und Zeitangaben an. Mit `autoplay` wird der Film beim Laden der Seite automatisch abgespielt. `preload` lädt das File herunter, startet aber die Wiedergabe erst auf Anfrage des Benutzers.

Es ist möglich, im Player während der Ladezeit ein Standbild als eine Art Filmplakat anzeigen zu lassen (siehe Abbildung 12). Dazu wird das Attribut `poster="file.jpg"` eingesetzt.

`<video>` ist kein empty-Tag wie `<img>`, sondern kann zwischen den beiden Tags Text enthalten, der angezeigt wird, falls der Browser das neue Element nicht unterstützt.

#### Listing 30: Film in Webseite einbinden

```
<video width="640" height="368" src="file.mp4" poster="file.jpg" controls>Ihr Browser unterstützt das HTML5-Tag für die Videowiedergabe nicht</video>
```

Wie bereits erwähnt setzen verschiedene Browser wegen der fehlenden Standardisierung verschiedene Codecs ein. Mit dem `<video>` Tag, wie es bis jetzt beschrieben wurde, würde man nur die Hälfte der Benutzer erreichen. Man muss demnach seine Filme gezwungenermaßen mehrere Male unterschiedlich kodieren. Im Attribut `src` kann nur eine Quelle eingetragen werden. Für mehrere Dateien weicht man deshalb auf den empty-Tag `<source>` aus.

#### Listing 31: Gleicher Film mit verschiedenen Codecs






```
<video width="640" height="368" poster="file.jpg" controls>
  <source src="file.mp4" />
  <source src="file.ogv" />
</video>
```

Wie weiss der Browser, welche Videodatei er herunterladen muss. Im obigen Beispiel würde er zuerst mit dem Download der ersten Datei beginnen und falls der Codec nicht unterstützt wird, den zweiten Film laden. Diese Lösung ist nicht befriedigend, da sie im schlechtesten Fall unnötig viel Bandbreite und Zeit beanspruchen würde. Aus diesem Grund kann ein Video identifiziert werden, indem der verwendete Codec im Attribut `type` mitgegeben wird. Der Browser lädt aufgrund dieser Informationen schliesslich nur die Datei herunter, die er auch abspielen kann.

**Listing 32: Zusätzlich mit Type-Angaben**

```
<video width="640" height="368" poster="file.jpg" controls>
  <source src="file.mp4" type="video/mp4; codecs="avc1.42E01E, mp4a.40.2"
 />
  <source src="file.ogv" type="video/ogg; codecs="theora, vorbis" />
</video>
```

**Tabelle 10: Welcher Browser unterstützt welche Video-Codecs**

					
H.264	nein	nein	nein	ja	ja
Theora Video	nein	ja	ja	nein	ja
VP8	nein	ja	ja	nein	ja

Damit der Betreiber einer Webseite bereits jetzt die neue Funktion trotz der fehlenden Integration des Internet Explorers einsetzen kann, haben Programmierer einen speziellen JavaScript-Code entwickelt. Dieser bietet eine Fallback-Lösung für den Internet Explorer und jeden anderen Browser, falls ein nicht unterstützter Codec abgespielt werden soll.

Wird dieser Code in den `<head>`-Bereich einer Seite kopiert, erkennt der Internet Explorer den Tag `<video>` und lädt einen Player basierend auf Adobe Flash. Es wird jedoch vorausgesetzt, dass mindestens eine Datei mit dem von Flash unterstützten Typ H.264 kodiert ist. Folgender Code von der Google Code Webseite<sup>3</sup> muss integriert werden (vgl. Listing 33):

**Listing 33: Fallback-Lösung**

```
<head>
  <script
  src="http://html5media.googlecode.com/svn/trunk/src/html5media.min.js"></
  script>
</head>
```

Eine weitere Eigenart hat auch Apple mit seinem Browser Safari. Sie unterstützen nur den H.264 Codec. Damit aber Filme abgespielt werden können, muss der Windows Benutzer den hauseigene Quicktime Player installieren. Demnach wurde die Möglichkeit der Filmwiedergabe nicht direkt in den Browser integriert, sondern dieser greift nur auf die Engine von Quicktime zu. Apple zwingt die User nebst Safari weitere Software zu installieren. Ob dies im Sinne von HTML5 ist, ist fraglich.

Auch wenn nun eine neue Möglichkeit angeboten wird, um Filme abspielen zu können, wird es wahrscheinlich noch Jahre dauern, bis sie vollständig akzeptiert wird. Zu viele Fragen und auch Probleme sind noch offen. Wie sieht es aus mit Streaming, Kapiteln, verschiedenen Sprachen oder auch Untertiteln?

Zusätzlich definiert das W3C und die WHAT Arbeitsgruppe nicht, welche Codecs den offiziellen Standard bilden. Es macht kaum Sinn, dass man seine Filme gleich dreimal verschieden kodieren muss. Das kostet unnötig Zeit und Speicherplatz.

Zuerst musste man sich zwischen H.264 und Theora Video entscheiden. Nachdem sich Microsoft offiziell für H.264 entschieden hatte, zeichnete sich langsam ein Sieger heraus. Dieser war aber schnell wieder vergessen, nachdem Google zusammen mit seiner Allianz den neuen Codec VP8 ankündigte, zu dem sich bisher nur Apple nicht bekannte. Damit dürfte wohl Theora Video aus dem Spiel sein.

Ein weiteres Problem dürfte sein, dass Microsoft erst mit dem Internet Explorer 9 den neuen Tag `<video>` unterstützen wird, der bekanntlich nicht mehr für Windows XP angeboten wird. Trotzdem kann man das neue Element bereits heute einsetzen. Dank der oben beschriebenen Fallback-Lösung auf Adobe Flash (siehe Listing 33), ist es möglich, dass alle User einen

<sup>3</sup> <http://code.google.com/p/html5media>

Film sehen können. Dazu muss man den Film nur mit H.264 kodieren. Sobald Flash offiziell den VP8 Codec unterstützt, kann auch dieser verwendet werden.

### 6.3 Audio

Nebst Filmen können auch Soundfiles einfacher in eine Seite integriert werden. Der Browser übernimmt die Arbeit für den Programmierer. Die Attribute entsprechen analog dem `<video>` Tag. Mit `src` wird die Quelle definiert und `controls` generiert Kontrollelemente wie Play/Pause-Buttons und die Lautstärke.

#### Listing 34: Einbetten einer Audio-Datei in die Webseite

```
<audio src="file.mp3" controls></audio>
```



Abbildung 13: Abspielen einer Audio-Datei in Opera

Ebenfalls können mit `<source>` mehrere Quellen für verschiedene Codecs definiert werden, damit möglichst jeder Browser unterstützt wird.

#### Listing 35: Gleiche Audio-Datei mit verschiedenen Codecs

```
<audio controls>
  <source src="file.mp3" type="audio/mp3" />
  <source src="file.wav" type="audio/wav" />
  <source src="file.ogg" type="audio/ogg" />
</audio>
```

Tabelle 11: Welcher Browser unterstützt welche Audio-Codecs

					
mp3	nein	ja	ja	ja	nein
wav	nein	nein	nein	ja	ja
Vorbis Audio	nein	ja	ja	nein	ja
aac	nein	nein	nein	ja	ja

Der Internet Explorer wird erst mit der finalen Version 9 den Audio-Tag unterstützen. Bis dahin muss auf Flash oder Silverlight ausgewichen werden oder es wird ebenfalls die Fallback-Lösung der Videofunktion (*siehe Kapitel 6.2*) unterstützt. Welche Codecs unterstützt werden, wurde noch nicht offiziell kommuniziert. Sicher sind die Formate Vorbis Audio und AAC, weil diese für die Video-Container MP4 und WebM benötigt werden.

## 7 Schlussbemerkungen

HTML5 bietet einige interessante Neuerungen, die dem Website-Entwickler die Arbeit massiv erleichtern können. Darunter fallen vor allem die neuen semantischen Tags, die eine klare Abgrenzung zum restlichen Inhalt bieten und viele DIV-Container überflüssig machen. Auch wird der Entwicklung im Web Rechnung getragen, indem neue Elemente für das sogenannte Web 2.0 geboten werden. Erwähnenswert ist der Tag `<article>` - ein neues Blockelement, das für Newsartikel, Blog-Posts oder Kommentaren von Usern zu einer Seite oder einem Artikel verwendet werden kann.

Mit den neuen semantischen Tags wird ebenfalls Personen mit einer Behinderung das Surfen im Web erleichtert. Screenreader erkennen beispielsweise wegen dem Tag `<nav>` schneller, wo sich der Hauptnavigationsblock befindet.

Browser merken sich neu auf Wunsch des Entwicklers die Eingabe des Users und trägt ihn automatisch wieder in die Felder ein, falls die Aktivität zwischendurch unterbrochen werden musste. Das einfache Textfeld hat praktisch ausgedient. Es wird erweitert mit Feldern für E-Mail Adressen, Suchfeldern, Datums-Pickern, Zahlenfeldern oder -slidern und weiteren. Alle diese neuen Funktionen ersetzen bisherigen JavaScript, der mühsam entwickelt oder integriert werden musste. Ist das Formular bereit für die Übermittlung zum Server, führt der Browser kurz eine client-seitige Validierung durch, bei der er prüft, ob alle notwendigen Felder ausgefüllt sind und den Inhalt enthalten, der vorgegeben wurde. Auch dies wurde bis anhin mühevoll mit JavaScript-Code realisiert.

Zu guter Letzt können noch die neuen Multimediaelemente erwähnt werden, die bereits einen guten Anfang bilden, jedoch muss noch viel Arbeit geleistet werden, bis sie bisherige Plug-Ins wie Adobe Flash vollständig ersetzen können. Unglücklicherweise definiert die WHAT Arbeitsgruppe keinen Standardcodec für Videofilme oder Soundfiles, weshalb die Dateien mit drei verschiedenen Codecs kodiert und bereitgestellt werden müssen. Die JavaScript-Entwicklergemeinde programmierte bereits brauchbare Fallback-Lösungen, damit Filme auch in Browsern abgespielt werden, die einen Codec nicht unterstützen. Dafür wird auf Flash zurückgegriffen.

Canvas bietet eine Leinwand für aufwändige Grafiken und Tools. Vielleicht wird es in ein paar Jahren, sobald 3D-Grafiken erstellt werden können, ebenfalls eine Alternative für Flash sein.

Eine grundsätzliche Frage, die man sich stellen sollte: Wie wird die Zukunft von Adobe Flash und Microsoft Silverlight aussehen?

In kurzer Frist werden sie wahrscheinlich kaum etwas an ihrer Attraktivität einbüßen müssen. Viele grosse Websites setzen auf Flash, um bewegte Bilder oder mächtige Tools anbieten zu können. Längerfristig dürfte es schwieriger werden. Apple verbietet beide Programme auf dem iPhone und dem iPad. Sie argumentieren, dass die Technik zu viele Ressourcen benötigt, Sicherheitsprobleme aufweisen und HTML5 die gleichen Möglichkeiten bietet.

Weniger schwierig wird es wahrscheinlich Silverlight von Microsoft haben. Die neue Smartphone-Software von Microsoft genannt „Windows Phone 7“ wird nur externe Applikationen ausführen, die auf Silverlight basieren. Sie entwickeln für ihre Technik demnach einen neuen Markt. Gerade im World Wide Web werden auch sie einen schweren Stand haben. Man könnte sagen, dass sie Silverlight karnalisieren, weil der Internet Explorer 9 ebenfalls Codecs zur Filmwiedergabe enthalten wird.

Viele aktuelle Browser unterstützen bereits die meisten neuen Funktionen von HTML5, während andere wie der Internet Explorer 8 noch Hilfe durch JavaScripts und Fallback-Lösungen benötigen. Trotzdem gibt es keinen Grund, weshalb man noch nicht auf die neue Standards setzen sollte. Sie erleichtern die Arbeit der Entwickler und die Websites sind gleich für die Zukunft gerüstet, die ganz klar im Zeichen von HTML5 sein wird.

Wann jedoch die Sprache endgültig durch das W3C standardisiert wird, ist noch ungewiss. Das kann vielleicht noch Jahre dauern. Weitere Jahre wird es dauern, bis HTML 4.01 und XHTML 1.0 vollständig abgelöst sein werden. Dank der Rückwärtskompatibilität von HTML5 stellt dies jedoch kein Problem dar.

Was kommt nach HTML5? Wird eventuell HTML6 entwickelt? Denkbar wäre es, aber es könnte auch sein, dass in Zukunft HTML5 kontinuierlich dem Fortschritt im Web angepasst und erweitert wird, ohne dass gleich ein neuer Standard verabschiedet werden muss. Somit könnten lange Entwicklungszeiten und Migrationen, die beide mehrere Jahre dauern, umgangen werden. Die Zukunft und die Bedürfnisse der User, Websiteentwickler und Browserhersteller werden es zeigen.

## 8 Literaturverzeichnis

[WHATWG 2010] WHATWG community: *HTML5 – Draft Standard*. Available: <http://www.whatwg.org/html5>, zugegriffen am 17.06.2010.

[Pilgrim 2010] Pilgrim, Mark: *Dive into HTML5*. Available: <http://diveintohtml5.org>, zugegriffen am 17.06.2010.

[Wöhr 2004] Wöhr, Heiko: *Web-Technologien – Konzepte – Programmiermodelle – Architekturen*, 1. Auflage, dpunkt.verlag GmbH, Heidelberg, 2004.

[WebHits 2010] WebHits: *Hit Counter und Live-Statistiken – Web-Barometer*. Available: <http://www.webhits.de>, zugegriffen am 15.05.2010.

[HTML5-Doctor 2010] *HTML5-Doctor - helping you implement HTML5 today*. Available: <http://www.html5doctor.com>, zugegriffen am 17.06.2010.

[Microsoft 2010] Microsoft Corporation: *IEBlog - The Windows Internet Explorer Weblog*. Available: <http://blogs.msdn.com/b/ie>, zugegriffen am 17.06.2010.

[WebM Project 2010] WebM Project: *WebM - an open web media project*. Available: <http://www.webmproject.org>, zugegriffen am 17.06.2010.

[Kröner 2010] Kröner, Peter: *HTML5 - Webseiten innovativ und zukunftssicher*. 1. Auflage, Open Source Press, München, 2010.

[Pilgrim 2010] Pilgrim, Mark: *HTML5 - Up and Running*. 1. Auflage, O'Reilly Media, 2010.

[Murphy 2010] Murphy, Christopher: *Beginning HTML5 and CSS3 - Next Generation Web Standards*. 1. Auflage, Apress, 2010.

[Spiering 2010] Spiering, Markus: *HTML5 Mobile - Webanwendungen für iPhone, Android & Co. entwickeln*. 1. Auflage, Franzis Verlag, 2010.

[Lubbers et al. 2010] Lubbers, Peter; Albers, Brian; Salem, Frank: *Pro HTML5 Programming - Powerful APIs for Richer Internet Application Development*. 1. Auflage, Apress, 2010.