WIRTSCHAFTSUNIVERSITÄT WIEN
Vienna University of Economics and Business

Bachelor Thesis

Title of the Bachelor Thesis:	W3C: HTML5
Author (last name, first name):	Schwarz, Sebastian
Student ID number:	0551854
Degree program:	BAWiSo-06/SZWWinf
Examiner (degree, first name, last name):	ao. Univ.Prof. Mag. Dr. Rony G. Flatscher

Date 20. 06. 2011

Signature

Contents

List of Figures

List of Tables

List of listings

Abstract

The W3C: HTML5 specification is developed by the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG) and is planned to become a W3C recommendation in 2014. HTML5 is also often used as a buzz word to refer to a number of other technologies related to the core specification, like CSS3 and JavaScript APIs like Geolocation or Web Workers. One of the main goals is better support for web applications.

Since 25 May 2011, the specification is feature complete and the stable features of the specification can already be used. Various web sites offer tutorials and browser compatibility information for HTML5 features. JavaScript frameworks and libraries allow developing HTML5 web pages that are fully backwards compatible.

HTML5 is also important for mobile devices, providing features like offline support for web applications, native support for audio and video playback and graphic rendering. Frameworks like PhoneGap allow the development of native mobile applications using HTML5, CSS3 and JavaScript.

However, HTML5's future as a W3C recommendation in 2014 does not look too bright. The WHATWG HTML Living Standard as a second specification for HTML that stays constantly in development compromises the integrity of the language. Tensions between the W3C and the WHATWG make future joint projects unlikely.

Die W3C: HTML5 Spezifikation wird vom World Wide Web Consortium (W3C) und der Web Hypertext Application Technology Working Group (WHATWG) entwickelt und soll 2014 eine W3C Recommendation werden. Unter dem Begriff "HTML5" werden auch häufig andere Technologien, wie z.B. CSS3 und JavaScript APIs wie Geolocation oder Web Workers zusammengefasst, die mit der Kernspezifikation verwandt sind.

Seit Mai 2011 ist die Spezifikation funktional komplett und die stabilen Funktionen der Spezifikation können bereits eingesetzt werden. Einige Webseiten bieten Tutorials und Browserkompatibilitätsinformationen für HTML5 Funktionen. JavaScript Frameworks und Libraries erlauben es, HTML5 Webseiten zu entwickeln, die rückwärtskompatibel sind.

HTML5 ist auSSerdem wichtig für mobile Endgeräte, da es Funktionen wie Offline-Support für Web Applikationen, natives Audio und Video Wiedergabe und Grafikrendering unterstützt. Frameworks wie PhoneGap erlauben es, native mobile Applikationen mit HTML5, CSS3 und JavaScript zu entwickeln.

Aber die Zukunft von HTML5 im Jahr 2014 sieht nicht unbedingt gut aus. Der HTML Living Standard der WHATWG ist eine zweite Spezifikation für HTML, die ständig in Entwicklung bleibt und die Integrität der Sprache gefährdet. Spannungen zwischen dem W3C und der WHATWG machen zukünftige gemeinsame Projekte unwahrscheinlich.

1. Introduction

The Hypertext Markup Language (HTML) is the markup language for creating web sites. HTML5 is the latest Version of HTML and is currently in development. In 1998 the World Wide Web Consortium (W3C) completed the specification of HTML4. It then started to develop XHTML 1.0, an XML-based Version of HTML4 that introduced no new features. After the completion of XHTML 1.0 in 2000, the W3C started working on XHTML2, which was not to be compatible with other Versions of HTML or XHTML. In 2004, Opera Software and the Mozilla Foundation proposed at a W3C workshop that the development of HTML should be continued and presented a first draft that described new features for HTML-forms called XForms. This proposal was rejected in favor of developing XML-based replacements for HTML. Therefore, Apple, Mozilla and Opera founded the Web Hypertext Application Technology Working Group (WHATWG) that continued to work on the presented draft. In 2007 the W3C created the W3C HTML Working Group and started working together with the WHATWG on a HTML specification, thereby discontinuing the development of XHTML2. The HTML5 specification published by the W3C differs from the HTML Living Standard specification of the WHATWG. However, most parts of the W3C: HTML5 specification are also part of the WHATWG HTML Living Standard [19].

HTML5 introduces many new features that are focused on supporting web applications and tries to fix many of the issues from previous versions while being backwards compatible to older versions of HTML. As the core of the W3C's Open Web Platform, HTML5 is also often used to refer to a number of technologies that are part of the platform. A selection of them is listed in table 1.1 [100].

Name	**Specification**	**W3C Work Group**
CSS3	`http://www.w3.org/TR/CSS/`	`http://www.w3.org/Style/CSS/current-work#CSS3`
Geolocation API	`http://www.w3.org/TR/geolocation-API/`	`http://www.w3.org/2008/geolocation/`
Indexed Database API	`http://www.w3.org/TR/IndexedDB/`	`http://www.w3.org/2008/webapps/`
MathML	`http://www.w3.org/TR/REC-MathML/`	`http://www.w3.org/2008/geolocation/`
SVG	`http://www.w3.org/TR/SVG/`	`http://www.w3.org/Graphics/SVG`
WAI-ARIA	`http://www.w3.org/TR/wai-aria/`	`http://www.w3.org/WAI/intro/aria.php`
Web Socket	`http://dev.w3.org/html5/websockets/`	`www.w3.org/2008/webapps/`
Web Storage	`http://www.w3.org/TR/webstorage/`	`www.w3.org/2008/webapps/`
Web Workers	`http://dev.w3.org/html5/workers/`	`www.w3.org/2008/webapps/`
WOFF	`http://www.w3.org/TR/WOFF/`	`http://www.w3.org/Fonts/WG/`
XMLHttpRequest API	`http://www.w3.org/TR/XMLHttpRequest/`	`http://www.w3.org/2008/webapps/`

Table 1.1.: Other technologies of the W3C's Open Web Platform[100]

2. The Development Process of HTML5

HTML5 is developed by the WHATWG and the W3C HTML5 Working Group. The W3C HTML5 specification is called a “snapshot” of the WHATWG’s HTML Living Standard” in the FAQs on the WHATWG’s homepage [159]. However, both specifications have been growing apart as the development processes are too different and parts of the W3C HTML5 specification have been transfered to separate W3C specifications. Still, most parts of the W3C: HTML5 specification and the related specifications can be found in the WHATWG HTML Living Standard, as convergence with the WHATWG is a goal of the W3C HTML Working Group [159, 19, 122].

2.1. Web Hypertext Application Technology Working Group (WHATWG)

The WHATWG was founded in 2004 by employees of Apple, the Mozilla Foundation and Opera Software, who wanted to continue developing HTML and not replace it with XHTML, as the W3C planned to do. According to the charter, the goal of the WHATWG is to “address the need for one coherent development environment for Web applications, through the creation of technical specifications that are intended to be implemented in mass-market Web browsers.” The WHATWG thinks this is necessary, because they are of the opinion that the specifications of the Internet Engineering Task-Force and the W3C do not cover this aspect [157].

2.1.1. Members

The members of the WHATWG are representatives from various browser manufacturers. You can only become a member by invitation. It is not made clear in the Charta, who can issue an invitation. The members of the WHATWG and their employers are listed in table 2.1 [157].

2.1.2. Specification Development Process

Everyone can contribute to the works of the WHATWG via mailing lists as contributing members. Potential contributers have to subscribe to them via the WHATWG website to be able to participate [153].

Apple	**Google**	**Mozilla**	**Opera**	**freelance**
David Hyatt	Ian Hickson	Brendan Eich	Anne van Kesteren	Dean Edwards
Maciej Stachowiak		David Baron	Håkon Wium Lie	
		Johnny Stenback		

Table 2.1.: Members of the WHATWG and their employers as of 9 June 2011 [157]

There are currently four mailing lists [155]:

1. Help for Web designers and HTML authors
2. The specifications - feedback on specifications, new proposals
3. Implementations - discussing the implementation of specifications
4. Commit-Watchers - notification of changes in the specification

The archives of this mailing lists are available online on the Homepage of the WHATWG under `http://www.whatwg.org/mailing-list`. Specifications are published on the website of the WHATWG (`http://www.whatwg.org`).

Each specification document has an assigned editor who is named by the members of the WHATWG. The responsibilities of the editor are to read e-mail feedback from the mailing list. He then decides based on the feedback and other sources (research, studies, blogs, forums, IRC) if the specification has to be changed. It is the responsibility of the editor that the specification is consistent and addresses the needs of the working group. The editor has the power of decision and there is no voting of any kind. The WHATWG members have the right to overrule the decisions of the editor and to replace him [150].

In urgent cases, contributing members can also contact the editor directly via E-Mail or IRC [151].

The charta states that a specification document has four stages it goes through [157]:

1. unstable draft
2. stable draft
3. ready for implementation in Web browsers
4. recommendation

At the beginning, a specification document is an unstable draft. A stable version of a document is published when the majority of the WHATWG members are of the opinion that it is ready. The current version of the document is then archived and a call-for-comments issued. At this stage, the document is still a draft and is not ready for implementation in Web browsers, unless for experimental reasons. The WHATWG members can then decide after considering the comments on the draft that the specification is ready for implementation. At this point, the document is again archived and a call-for-implementations announced. A specification becomes a recommendation if each section of the specification is implemented by at least two publicly available stable releases of Web browsers and if these implementations pass the test cases for this section [157].

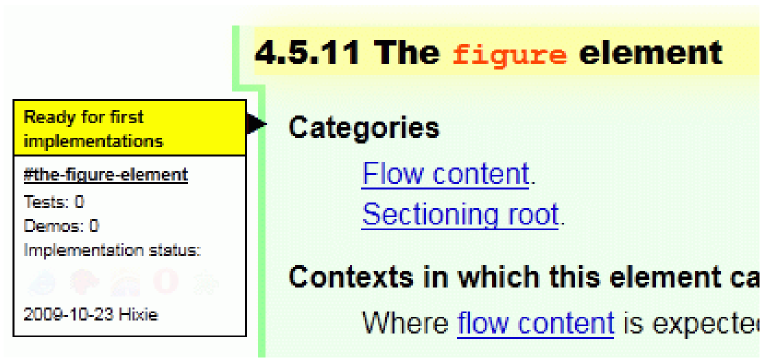
It is likely that the WHATWG charter has not been updated for some time. A polish translation of the charter dates back to 2007, otherwise there are no dates visible [157]. At least the specification development process does not seem to be up to date, because the WHATWG HTML Living Standard is developed differently and does not go through stages (see section 2.1.3) [154].

2.1.3. WHATWG HTML Living Standard

The WHATWG HTML specification is a part of the "Web Applications 1.0" specification [152]. Currently, the editor of the HTML-specification is Ian Hickson [73]. The latest version of the specification is available online under `http://www.whatwg.org/specs/web-apps/current-work/multipage/`.

There is no versioning, meaning that the specification should be a "living standard" that continually evolves. Therefore the WHATWG does not discern between different versions of HTML in the specification [158, 154].

Instead each section of the specification can have an individual status like "Idea", "Working draft" or "Implemented and widely deployed" that can be set through the annotation system [156]. Annotations become visible in the specification when the heading of a section is clicked on (see figure 2.1). Therefore the development process of the specification differs from the process outlined in the group charter.



4.5.11 The `figure` element

Ready for first implementations

#the-figure-element

Tests: 0

Demos: 0

Implementation status:

2009-10-23 Hixie

Categories

Flow content.

Sectioning root.

Contexts in which this element ca

Where flow content is expecte

Figure 2.1.: WHATWG HTML Living Standard with annotation visible on the left.

2.2. World Wide Web Consortium (W3C)

The W3C was founded in 1994 by Tim Berners-Lee, the inventor of the World Wide Web (WWW), and CERN (European Organization for Nuclear Research) supported by the Defense Advanced Research Projects Agency (DARPA) from the USA and the European Commission. It develops standards for the World Wide Web (WWW) [117].

2.2.1. Organizational structure

Host Institutions

Because the W3C is not incorporated it has for legal reasons three “Host Institutions” that also place most of the W3C staff, but are not W3C Members [139]:

- Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory [MIT/CSAIL] (US)
- European Research Consortium for Informatics and Mathematics [ERCIM] (France)
- Keio University Shonan Fujisawa Campus (Japan)

W3C Team

The W3C-Team is lead by Director Tim Berners-Lee and CEO Dr. Jeffrey Jaffe and consists of roughly 60-70 members. It is responsible for administrative tasks and technical leadership of the W3C [131]. The Director is the “lead technical architect” and among other things publishes the specifications and guidelines from the W3C [139].

Advisory Committee

The Advisory Committee (AC) advises to the W3C Team. Each organization that joins the W3C has one seat in the AC. The AC communicates over a mailing list and meets twice a year to receive information on the current resources and allocations of the W3C from the W3C-Team. The AC elects the nine participants of the Advisory Board (AB) [108].

Advisory Board

The Advisory Board (AB) consists of nine members elected for two years by the Advisory Committee and one Chair nominated by the W3C Team. Its main task is the management of the W3C processes [138] and it has no power to make decisions. It uses a confidential mailing list to communicate with itself and the W3C Team [136].

Technical Architecture Group (TAG)

The TAG concerns with issues about Web architecture. It consists of one Chair, who is appointed by the W3C-Team and usually the Director, three members appointed by the Director and five members elected by the Advisory Committee for two years terms [135].

Groups

The W3C groups are temporary teams of W3C Member Employees and Invited Experts that work together on a mission. Invited Experts are individuals that are not W3C Member Employees but are nominated to work in the group by the group Chair or by the Director. The group Chair is nominated by the Director. The Chair coordinates the group tasks [143].

Decision making inside a group and in the W3C in general is based on consensus. Before decisions are made, all feedback and opinions (also from the public) should be considered. Decisions can be made via the mailing lists or in meetings. A group member can either agree to, abstain from or dissent from a group decision by submitting a Formal Objection. In general, the number of dissents and abstentions should be as low as possible for each decision but groups can also specify treshholds in their Group Charta [112].

There are four kinds of groups [116]:

- Working Groups for creating deliverables (e.g. specifications, software)
- Interest Groups for discussing ideas
- Coordination Groups for communication management
- Incubator Groups for the concretization of new ideas

2.2.2. Membership

Any organization can apply for membership in the W3C. The application process via web form has three steps [128]:

1. Provide basic contact information
2. Provide organizational details
3. Request W3C review of application

The Director is able to reject applications for certain reasons defined in the W3C Process Document [133]. Members have to pay an annual membership fee which covers most of the funding of the W3C. It is calculated based on the country in which the headquarters of the organization are located, the organization type and the annual gross revenue [130].

The W3C has over 300 members. A full list of them is available at the W3C homepage. The members include major companies like Siemens, Sony, IBM, Microsoft, Apple and Google, universities like Oxford, Stanford and Amsterdam and browser vendors like Opera and Mozilla [113].

2.2.3. Contribution

All employees of a member organization have access to the member space and they can participate in a W3C group as W3C Member Employees [111]. Individuals that are not part of a member organization can only participate as Invited Experts [127].

2.2.4. W3C Recommendations

The W3C refers to specifications and guidelines created by a Working Group as Technical Reports [140].Technical Reports go through a number of Maturity Levels before finally becoming a W3C Recommendation [129]:

- Working Draft (WD)
- Candidate Recommendation (CR)
- Proposed Recommendation (PR)
- W3C Recommendation (REC)

Working Draft (WD)

A Working Draft is a public release of a Technical Report that is a work in progress. Everyone, not only W3C members, can give feedback. Normally, a Working Draft is intended to become a Recommendation. Before that can happen, a Last Call Announcement has to be made by the Working Group. After the Last Call there should not be any substantial changes to the document. There is a limited time frame to comment on the technical report until the Call for Implementations is issued and the Document Maturity Level changes to Candidate Recommendation [107].

Candidate Recommendation (CR)

A Candidate Recommendation is a Technical Report that is ready for experimental implementation. It has been thoroughly reviewed and meets the Technical Requirements defined by the Working Group but it can still change from implementation feedback [129].

Proposed Recommendation (PR)

Technical Reports that shall become a W3C recommendation can be sent to the Advisory Committee for review if all features in it have been implemented. The minimum duration of a review is four weeks [110].

W3C Recommendation

A W3C Recommendation is a Technical Report that the W3C has decided to support and can be seen as a standard [129].

2.2.5. The W3C HTML Working Group

The W3C HTML Working group is currently the only W3C group working entirely in public. The mission of the W3C HTML Working group is to continue to work on the latest W3C HTML specification: currently HTML5. The Chairs of the group and their employers are [122]:

- Sam Ruby (IBM)
- Paul Cotton (Microsoft)
- Maciej Stachowiak (Apple)

The group has its own publicly available mailing list for discussion [132]. To join the mailing list, one has to create a W3C public account first to be able to apply as a "Public Invited Expert". Only if the application is accepted, one is allowed to participate. The application criteria are however not as strict as in other W3C Groups [115]. Public contribution is possible through a feedback web form in the HTML5-specification and a public bug database [109].

The homepage of the W3C HTML working group is available under `http://www.w3.org/html/`.

The HTML5 specification is one of the deliverables of the W3C HTML Working Group. All deliverables can be found in the Working Group Charter [121]. The W3C HTML5 specification is part of the Open Web Platform, which includes Open Technologies for the Web like CSS3, SVG and MathML [159].

The W3C HTML Working Group publishes two versions of the HTML5 specification:

- Editor's Draft: `http://dev.w3.org/html5/spec/Overview.html`
- Working Draft: `http://www.w3.org/TR/html5/`

The editor of both documents is Ian Hickson, who is also editor of the WHATWG HTML specification. The Editor's Draft is the latest daily version based on the current version of the WHATWG HTML Living standard. However, the WHATWG HTML specification has a larger scope and parts of the specification are not relevant for HTML5. There are also differences between the two versions based on W3C HTML Working Group decisions. These differences are listed in the WHATWG HTML specification in section "1.1 Is this HTML5?" [154]. The Working Draft is the latest published version by the W3C HTML Working Group based on the Editor's Draft.

2.3. Design Principles

2.3.1. Backwards Compatibility

HTML5 has to be backwards compatible, meaning that a parsing engine that supports HTML5 also supports all previous versions of HTML and XHTML1. Therefore also features that have already been implemented in web browsers but are not part of any HTML specification are reverse engineered and documented for the HTML5 specification. Also, error handling is defined in the specification. Therefore, web pages written in previous versions of HTML can still be displayed by browsers that support HTML5 [64, 142].

2.3.2. Compliance with other specifications

The W3C HTML5 specification complies in most cases to several other W3C specifications it is related to. The only exceptions are "willful violations", which reasons are noted in the specification [64].

2.3.3. Test Cases and Validation Tools

The Group must provide test cases for all features in the specification. These tests shall help implementing the specification (in a web browser) and determine to what extent an implementation meets the specification. The current test suite is available under `http://w3c-test.org/html/tests/harness/harness.htm`. Also, validation tools for HTML5 have to be available to allow web authors to validate their HTML5 documents [160].

2.3.4. Interoperability

Implementations according to the specification must be interoperable. Two independent implementations should have the same test results. This ensures that different Browsers that support the W3C HTML5 specification display a website the same way [126].

2.3.5. General Adoption

The HTML5 specification has to be accepted and used by the user community and the industry. It has to have practical relevance. Therefore Invited Experts and the "communities of expertise" play an important role in the development process [122].

2.3.6. Support for Web Applications

HTML5 provides features that make it easier to develop feature-rich web applications, like native multimedia capabilites, offline support and drag and drop. Several JavaScript APIs developed together with the HTML5 specification provide features like simultaneous execution of scripts (Web Workers API) and graphics rendering (Canvas 2D Context API) [142].

2.3.7. Adaption to user device

HTML5 is intended to provides more possibilities for structuring content that enable the adaption of web pages by the web browser to the user device it is displayed on. This means better support for devices like mobile phones, televisions or tablet pcs [142].

2.4. Timeline

The planned end date for the W3C HTML working group is December 31, 2014 [122]. By this time, the HTML5 specification is planned to be a W3C recommendation with a complete test suite.

The specification is already feature complete. "Last Call" was issued on 25 May 2011 [114]. Feedback for the Last Call specification can be given until 2 August 2011 [5].

The milestone plan of the W3C HTML Working Group is listed in table 2.2.

Milestone	**Date**
Last Call Working Draft	25 May 2011 – reached
Candidate Recommendation	2012 Q2
Proposed Recommendation	2014 Q1
W3C Recommendation	2014 Q2

Table 2.2.: Milestones of the HTML Working Group, as listed on the Group homepage [121]

2.5. Current usage

Although the W3C HTML5 specification is still a draft, some of its features have already been implemented in web browsers. There are already web sites that use HTML5-based syntax, sometimes combined with compatibility hacks for older browsers where necessary (See section 6). The W3C also states that the parts of HTML5 that are already interoperable and provide fallback mechanisms can already be used [141, 114].

3. Syntax and Main Features

This chapter contains a short introduction to the syntax and main features of HTML5, including code examples. In addition, four JavaScript APIs – namely HTML Canvas 2D Context (3.7), Web Storage (3.11), Web Workers (3.12) and (Web Sockets3.13) – are introduced that are not part of the W3C HTML5 specification. However these JavaScript APIs are closely related to HTML5 [1] and often referred to as part of HTML5 or HTML5 technology [142]. Since they also provide important features for web applications, the author has decided to include them.

3.1. Basic HTML5 Document

An HTML document is a text file that uses the filename extension ".html". To structure a text, HTML uses elements that are wrapped around the text with start tags, e.g. <html>, and end tags, e.g. </html>, and create a tree structure. Most elements must have a start tag and an end tag but there are elements that only have one tag, like <meta>. Also an element must not be closed, if it has another element inside that is still open. This ensures a valid tree structure. Elements can have attributes that are used to specify the characteristics of the element. Attributes are defined inside the start tag, e.g. <html lang="en">. Quotes around attribute values are optional. HTML is not case sensitive [20]. Listing 1 shows the code for a basic HTML5 document.

```html
<!DOCTYPE html>
<html lang="en">
<!-- A simple HTML document -->
 <head>
  <meta charset="utf-8">
  <title>Hello World!</title>
 </head>
 <body>
  <p>Hello.</p>
 </body>
</html>
```

Listing 1: Simple HTML document

[1] The Canvas 2D Context was part of the HTML5 specification until January 2010 [65].

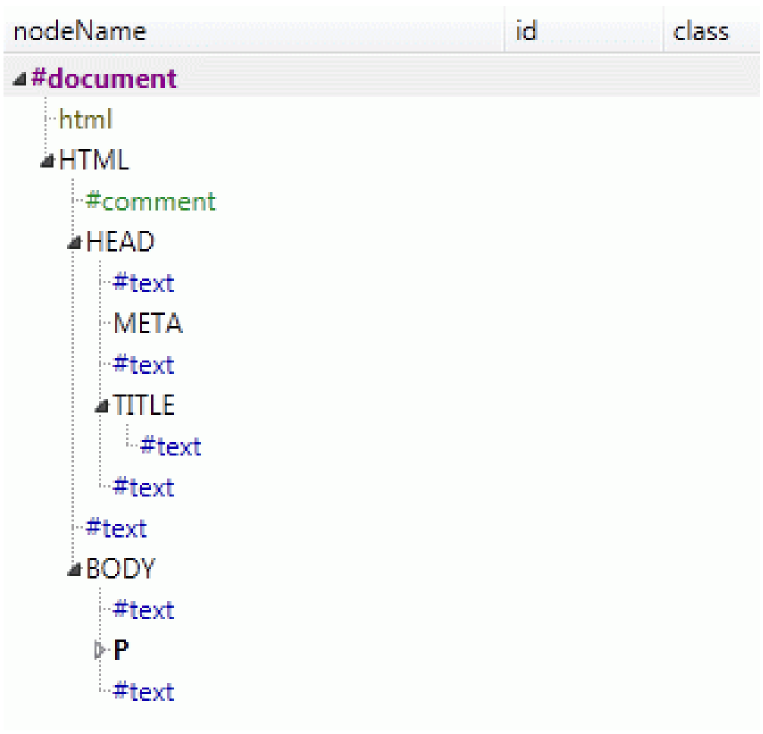
In the first line, the Document Type Declaration is made via the DOCTYPE element. A Document Type Declaration defines the type of Markup Language that is used in a document. It is also possible to write HTML5 documents in XML syntax, called XHTML5. XHTML documents use basically the same syntax as HTML documents, but they are XML documents and therefore they must be well-formed and valid according to the XML specifications or they can not be parsed[2].

The <html> element is the root element of the document. lang is an optional attribute of the HTML element that specifies in which language the content is written, in this case English.

The <head> element contains information on the content of the document, like the character encoding and the title.

The <body> element wraps around the actual content of the document. The <p> element marks a paragraph.

The in-memory representation of an HTML document is called a Document Object Model (DOM) tree that shows the tree structure of the elements. Figure 3.1 shows how the DOM tree of the example HTML document looks [18].



```
nodeName                         id        class
#document
  html
  HTML
    #comment
    HEAD
      #text
      META
      #text
      TITLE
        #text
      #text
    #text
    BODY
      #text
      P
      #text
```

Figure 3.1.: Dom tree of the basic HTML document displayed in the DOM Inspector Addon for Firefox 4 (available under `https://addons.mozilla.org/de/firefox/addon/dom-inspector-6622/`)

[2]For more information on XML please visit `http://www.w3schools.com/xml/default.asp`

HTML syntax is more forgiving than XML syntax because a document is also displayed if the syntax is not correct or invalid. The HTML parser then tries to correct the errors found in the code according to the error handling described in the specification. A web browser that implements the HTML5 specification would display the HTML document in listing 2 after correcting the errors found in the syntax [76].

```
<!-- An HTML document with wrong syntax-->
<title>Hello World!</title>
<h1>Hello.</h1><p>This is a test.
<h2>It is not <p>over</h2> yet.</p>
```

Listing 2: Minimal HTML document with syntax errors

The syntax errors in this document are:

1. There is no DOCTYPE specified
2. The root element <html> is missing
3. The <head> element is missing
4. The <body> element is missing
5. Line 6: The <p> element is not closed
6. Line 8: The <h2> element is closed, but the <p> element inside is still open.

The Internet Explorer 9 displays this corrected HTML document when opening the previous code:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML Strict//EN">
<META http-equiv="Content-Type" content="text/html; charset=windows-1252">

<html>
<head>
 <style></style>
</head>

<body>
 <h1>Hello.</h1><p>This is a test.</p>
 <h2>It is not <p>over</p></h2><p>yet.</p>
</body>
</html>
```

Listing 3: Corrected code of minimal HTML document with syntax errors displayed in Internet Explorer 9

The HTML parser of the Internet Explorer 9 (IE9) corrected the syntax errors:

1. Added DOCTYPE[3]
2. Added <html> element
3. Added <header> element (including an empty <style> element, although this is not mandatory)
4. Added <body> element
5. The <p> element is now closed
6. The <p> element inside the <h2> element is now closed before the <h2> element and a new <p> starttag was added.

3.2. Differences from HTML4

The current differences between HTML4 and HTML5 are described in a separate document of the W3C named "HTML5 differences from HTML4" that is available under `http://dev.w3.org/html5/html4-differences/`.

Because HTML5 is backwards compatible, all elements from HTML4 have to be supported by user agents (in most cases web browsers). However, there are several elements and attributes that must not be used in HTML5. These includes all elements that are used to specify the formatting or appearance of an HTML document. In HTML5 Cascading Stylesheets (CSS) have to be used for this purpose[4] [103]. HTML5 also introduces new elements and attributes and changes existing elements and attribute values. E.g. to structure a web page the elements <header> and <footer> are available and the type attribute of the <input> element has more allowed values in HTML5 to increase its usability in HTML forms [103]. In addition to that, HTML5 makes use of new Application Programming Interfaces (API) that are intended to provide features like drag&drop, audio/video playback and offline use for web applications. The existing APIs HTMLDocument and HTMLElement have been extended to provide more functionality [102].

Unlike previous versions of HTML, HTML5 is not a Standard Generalized Markup Language (SGML) anymore. SGML-Documents use Document Type Definitions (DTD) to describe their structure. DTDs can be used to validate SGML documents and an SGML document must reference the DTD it is based on. Therefore, HTML4 has a longer document type definition (DOCTYPE) because it includes a reference to a DTD. In HTML5, the DOCTYPE is only necessary for correct rendering of HTML5 documents in current web browsers. <!DOCTYPE html> is the shortest syntax for that [106].

[3]The IE9 inserts a DOCTYPE that provides more information than required by the HTML5 specification. Please see section 3.2

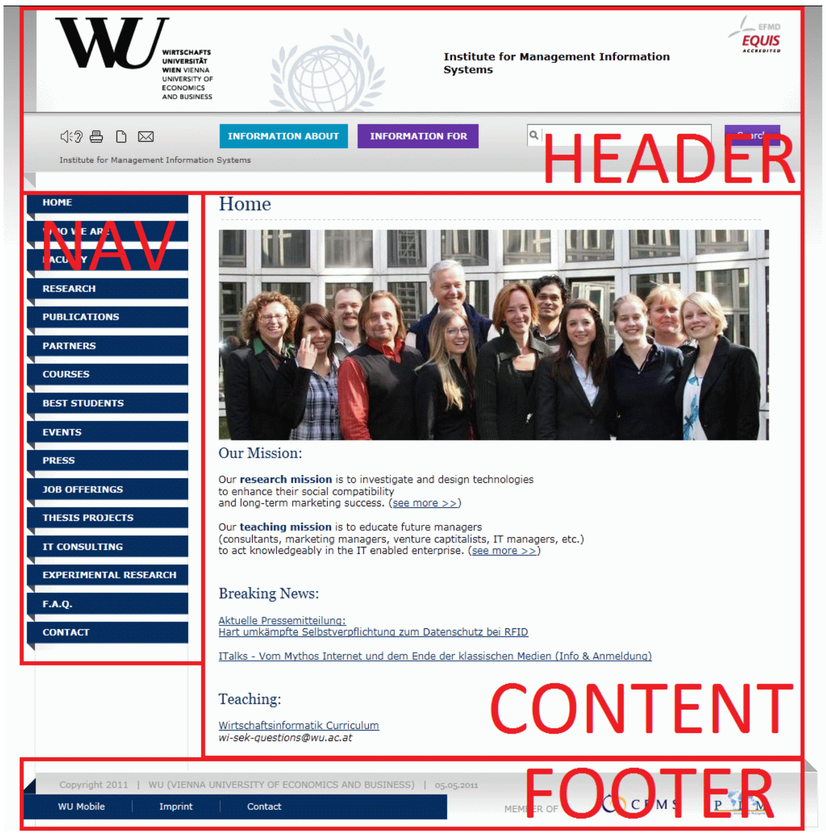
[4]For more information on CSS, please visit `http://www.w3.org/Style/CSS/`

HTML5 supports Internationalized Resource Identifiers (IRI) if the document uses UTF-8 or UTF-16 for character encoding. IRIs are like Uniform Resource Identifiers (URI) except that URIs can only be composed of ASCII characters and IRIs allow the universal character set [105, 7].

HTML5 allows the use of Scalable Vector Graphics (SVG) and Mathematical Markup Language (MathML) elements inside an HTML document within the elements <svg> and <math>. SVG is an XML language for creating scalable vector graphics and MathML is an xml language for describing mathematical notations[5] [104].

3.3. Structuring Content

HTML5 introduces new elements for structuring content. These elements are based on how most web pages in the Internet are structured. Usually there is a page header with the title of the page and a logo, one or more navigation menus, the main content and a footer with copyright and contact information, as shown in figure 3.2.



Our Mission:

Our **research mission** is to investigate and design technologies to enhance their social compatibility and long-term marketing success. (see more >>)

Our **teaching mission** is to educate future managers (consultants, marketing managers, venture captitalists, IT managers, etc.) to act knowledgeably in the IT enabled enterprise. (see more >>)

Breaking News:

Aktuelle Pressemitteilung:
Hart umkämpfte Selbstverpflichtung zum Datenschutz bei RFID

ITalks - Vom Mythos Internet und dem Ende der klassischen Medien (Info & Anmeldung)

Teaching:

Wirtschaftsinformatik Curriculum
wi-sek-questions@wu.ac.at

Copyright 2011 | WU (VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS) | 05.05.2011

Figure 3.2.: The structure of the Homepage of the Institute for Management Information Systems of the Vienna University of Economics and Business (`http://www.wu.ac.at/ec/`

[5]For more information please visit `http://www.w3.org/Math/` and `http://www.w3schools.com/svg/`

In HTML4, this structure is in most cases created with <div> elements and a class or id attribute, e.g. <div class="header">. In HTML5, there are several new elements for creating a structure to allow web browsers to understand it [13, 88].

There are also more elements for creating an outline available. Outlines are created using sections and headings. There are sectioning root elements (3.3.1) and sectioning content elements (3.3.2). Elements of heading content (3.3.7) define the header of a section or can imply a section themselves.

3.3.1. Sectioning Root Elements

Sectioning Root Elements can have their own independent outlines, they are not part of the outline of their ancestor elements. The following elements are sectioning root elements[36]:

- blockquote
- body
- details
- fieldset
- figure
- td

3.3.2. Sectioning Content Elements

Sectioning content elements are used to create subsections inside sectioning root elements or other sectioning content elements. They are therefore part of the outline of their ancestor elements. These are [35]:

- article
- aside
- nav
- section

3.3.3. The <article> Element

Content inside an <article> element should be independently distributable or reusable, e.g. in an RSS feed[6]. Examples are a blog/forum post, an e-mail, a newspaper article or an interactive widget. <article> elements can also be nested to relate articles with eachother, e.g. to relate comments to a blog post. The <article> element usually has a heading, a <header> and/or a <footer> element [39].

[6]For more information on RSS, please visit `http://www.w3schools.com/rss/`.

```
<article>
        <h1>Blog Post</h1>
        [...]
        <article>
                <p>Comment</p>
        </article>
<article>
```

Listing 4: article Element example

3.3.4. The <aside> Element

The aside element encompasses content that is related to the content around it. It can be used for sidebars, advertisements, pull quotes or to group <nav> elements [40].

```
<aside>
        <nav>
                <h1>Main Menu</h1>
                [...]
        <nav>
        <nav>
                <h1>Administrator Menu</h1>
                [...]
        <nav>
</aside>
```

Listing 5: aside element example

3.3.5. The <nav> Element

The <nav> element is wrapped around groups of navigation links and creates a section. It is only intended for main navigation links of a web page [54].

```
<nav>
        <h1>Main menu</h1>
        <ul>
                <li><a href="news.html">News</a></li>
                <li><a href="about.html">About me</a></li>
                <li><a href="blog.html">Blog</a></li>
        </ul>
<nav>
```

Listing 6: nav element example

3.3.6. The <section> Element

The <section> element creates a new section for thematic grouping of content that should be listed in the document's outline. It normally contains a heading [60].

```html
<article>
	<h1>Cars</h1>
	<section>
		<h1>European Cars</h1>
		<h2>Mercedes-Benz SLS AMG<h2>
		<h2>Porsche 911<h2>
	</section>
	<section>
		<h1>U.S. Cars</h1>
		<h2>Chevrolet Corvette (C6)</h2>
		<h2>Shelby Mustang GT 500</h2>
	</section>
</article>
```

Listing 7: section element example

3.3.7. Headings

The heading of a section is defined using elements of heading content. These are:

- <h1> to <h6>, 1 being the top rank and 6 the lowest
- <hgroup>

The first heading element inside a sectioning content element is the heading for that section. All following headings that are of higher or equal rank (1-6) start new implied sections and all with lower rank start implied subsections to the previous section [31].

The <hgroup> element is used to group multiple headings to add a tagline, a subheading or alternative titles to a heading. In the outline, only the heading element with the highest rank is visible. It was shortly removed from the W3C HTML5 specification at the beginning of May 2011 but added again after Sam Ruby (HTML5 WG Chair, see section 2.2.5) requested a revert of the change [94].

```html
<hgroup>
	<h2>Vienna University of Economics and Business</h2>
	<h1>WU</h1>
</hgroup>
```

Listing 8: hgroup element example

3.3.8. The <header> Element

The header element is wrapped around introductory content, navigational aids, like a table of contents, logos or search fields. However, it does not create a new section. It normally contains heading (<h1> – <h6> or <hgroup> elements [49].

```
<body>
        <header>
                <h1>Welcome to my homepage!</h1>
                <p>Enjoy your stay!</p>
        </header>
        [...]
</body>
```

Listing 9: header element example

3.3.9. The <footer> Element

The <footer> element is used to add additional information, like a date, the author or copyright information, to a sectioning content or sectioning root element. A section can also have multiple footers, e.g. at the top and the bottom of a section. The <footer> element can also be wrapped around fat footers. “Fat Footers” is a term coined by web designers to refer to footers at the bottom of a page that contain a lot of material, like images, additional navigation links or related content. Normally, fat footers have the <body> element as nearest ancestor [48].

```
<article>
<h1>HTML5</h1>
<p>This is an article about HTML5.</p>
<footer>
        <p>Written by John Smith</p>
</footer>
</article>
```

Listing 10: footer element example

3.3.10. The <address> Element

The <address> element contains contact information for the nearest <body> or <article> element ancestor. It must only be used for contact information relevant to a web site or an article and must not contain other information. It is typically used inside <footer> elements [38].

```
<footer>
        <p>Written by John Smith</p>
        <address>
                You can contact John Smith under <a href="mailto:john.smith@smith.com'
        <address>
</footer>
```

Listing 11: address element example

3.3.11. The <figure> Element

The <figure> element is used for self-contained content like images, videos, audio tracks, code listings and diagrams. A caption or legend can be added with the <figcaption> element. It is also possible to include multiple elements in one <figure> element [47].

```
<figure>
 <img src="http://www.wu.ac.at/wuneubau/bilder/Campus_480_280.gif"
      alt="New WU campus">
 <figcaption>Modell of the new WU campus</figcaption>
</figure>
```

Listing 12: figure element example

3.3.12. Outline Example

A simple outline can be created by using different levels of heading elements like in example 13. This example does not make use of any new HTML5 features.

```
<h1>Vehicles</h1>
<h2>Sea</h2>
<h3>Ships</h3>
<h3>submarines</h3>
<h2>Air</h2>
<h3>Plane</h3>
<h3>Helicopter</h3>
```

Listing 13: Outline example 1

Example 13 would result in the following outline:

1. Vehicles
   a) Sea
      i. Ships
      ii. Submarines

a) Air
   i. Planes
   ii. Helicopters

In HTML5, the same outline can be created using only the <h1> element and Sectioning Content Elements. In listing 14, the <article> element is used, assuming that each content of a section could be published as independent content.

```
<article>
        <h1>Vehicles</h1>
        <article>
                <h1>Sea</h1>
                <article>
                        <h1>Ships</h1>
                </article>
        </article>
</article>
```

Listing 14: Outline example 2

Example 14 would however result in a different outline in older HTML versions. All headings would be on the same level [23].

3.3.13. Structuring Example

The code for an HTML5 document that creates a basic structure and outline using the elements described in this section can be found in appendix A.1.1.

3.4. Forms

An HTML form is used to allow users to enter data. This data can then either be sent to a server for processing or used in client-side scripting. It can be created using the <form> element. The main element that is used to prompt data is the <input> element. The type attribute is used to specify the kind of the input field, e.g. plain text, radio button or check box. HTML5 adds new attributes and input types to the <input> element to enable browsers to increase the usability of html forms through appropriate user interfaces for each input type and client side input validation. In HTML5, the <input> element can also be used outside a <form> element but then it must be associated with a form using the form attribute [30].

For the following listings an online hotel reservation form is used as an example to introduce some of the features of HTML5 forms. The user data which is requested are:

- full name
- salutation (Mr., Ms. or user defined)
- e-mail
- homepage
- phone number
- date of arrival
- date of departure
- beds (1 to 4)

To create an HTML form, a <form> element is used inside the <body>. Normally, the method attribute would be used to define how the form sends submitted data (in our example POST) and the server URL the data is sent to via the action attribute. But this is not relevant for this example and these attributes were also available in previous HTML versions and their usage has not been changed.

Inside the <form> element the basic layout of the form is defined with the <fieldset> and the <legend> elements. The <fieldset> element groups elements inside forms. The <legend> element names the section it is surrounded by [118, 120].

```
<form>
<h1>Online Hotel Reservation</h1>
<fieldset>
<legend>Personal Data</legend>

</fieldset>

<fieldset>
<legend>Reservation Details</legend>

</fieldset>

</form>
```

Listing 15: Form example – Basic Form Structure

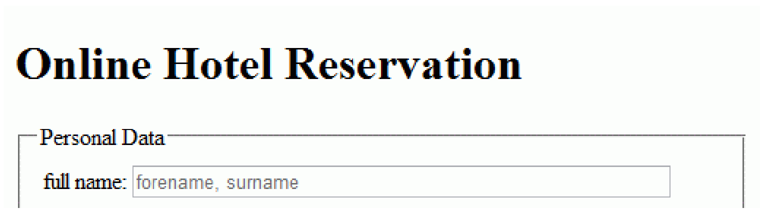
In listing 16, the input fields for the personal data are added to the form:

```
<fieldset>
 <legend>Personal Data</legend>
 <label for="fullname">full name:
  <input type="text" id="fullname" name="fullname" size="50"
  placeholder="forename, surname"
  pattern="([a-zA-Z]+ *)+, ([a-zA-Z]+ *)+" autofocus required>
 </label>
 <label for="salutation">salutation:
  <input type="text" list="salutations" id="salutation" name="salutation">
  <datalist id="salutations">
  <option label="Mr." value="Mr.">
  <option label="Ms." value="Ms.">
  </datalist>
 </label>
 <label for="email">e-mail:
  <input type="email" id="email" name="email" required>
 </label>
 <label for="phone">phone: <input type="tel" id="phone" name="phone">
 </label>
 <label for="hp">homepage: <input type="url" id="hp" name="homepage">
 </label>
        </fieldset>
```

Listing 16: Form example – Input Fields

The <label> element is used to relate a description and an input field. The for attribute of the <label> element references an id attribute of an input element. This enables users to jump to the input field with clicking on the description text. This element is also available in HTML4 [119].

The content of the placeholder attribute of the <input> element is displayed as a hint in an empty text input field. It disappears when the cursor is placed in the field (see figure 3.3) [56].


Online Hotel Reservation

Personal Data

full name: forename, surname

Figure 3.3.: HTML form example rendered in Opera 11 with the value of the placeholder attribute visible in the “full name” input field.

The pattern attribute is for input validation and must be a regular expression for input validation of the entered text. In this case, two sequences of one or more small or capital letters seperated by spaces and these two sequences must be separated by a comma and a space, e.g. “Edwin, van der Sar” would be a valid input (see figure 3.4) [55].

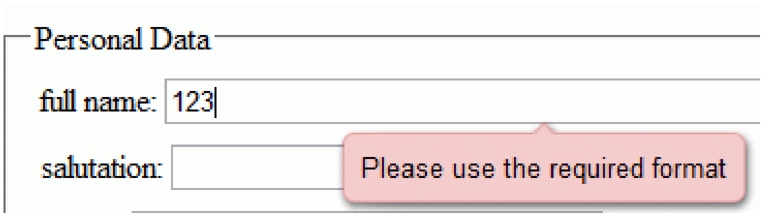


Figure 3.4.: HTML form example rendered in Opera 11 after pressing the submit button with error message indicating that the content of the “full name” input field does not match the defined pattern.

The autofocus attribute is a boolean attribute that focuses a control right after the html document is loaded. There must only be one element with this attribute [22].

The required attribute indicates that the input element must be filled out by the user [59]. The datalist element specifies predefined values for the salutation text field. It is associated with the input element via the list attribute inside the <input> element [46].

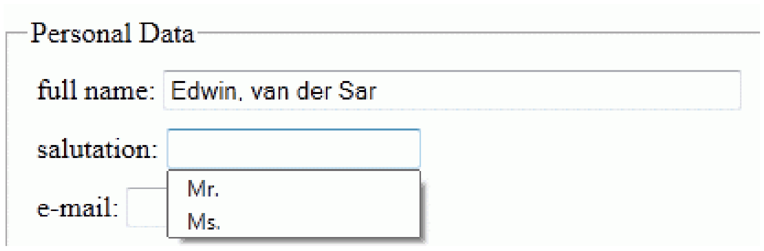


Figure 3.5.: HTML form example rendered in Firefox 4. After clicking on the “salutation” input field the user can chose between the predefined values in the <datalist> or enter another text.

The <input> element with the type value email is used to enter e-mail addresses. It allows browsers to check if the entered text is a valid format for an e-mail address (it does not verify if the e-mail address exists). The multiple attribute for the <input> element could be used to allow entering multiple e-mail addresses [28].

The input type tel is used for phone numbers. There is no validation defined in the specification but it can be done using the pattern attribute. But this information could be used by browsers to adapt the user interface for simple fill in, like displaying a contact list out of an address book. E.g. Safari for iPhone displays a keypad when focusing on an <input type="tel"> element [37]. For URLs, the input type url is used. The entered text must be a valid and absolute URL [63].

Finally, the fields for the reservation details have to be added:

```
<fieldset>
 <legend>Reservation Details</legend>
 <label for="doa">
  date and time of arrival
  <input type="datetime" id="doa" name="doa" required>
 </label>
 <label for="dod">
  date and time of departure
  <input type="datetime" id="dod" name="dod" required>
 </label>
 <label for="beds">
  beds
  <input id="beds" name="beds" type="range" min="1" max="4" step="1">
 </label>
 </fieldset>
```

Listing 17: Form example – reservation details

The input type datetime requires the user to enter a valid date and time (24-hour format) [24].

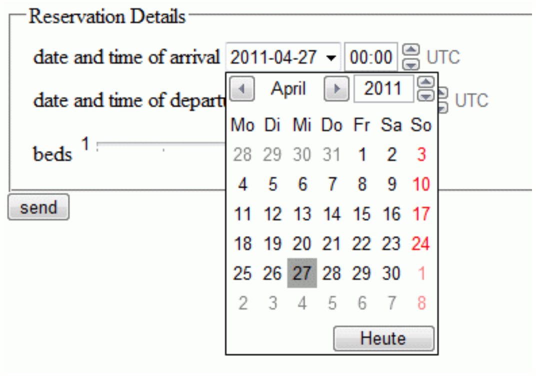


Figure 3.6.: HTML form example rendered in Opera 11. The browser provides a calendar widget for picking a date and a time.

For defining the number of beds required we use the range input type. The attributes min="1"max="4"step="1" define the size of the range (minimum 1, maximum 4) and the granularity of the range (step="1"). Therefore, valid values for this field are 1, 2, 3 and 4 [34]. Figure 3.7 shows how Opera 11 renders the input field.

The complete code for the example with added CSS is available in the appendix A.1.2. All other input types specified in HTML5 can be found in the W3C HTML5 specification under the section named "The input element" (`http://dev.w3.org/html5/spec/Overview.html#the-input-element`).

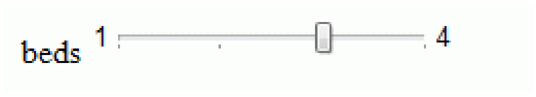


Figure 3.7.: HTML form example rendered in Opera 11. The <input type="range"> element is displayed as a slider. The minimum and maximum values on the left and on the right have been added using CSS.

The new input types are aimed at increasing the usability of web forms and making the HTML code for web forms leaner by allowing browsers to validate and facilitate user input and to provide appropriate user interfaces for each input type. However, the client-side validation of form content through the browser does not make server-side validation unnecessary, because it can easily be bypassed. The main advantage is increased usability: users do not have to send data to the server and wait for a reply for input validation [29].

3.5. Audio and Video

In HTML5, the elements <audio> and <video> enable the embedding of audio and video files in browsers with playback controls and without the need for additional plugins like Adobe Flash. Both are media elements and therefore can use the following attributes that all media elements have:

Attribute	**values**	**explanation**
src	valid, non-empty URL	address of the media file [32]
preload	none, metadata, auto	none: no preload through the browser, metadata: only metadata of the source file is preloaded (e.g. duration and title) [58].
autoplay	boolean	The browser will begin playback automatically [42].
mediagroup	free text (group name)	Used to link media elements. E.g. for playing an audio file synchronized with a video [52].
loop	boolean	The media playback will restart after reaching the end [50].
muted	boolean	The sound of the media file is muted by default but it can be turned on by the user. E.g. for playing video advertisements without sound [53].
controls	boolean	The controls for media playback should be provided by the browser if set. Otherwise, custom controls have to be created [45].

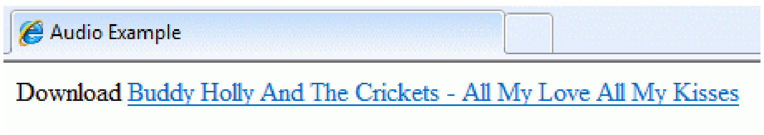
Table 3.1.: HTML5 media elements attributes

Listing 18 shows an example for embedding an audio file with playback controls (controls attribute).

```
<!DOCTYPE html>

<html>
<head>
<title>Audio Example</title>
</head>

<body>

<audio
src="http://ia600101.us.archive.org/7/items/
BuddyHollyAndTheCrickets-01-25/
BuddyHollyAndTheCrickets-AllMyLoveAllMyKisses.ogg"
controls>
        Download
        <a href="http://ia600101.us.archive.org/7/items/
        BuddyHollyAndTheCrickets-01-25/
        BuddyHollyAndTheCrickets-AllMyLoveAllMyKisses.ogg">
        Buddy Holly And The Crickets - All My Love All My Kisses</a>
</audio>

</body>
</html>
```

Listing 18: audio element example

The content surrounded by the <audio> and <video> tags is intended for older browsers that do not support these elements. Newer Browsers ignore the content [41, 62].



Download Buddy Holly And The Crickets - All My Love All My Kisses

Figure 3.8.: Listing 18 displayed in Internet Explorer 8.

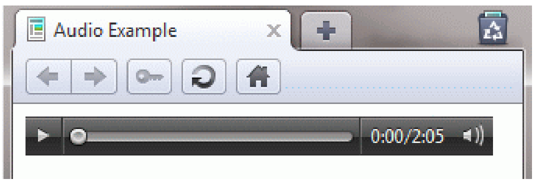


Figure 3.9.: Listing 18 displayed in Opera 11.

In addition, it is possible to specify alternative media sources by placing multiple <source> elements inside without using the src attribute. The browser can then pick a <source> element that it supports. This can be used to include multiple versions of the same media file with different encodings to ensure (backwards-) compatibility. The type attribute should be used to specify the MIME type of the media file and the codec to enable the browser to ensure that it can play it. This is important because the W3C HTML5 specification currently does not specify codecs that have to be supported by a HTML5 compliant browser (see section 7.1) [61].

```
<audio controls>
 <source src="AllMyLoveAllMyKisses.ogg"
         type="audio/ogg; codecs=vorbis">
 <source src="AllMyLoveAllMyKisses.mp3"
         type="audio/mpeg; codecs=mp4a.E1">
</audio>
```

Listing 19: audio element example with multiple sources

The <video> element has three additional attributes compared to the <audio> element. These are:

Attribute	**values**	**explanation**
poster	valid, non-empty URL	Address of an image file the browser shows when the video playback has not started yet [57]
width	valid non-negative integer	Width of the video relative to the nominal direction of the output medium in CSS pixels[7] [25]
height	valid non-negative integer	Height of the video relative to the nominal direction of the output medium in CSS pixels [25]

Table 3.2.: Specific attributes of the video element

Otherwise, the usage of the <video> element is the same as of the <audio> element:

```
<!DOCTYPE html>

<html>
<head>
<title>Video Example</title>
</head>

<body>

<video src="http://media.tinyvid.tv/t34ujqkkgkrq.ogg" width="300"
       poster="Play-Normal-icon.png"
       controls>
        Please download the video
        <a href="http://media.tinyvid.tv/t34ujqkkgkrq.ogg">here</a>.
</video>

</body>
</html>
```

Listing 20: video element example

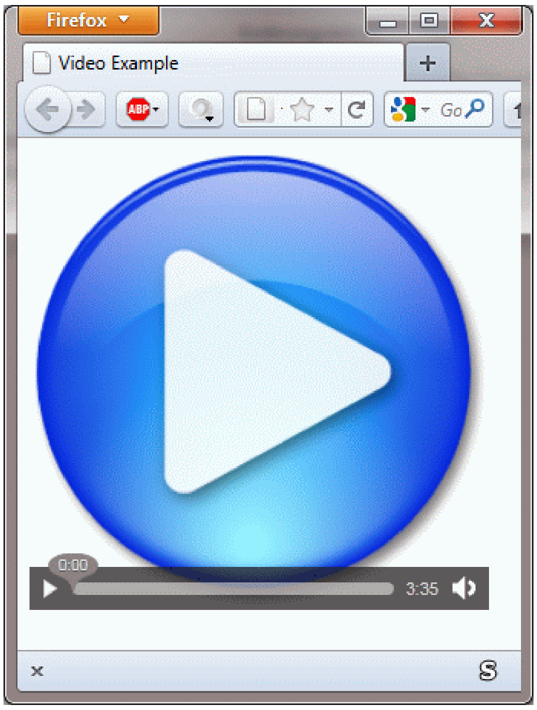


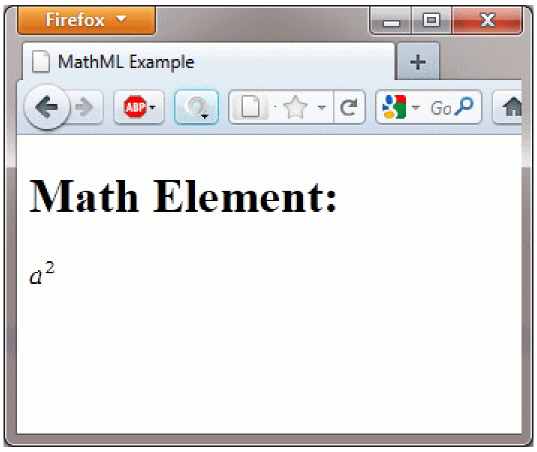
Figure 3.10.: HTML video element example in Firefox 4 with poster image and playback controls.

3.6. MathML

The <math> element is used to embed the Mathematical Markup Language (MathML)in HTML5. This way, mathematical notations can be displayed in HTML documents without using images. The <math> element can also be used with XHTML1 but it is necessary to add an xmlns attribute for referencing the MathML namespace [33].

```
<!DOCTYPE html>

<html>
<head>
<title>MathML Example</title>
</head>

<body>

<h1>Math Element:</h1>
<math>
              <msup>
                <mi>a</mi>
                <mn>2</mn>
        </msup>
</math>

</body>
</html>
```

Listing 21: math element example



Math Element:

a^2

Figure 3.11.: Math element example in Firefox 4

3.7. Canvas

The <canvas> element provides an area for rendering visual images on bitmaps on the fly. Currently, there are two APIs available for drawing on canvas: 2D and webGL, a 3D context which is similar to OpenGL [147]. The Canvas 2D context that is introduced in this section is specified by the W3C in a separate specification available under `http://dev.w3.org/html5/2dcontext/`. The attributes width and height of the <canvas> element can be used to specify the size of the canvas in pixels. The default size is 300 pixels by 150 pixels [44].The content of the element is intended for browsers that do not support the element.

The Canvas API can among other things be used to draw images, create animations and interactive applications like games without requiring a browser plugin (See section 6). Listing 22 shows how to create a simple text animation. The result is a blue rectangle in which the text “HTML5” scrolls from left to right (figure 3.12).

```
<!DOCTYPE html>

<html>
<head>
<title>Canvas Example</title>

<script>
x=0;
r=1;
//call function drawLogo every 5 milliseconds
setInterval(drawLogo,5);

function drawLogo() {
        //get the canvas context
        var context = document.querySelector('canvas').getContext('2d');
        //Set fill color to blue
        context.fillStyle="#0004FF";
        //draw rectangle
        context.fillRect(0, 10, 300, 50);
        //Set font, font weight = 600 (standard 400)
        //font height = 28px, font name = Arial
        context.font = '600 30px Arial';
        //Set fill color to white
        context.fillStyle="#FFFFFF";
        //Write Text x=0, y=45, width of the text (optional)
        context.fillText("HTML5", 0, 45);

        //if r=1, move context right
        if (r == 1) {
                context.translate(1,0);
                x+=1;
        }
        //else: move context left
        else {
                context.translate(-1,0);
                x-=1;
        }
        //if text reaches end of canvas, change scroll direction
        if (x == 200) {
                r=0;
        }
        if (x == 0) {
                r=1;
        }
}
</script>

</head>

<body>

<canvas style="border: 1px solid black;" width="300" height="70">
<p>HTML5 scrolling text with Canvas.</p>
</canvas>

</body>
</html>
```

Listing 22: Scrolling text with canvas

More information on the usage of the Canvas API can be found in the Canvas specification, or on websites like `http://www.canvasdemos.com`. Section 6 also references to examples of the Canvas API in use. A Canvas tutorial from the Mozilla Corporation is available under `https://developer.mozilla.org/en/Canvas_tutorial`.

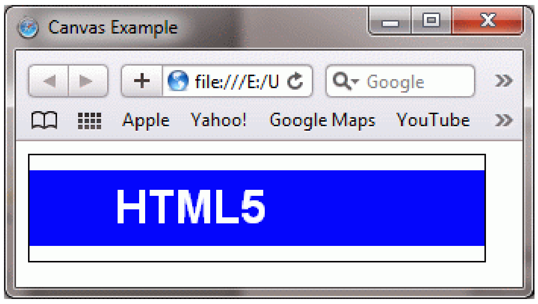


Figure 3.12.: Canvas 2D API: Listing 22 rendered in Safari 5

3.8. SVG

Scalable Vector Graphics (SVG) is an XML-based language recommended by the W3C to describe two-dimensional vector graphics. It can be embedded in HTML using the <svg> element. Because it is vector based, images can easily be scaled without quality loss[8]. A well-known open-source WYSIWYG SVG editor is Inkscape (`http://inkscape.org`). The <svg> element allows memory space-saving inclusion of graphics and animations in web sites.

Since SVG is an XML-based language it is a retained-mode API, which means that the current state of all drawn objects is retained in the DOM tree which allows better user interactivity. In contrast to that, the Canvas API is an immediate-mode API [77]. A detailed comparison between Canvas and SVG by Opera can be found here: `http://dev.opera.com/articles/view/svg-or-canvas-choosing-between-the-two/`.

To draw the same animation shown in the Canvas example 22 with SVG, the HTML5/SVG code in example 23 can be used. JavaScript is not needed to create the animation, because the <animateMotion> element of SVG inside the <text> element can be used to animate it. The values attribute specifies the path on which the <text> element moves, relative to its original coordinates x=0 and y=45 [134].

```
<svg>
 <rect x="0" y="10" width="300" height="50" style="fill:blue;" />
 <text x="0"  y="45" style="font-family: Arial;
                                  font-size: 30px;
                                  fill: #FFFFFF;
                                  font-weight: 600;" width="80">
        HTML5
 <animateMotion dur="5s" values="0,0;200,0;0,0" repeatCount="indefinite" />
 </text>
</svg>
```

Listing 23: SVG element example

[8]For more information on SVG, please visit `http://www.w3schools.com/svg`

Like the <math> element, the <svg> element can also be used in XHTML1, but then also needs the xlmns attribute with a reference to the SVG namespace.

3.9. Application Cache

To allow using web applications without a network connection or continue using them with interrupted or poor connections the cache manifest is introduced in HTML5. The cache manifest is a text file of the MIME type ”text/cache-manifest” with the file extension “.manifest” that contains a list of files the web application needs to work offline. This is especially useful for using web applications on mobile devices on the move to allow sustained work during times with no network connection [21].

A sample manifest file is displayed in listing 24.

```
CACHE MANIFEST
# version 1

CACHE:
main.css
index.html

FALLBACK:
online.css offline.css
/ /offline.html

NETWORK:
online.html
```

Listing 24: Cache manifest file

The hash sign is used to mark comment lines. The files the web browser needs to cache for offline functionality are listed under the section CACHE.

The FALLBACK section is used to define online and offline versions of files. First, the name of the file that should be used with an established network connection is specified seperated by a space from the name of the file that should be used instead if there is no network connection. In the previous example, the online.css stylesheet will be replaced by the offline.css stylesheet when there is no connection. The offline file (offline.css) will also be cached and does not have to be listed separately under the CACHE section. The line / /offline .html means that any page that can not be accessed over the network and is not in the cache is redirected to the “offline.html” page.

The NETWORK header precedes files that should only be available through the network and therefore not be cached. It is optional and the default value is “*” which means that every file that is not listed under the CACHE category must be accessed over the network.

The order of the sections is irrelevant. It is also possible to reopen sections, i.e. to have multiple section headers of the same type [43].

The files that are cached are only updated by the browser if the content of the manifest file changes, not if the content of the files in the CACHE section changes. This is why it is advisable to add a version number or a date/time stamp in a comment line in the manifest that is changed to trigger a cache update [78].

The cache manifest file is specified in the manifest attribute of the <html> element in each HTML document of the web application [51]:

```
<!DOCTYPE html>
<html lang="en" manifest="test.manifest">
```

Listing 25: HTML manifest attribute

3.10. Drag and Drop

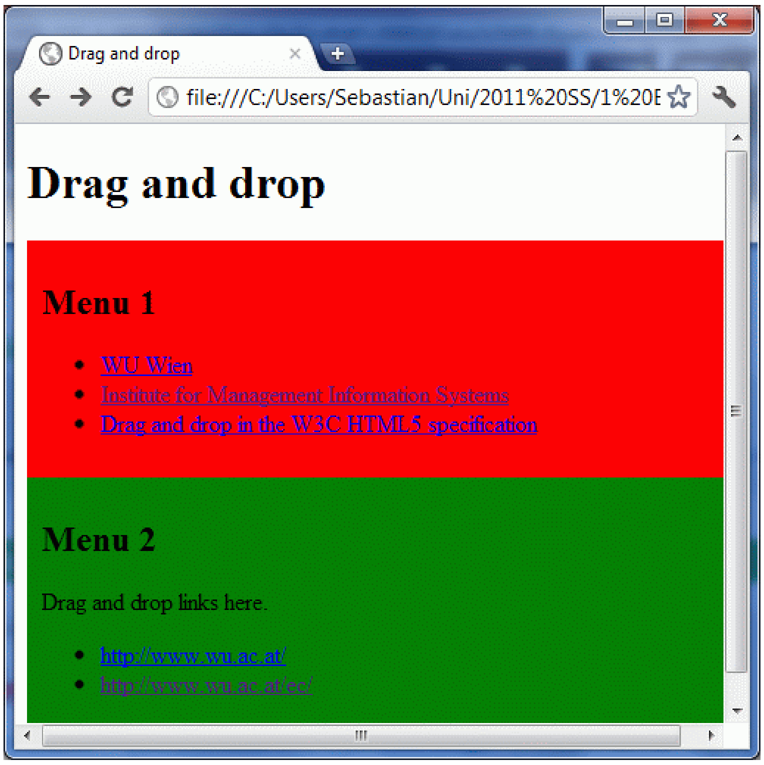
The Drag and Drop API is based on the first implementation of drag and drop in the Internet Explorer in 1995 and was reverse engineered for HTML5 by Ian Hickson [16]. It allows drag and drop inside the browser on a web site and from the browser in other applications. All elements can be made draggable by setting the attribute draggable to "true". Links and images are draggable by default. This can be used to drag an image from a browser window into another program or to drag links from one browser window into another to open the link in that window [26].

In the example provided in listing 26, users can define their own navigation menu.

```
<!DOCTYPE html>

<html>
<head>
<title>Drag and drop</title>
</head>

<body>
<h1>Drag and drop</h1>
<nav id="menu1"
     style="height: 100%; width: 100%; background-color: red;padding: 10px;"
     ondragstart="this.style.background = 'orange';"
     ondragend="this.style.background = 'red';">
 <h1>Menu 1</h1>
 <ul>
 <li><a href="http://www.wu.ac.at">
 WU Wien
 </a></li>
 <li><a href="http://www.wu.ac.at/ec/">
 Institute for Management Information Systems
 </a></li>
 <li><a href="http://dev.w3.org/html5/spec/dnd.html#dnd">
 Drag and drop in the W3C HTML5 specification
 </a></li>
 </ul>
</nav>

<section
   style="height: 100%; width: 100%; background-color: green;padding: 10px;"
   dropzone="copy s:URL"
   ondrop="dropHandler(event);this.style.background = 'green';"
   ondragover="this.style.background = 'yellow';return false;"
   ondragenter="return false;">
 <h2>Menu 2</h2>
 <p>Drag and drop links here.</p>
 <ul id="dropzone">
 </ul>
</section>

<script type="text/javascript">
 //get nav element of menu2
 var drop = document.getElementById('dropzone');


 function dropHandler(event) {
  if (event.dataTransfer.getData('URL')) {
   drop.innerHTML += '<li><a href="'
                  + event.dataTransfer.getData('URL') + '">'
                  + event.dataTransfer.getData('URL') + '</a></li>';
  }
  // Stops Firefox from redirecting.
  if (event.preventDefault) {
   event.preventDefault();
  }
 }
</script>

</body>
</html>
```

Listing 26: Drag and drop example

Some CSS was added with the style attribute to add colors to the menus. In this example, links can be dragged from Menu 1 or from other web pages to Menu 2. When a link is dropped on Menu 2, a new link is created to the URL the dropped link refers to.



Drag and drop

Menu 1

- WU Wien
- Institute for Management Information Systems
- Drag and drop in the W3C HTML5 specification

Menu 2

Drag and drop links here.

- http://www.wu.ac.at/
- http://www.wu.ac.at/ec/

Figure 3.13.: Drag and drop example in Chrome. Links can be dragged from menu 1 to menu 2

Menu 2 is the "drop zone". It uses the attribute dropzone to specify the action that should be performed on items that are dropped in it and what kind of data can be dropped. In this example the action is "copy" and the item kind must be URLs formated as a string (s:URL). It is also possible to define custom data types [27].

Several event handlers are used in the code for the <nav> elements of menu 1 and menu 2 that fire at different moments:

- ondragstart - when the user begins to drag an item
- ondragend - when the drag stops
- ondrop - when the dragged item is dropped
- ondragenter - when item enters the element
- ondragover - when an item is dragged over the element

In listing 26 ondragstart, ondragend and ondragover are used to change the background color of the menus. The ondrag event handler of Menu 2 triggers the dropHandler function that adds a link to the menu, if the dropped item is an URL.

The following code prevents browsers from redirecting to the URL of the dropped link [96]:

```
                if (event.preventDefault) {
                        event.preventDefault();
                }
```

Listing 27: Drag and drop: javascript code for preventing browser redirect

3.11. Web Storage API

The Web Storage API is intended to replace Cookies for local storing of user data of web applications. It is specified by the W3C in a separate document available under `http://dev.w3.org/html5/webstorage/` and was once part of the HTML5 specification. It allows assigning values to keys in a storage object. The keys can either be saved for the current session in one browser window in a sessionStorage object or for an unlimited lifetime for the whole domain of the website available to all browser windows in a localStorage object. Both have access to the same methods as they are both storage objects [71].

For demonstration, the drag and drop example from section 3.10, listing 26 is extended so that the links that have been dropped in menu 2 are saved. The current state of the menu after each drop can be saved by inserting the following code in line 50:

```
localStorage.setItem('SavedMenu2', drop.innerHTML);
```

This example uses the localStorage object because the menu should also be available to the user on his next visit to the website. With the setItem method, we set a key named “SavedMenu2” with the value of the current html content of the menu2 <nav> element that is stored in the variable “drop” [70, 72].

Then it has to be checked at the beginning of the script, if a saved menu is already available and if that is true it has to be loaded:

```
        if (localStorage && localStorage.getItem('SavedMenu2')) {
                drop.innerHTML = localStorage.getItem('SavedMenu2');
                alert('Saved Menu 2 loaded!');
        }
```

Listing 28: Web Storage: localStorage Object

The if statement checks for a localStorage object for this domain and if the object contains an Item with the key SavedMenu2 with the getItem method [72]. If both are existing, the value of the key is added to the content of the menu2 <nav> element and a message shows the user that menu 2 has been loaded.

To delete an item in a storage object the method removeItem, e.g. localStorage.removeItem('menu2'), is used. All items in the object can be deleted using the clear() method [72]. So to allow the user to delete the custom menu 2, we can add the following link right after the menu:

```
<a href="javascript:if(localStorage) { localStorage.clear(); location.reload(true);};'
```

The whole code of this example can be found in appendix A.1.3.

3.12. Web Workers API

javascript does not allow multiple scripts running at the same time. This means that javascript code that has a high resource consumption will hang a browser and prevent the execution of any other scripts on a web site. The Web Workers API enables the execution of javascript code that takes long to compute in the background, meaning that other smaller scripts can still be executed by the browser. Scripts running in the background are referred to as “Workers”. The specification of the API is available in a separate document under `http://dev.w3.org/html5/workers/` [67].

Usage scenarios for the Web Workers API include keeping the user interface of a web application responsive during database updates or long calculations and updating page content in the background [68].

A Worker can be initialized with the following javascript code:

```
var worker = new Worker('script.js')
```

As a parameter, the constructor of the Worker requires an URL to a javascript file that will be executed by the Worker.

```html
<section>
        <h1>Countdown</h1>
        <p>Countdown executed by Worker: <span id="countdown"></span></p>

        <script>
                for (var i = 1000000000; i > 0; i--) {
                        document.querySelector('#countdown').innerHTML = i;
                }
        </script>
</section>
```

Listing 29: Countdown example

Listing 29 shows the code for a countdown from one billion to zero in a for loop. During the countdown, no other user input is possible and most browsers will become unresponsive until the loop is completed.

If the countdown is executed by a Worker, other scripts can be executed during the countdown. Listing 30 shows the adapted code.

```html
<section>
 <h1>Countdown</h1>
  <p>Countdown executed by Worker: <span id="countdown"></span></p>

  <script type="text/javascript">
   //create worker
   var worker = new Worker('countdown.js');

   //get current countdown number from worker
   worker.onmessage = function (event) {
     document.querySelector('#countdown').innerHTML = event.data;
   }
  </script>
</section>
```

Listing 30: Countdown example executed by Worker

```javascript
for (var i = 1000000000; i > 0; i--) {
    postMessage(i);
}
```

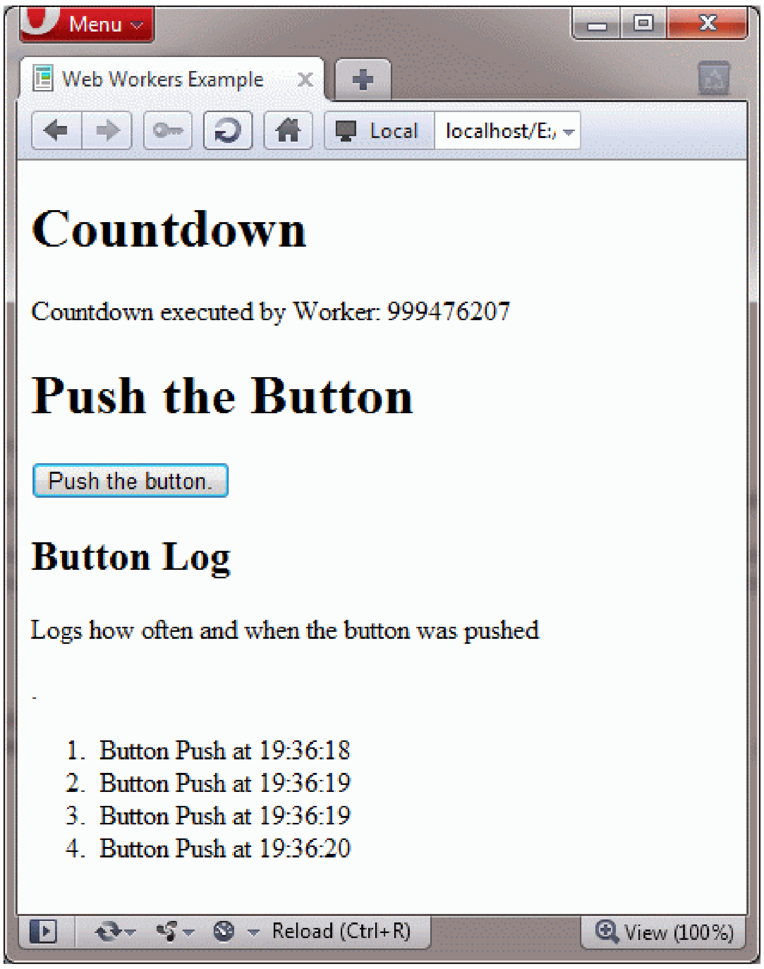
Listing 31: countdown.js

The postmessage method of a Worker is used to communicate with the script that created the worker. The onmessage event handler is used to access messages sent by the Worker [66].

If the countdown is executed by a Worker, other scripts on the same web page can be executed during the countdown. Appendix A.1.4 contains the complete code for an HTML document that uses a Worker to start a countdown and contains a button that executes a small script when pressed.

To stop a worker the terminate method is used [66]:

```
worker.terminate();
```



Countdown

Countdown executed by Worker: 999476207

Push the Button

Button Log

Logs how often and when the button was pushed

1. Button Push at 19:36:18
2. Button Push at 19:36:19
3. Button Push at 19:36:19
4. Button Push at 19:36:20

Figure 3.14.: Web Workers Example from Appendix A.1.4 rendered in Opera 11.

3.13. Web Sockets API

Communication between a web server and a wep page is usually one-directional. Only a web page can send a request to a server. This means that a web server can not send data to a web page that was not previously requested by it and therefore can not update data on the web page. For web applications, a two-way communication is often more practical or required, e.g. for pop-up notifications. A method called Polling can be used to simulate a two-way communication, which is used in AJAX (Asynchronous javascript and XML)[9]. Short Polling means that a web page repeatedly requests data and Long Polling means that a connection initialized by a request from a web page is kept open. The drawback of both methods is high resource consumption through increased cpu load on the server and increased network traffic [80, 146].

The Web Sockets API addresses these shortcomings and enables bidirectional communication between a web page that uses the API and a web server through a full-duplex single TCP socket connection [146].The specification can be found under `http://dev.w3.org/html5/websockets/`.

To create a WebSocket, the following code is used:

```
ws = new WebSocket(ws://example.org);
```

ws://example.org represents the URL to which the WebSocket connects. ws:// is the prefix for a web socket connection using port 81 and wss:// is for secure web socket connections over port 851. As an optional second parameter the WebSocket constructor can take one or more subprotocols. The server has to select one of the supplied protocols before a connection can be established [69].

There are three event handlers for the WebSocket interface:

1. open - when a connection is opened
2. onmessage - when a message is received over the connection
3. onclose - when the connection is closed

To send a message over a WebSocket connection, the send method is used:

```
ws.send("Hello World!");
```

A connection can be closed using the close method:

```
ws.close();
```

To use the WebSocket API, a properly configured web server is required. For demonstration purposes, the URL echo.websocket.org is used in listing 32 to establish a WebSocket connection. The server sends back each message it receives over a WebSocket connection. The code is a simplified version of the WebSockets example provided under `http://websocket.org/echo.html`.

After a connection is established (ws.onopen) the message "Hello World!" is sent to the server with the send method. When a response from the server is received (ws.onmessage) it is written inside the element with the id attribute set to "output" and the connection is closed again.

[9]For more information, please visit `http://www.w3schools.com/ajax`

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>WebSocket Test</title>

<script language="javascript" type="text/javascript">
        ws = new WebSocket('ws://echo.websocket.org');

        ws.onopen = function(event) {
                ws.send('Hello World!');
        }

        ws.onmessage = function(event) {
                document.querySelector('#output').innerHTML = event.data;
                ws.close();
        }
</script>

</head>

<body>

<h1>WebSocket Test - Server Response</h1>
<p id="output"></p>

</body>
</html>
```

Listing 32: Web Socket example

4. Advantages for Mobile Devices

4.1. Less need for additional plugins

The new features of HTML5 make it possible to provide user friendly and multimedia rich web sites without the need for additional third-party browser plugins, e.g. for video or audio playback or for animations. These plugins, like Adobe Flash or Microsoft Silverlight, are often not available for all mobile operating systems. As of June 2011, there is no Flash Plugin for the Apple iOS and Windows Phone 7. These plugins are also an additional security problem because attackers exploit vulnerabilities in the software.

4.2. Additional elements for structuring content

HTML5 introduces new elements to structure the content of a web site and to create an outline. This allows browsers to understand the structure of a website and enables them to display a table of contents and makes it possible to facilitate navigation on devices with smaller screens, e.g. with a button that allows flipping through all navigation menus of a website.

4.3. Offline Support

The Application Cache of HTML5 allows users to use web applications when no network connection is available. This is especially useful for mobile devices that are often used on the go and therefore have a higher risk of losing network connection than stationary devices.

4.4. Canvas and SVG

Canvas and SVG allow native support for rendering 2D and 3D graphics on mobile devices without the need for additional browser plugins. This enables web designers to create multimedia applications like browser games with open web standards that run on all devices with a web browser that supports HTML5. In addition, the embedding of SVG allows space-saving inclusion of graphics which reduces loading times of web sites.

4.5. New Form Features

The new HTML form features that allow input validation by the browser reduces the size and complexity of creating web forms through making javascript client-side validation and input interfaces (like calender widgets) unnecessary. This is especially an advantage for mobile devices because it makes the HTML code leaner and decreases the size of the HTML document therefore decreasing the loading times. Also, a local input validation by the browser will in most cases ensure valid form content and enable the user to successfully submit the content with the first try. In addition, mobile browsers can have customized user interfaces for input fields that are optimized for the device, e.g. touch screen friendly. The BlackBerry web browser has special input interfaces for the input types date and color that are optimized for touch screens since 2010 [95].

4.6. Opportunity for cross-platform mobile applications

HTML5 can be used to build mobile web applications that are comparable in their functionality to native mobile applications in combination with other Open Web Platform technologies. E.g. the Geolocation API allows web applications to access location information from various sources. It could help to break the tight control some manufacturers have of the application distribution for their mobile platforms.

4.7. Frameworks for developing native mobile applications

There are several open source frameworks available that allow the creation of native mobile application for various mobile operating systems with HTML5, javascript and CSS. A selection of them is listed in table 4.1. Section 4.7.1 gives a short introduction to the PhoneGap framework and a basic example on how to develop mobile applications for the mobile operating system Android with it.

Framework	**Licence**	**Homepage**
Jo	OpenBSD	`http://joapp.com`
Open Mobile	GNU LGPL	`http://www.openmobileis.org`
PhoneGap	modified BSD or MIT (2008)	`http://www.phonegap.com`
QuickConnect	MIT	`http://www.quickconnectfamily.org`
Rhodes	MIT	`http://rhomobile.com/products/rhodes`
Sencha Touch	GNU GPL	`http://www.sencha.com`
Titanium Plattform	Apache 2	`http://www.appcelerator.com`

Table 4.1.: Mobile Application Development Frameworks based on Web Technologies

4.7.1. Creating mobile applications with HTML5 and PhoneGap

PhoneGap is a free open source framework that allows developing native applications for the mobile operating systems iOS, Android, Blackberry, webOS and Symbian WRT with HTML5, javascript and CSS3. Installation and setup instructions for all supported mobile plattforms are available on the PhoneGap homepage: `http://www.phonegap.com/start`. Further documentation, including the API reference, can be accessed on `http://www.phonegap.com/docs`.

There is also an open book available called "Building iPhone Apps with HTML, CSS, and javascript" by Jonathan Stark that describes how to use PhoneGap to create iOS applications in chapter 7 that can be viewed on `http://ofps.oreilly.com/titles/9780596805784/chapPhoneGap.html`.

Using PhoneGap to create Android Applications

To create Android applications with PhoneGap, the following software is required [90]:

- Eclipse Classic IDE:
  `http://www.eclipse.org/downloads/`

  Android Development Tools (ADT) Plugin for Eclipse:
  `http://developer.android.com/sdk/eclipse-adt.html#installing`
- Android SDK:
  `http://developer.android.com/sdk/index.html`

  Requires Java SE Development Kit (JDK):
  `http://www.oracle.com/technetwork/java/javase/downloads/`
- Latest version of PhoneGap (in this example 0.9.5.):
  `http://www.phonegap.com/`

To create a new Android application, Eclipse must be opened, a workspace directory chosen and a new Android project must be created: File - New - Project... - Android - Android Project.

In the Project setup a Project name must be entered, e.g. "HelloWorldAndroid" and under Build Target an Android version must be selected, in this example, Android 2.2. Finally, the application name (HelloWorld) and a package name (com.phonegap.helloworld) must be entered. "Create Activity" must be checked and "App" must be entered in the text field. The "finish" button creates the project.

The Package Explorer in Eclipse should now show the new Android project directory. Two new folders must be created in this directory by right-clicking on it in the Package Explorer and choosing new - Folder in the menu. This opens a new window for creating folders in Eclipse. “libs” must be entered in the text field next to “Folder name:” and finish can be pressed. Another folder in the “assets” folder named “www” must be created.

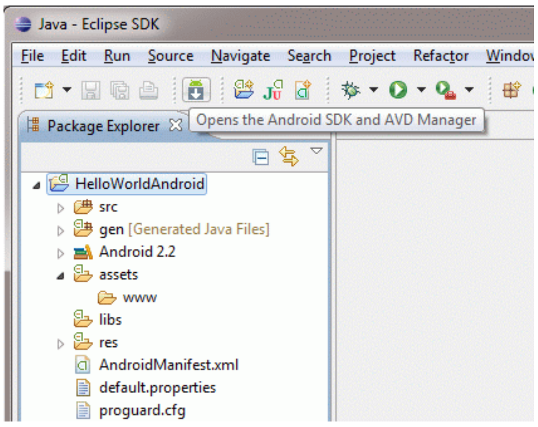


Figure 4.1.: The project directory in Eclipse should look like this.

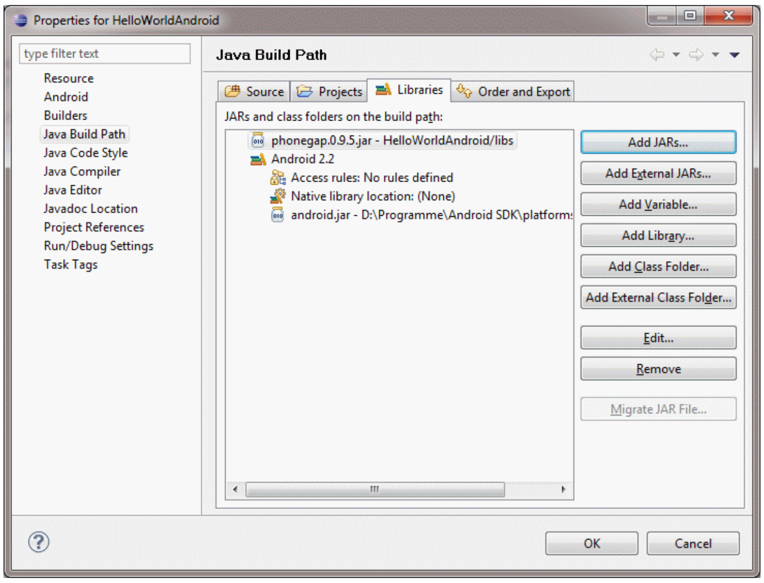
Now two files from the PhoneGap directory need to be copied to the Android Project directory. The project directory is located in the workspace directory and named like the project name, in this example “HelloWorldAndroid”. First the file phonegap.0.9.5.jar from `.../PhoneGap-0.9.5/Android/` must be copied to `.../workspace/HelloWorldAndroid/libs/`. Then, phonegap.0.9.5.js from `.../PhoneGap-0.9.5/Android` to `.../workspace/HelloWorldAndroid/assets/www/`.

To be able to see the files in the Package Explorer the project in Eclipse must be refreshed by pressing F5[1]. The “phonegap.0.9.5.jar” must be added to the Build Paths of the project. To do this, one has to right click on the “libs” folder and Build Paths - Configure Build Paths must be selected. In the new window, we select the Libraries tab, press on the Add JARs... button on the right and select the phonegap.0.9.5.jar from the “libs” folder of the project directory.

Now an HTML document in the “www”-folder can be created by right clicking on it in the Package Explorer and choosing New - File from the menu. As a filename, “index.html” must be entered. The HTML document can be edited by a right click on it in the Package Explorer and then “open with - text editor” must be selected from the menu. For this example the code of the basic HTML5 document from section 3.1 is pasted in Eclipse. Then it can be saved.

Next, the Java file named “App.java” (as specified in the project setup under “Create Activity”) in the src-folder must be added. A double-click on it in the Package Explorer opens it in Eclipse. The PhoneGap package must be imported, the super class of App must be changed to DroidGap and the setContentView() method replaced with super.loadUrl(). The finished code should look like in listing 33.

[1]It may be necessary to refresh the project more than once until all of the files are shown.



JARs and class folders on the build path:

- phonegap.0.9.5.jar - HelloWorldAndroid/libs
- Android 2.2
  - Access rules: No rules defined
  - Native library location: (None)
  - android.jar - D:\Programme\Android SDK\platform:

Figure 4.2.: The libraries in the build path for the android project.

```java
package com.phonegap.helloworld;

import android.app.Activity;
import android.os.Bundle;
import com.phonegap.*;

public class App extends DroidGap {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        super.loadUrl("file:///android_asset/www/index.html");
    }
}
```

Listing 33: PhoneGap: App.java

Now permissions for PhoneGap must be added to the “AndroidManifest.xml” file located in the project directory. To do this, it can be opened with the text editor, just like the “index.html” file. The permissions need to be added inside the manifest element. Also, an attribute must be added to the activity element. The finished XML document can be found in appendix A.1.5.

Now the project can be run as Android Application by right clicking on the project directory in the Package Explorer and selecting “Run As - Android Application”.

If there is no Android Virtual Device (AVD) available the user will be prompted to create one by the Android SDK and AVD Manager via the “New” button on the right. A name must be entered for the virtual device. For this example, "Android 2.2. - API Level 8" as target should be selected and then "Create AVD" can be pressed. Now the AVD should launch and open our hello world application (this may take some time).

The Android Package file (APK) can be found in the project directory in the "bin" folder and is named like the project name, in this example "[workspaceDirectory]/HelloWorldAndroid/bin/Hel
To run it on an Android device it must be copied on the device, opened (with a file manager for Android, like File Expert) and installed. To be able to install an application that is not from the Android Market, the Android application settings have to be changed to allow installation of applications from unknown sources [79].

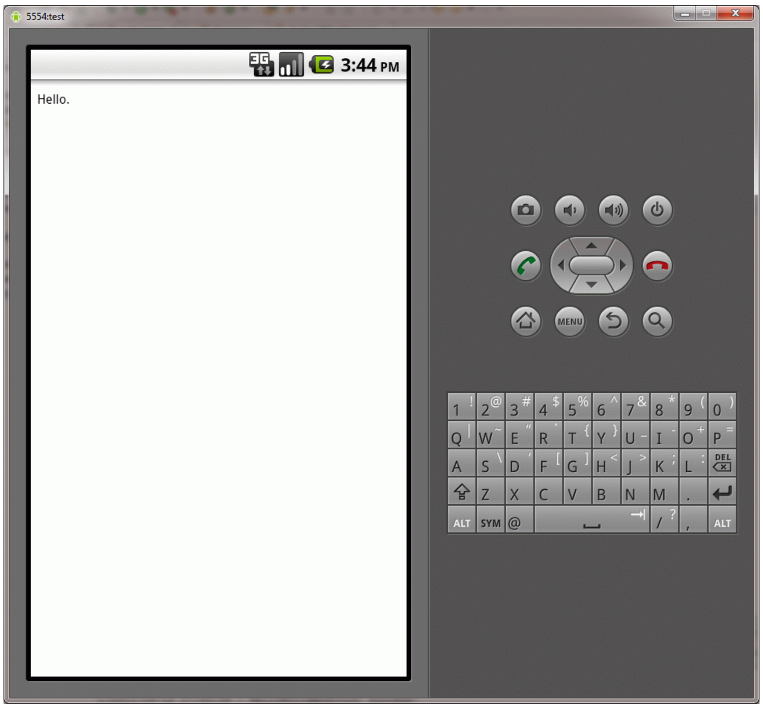


Figure 4.3.: The PhoneGap example application shown in the Android Virtual Device

PhoneGap Build

PhoneGap also offers an online service for creating native mobile applications out of web applications named PhoneGap Build that is currently only available in a closed beta test. It allows users to upload a web application created with HTML, CSS and javascript files and returns download links for the converted mobile application. To take part in the beta test, users have to apply via a web form (`https://build.phonegap.com/`). The service will be available free of charge for open source applications [91].

5. Browsersupport

Some parts of the HTML5 specification have already been implemented in web browsers for experimental reasons or they were already implemented and are now part of the specification. However, these implementations are mostly incomplete because the specification is not yet finished. Also the level of support for a technology is different from browser to browser. There are also several web sites that show the current level of implementation of the HTML5 specification for the most popular web browsers. A choice of them are:

- `http://caniuse.com`
- `http://html5readiness.com`
- `http://html5test.com`

This section provides a broad overview of the current state of implementation of the more stable features presented in section 3 in five popular web browsers.

5.1. Internet Explorer

The Internet Explorer is developed by the Microsoft Corporation for their Operating System Windows. It is the most widely used browser on the web. Therefore, there are still some web pages and web applications that can only be viewed correctly or are only accessible through Internet Explorer because in previous versions of the browser, web standards like CSS2 were not supported correctly. This has changed with Version 7 of the Internet Explorer, in part also because browsers with better support for open web standards, like Firefox and Opera, were gaining users.

The dominance of Internet Explorer on the Internet was also reduced by a ruling of the General Court of the European Union, that led Microsoft to allow users to choose third-party browsers during a Windows installation. Previously, the Internet Explorer was preinstalled and Windows Update could only be accessed through it [9].

Microsoft has created a Website named "Internet Explorer Testdrive" where users can view “HTML5 Demos” that show some of the capabilities of Internet Explorer 9. It is available under `http://ie.microsoft.com/testdrive/`. These demos are however mostly showing features that are not part of the HTML5 specification but of other W3C specifications, like CSS3 and the Geolocation API, and some features Microsoft introduced that are not part of any web standard from the W3C. Therefore they do not work very well in other browsers.

5.1.1. Internet Explorer 9

The current desktop release, Internet Explorer 9, was released in march 2011 by Microsoft [82].It can be installed for free on the operating systems Windows Server 2008, Microsoft Windows Vista and Microsoft Windows 7 [81].The official download page on the Microsoft homepage is `http://windows.microsoft.com/en-US/internet-explorer/downloads/ie-9/worldwide-languages`.

The Internet Explorer 9 has a "compatibility mode" that allows it to render web sites correctly that are optimized for the parsing engine of older versions of the browser - especially Internet Explorer 6. These pages use syntax that previous versions of the browser rendered not according to the W3C specifications or syntax that is only understandable by the Internet Explorer.

Of all current versions of the major desktop browsers the Internet Explorer 9 has the least support for the HTML5 specification. It supports Canvas, Video, Audio, Drag&Drop and the Web Storage API. It does not support the majority of the new form features of HTML5, the application cache, the Web Workers API, the Web Sockets API and the embedding of MathML in HTML documents [6].

5.1.2. Internet Explorer Mobile 7

Internet Explorer Mobile 7 is the current mobile variant of the Internet Explorer, usable only on Windows Phone 7 OS. It is based on Internet Explorer 7 technology and therefore does not support any of the new features in the HTML5 specification.

The next version, Internet Explorer Mobile 9, which should have support for some of the new HTML5 features, should be released in the second half of 2011 [74].

5.2. Firefox

The Firefox browser is developed by the Mozilla Corporation. All software of the corporation is released as open source freeware under the Mozilla Public License (MPL) [86]. HTML5 Demos from the Mozilla Corporation and Demos submitted by users can be viewed on `https://demos.mozilla.org`. At the beginning of some demos, the web browsers that can display it are listed.

5.2.1. Firefox 4

Firefox 4 is the latest desktop release of the browser and available since March 22, 2011 for Windows, Mac OS X and Linux. It can be downloaded from `http://www.mozilla.com/firefox/fx/`.

Firefox 4 supports most of the HTML5 specification but only some of the new form features [6]. It supports the Web Workers API, the Web Storage API but has disabled support for the Web Sockets API per default, because of security concerns [15, 6].

5.2.2. Firefox 4 for Android and Maemo

A Firefox 4 version optimized for mobile devices is available for the OS Android and Maemo (Linux-based mobile OS by Nokia) and can be downloaded from `http://www.mozilla.com/en-US/m/`. Versions for Blackberry and Symbian are to be released later in 2011. According to a blog post on the Mozilla Mobile Blog, a version for the iPhone/iOS is highly unlikely due to the restrictions of the platform [85].

5.3. Opera

The Opera Web Browser is developed by Opera Software ASA. Opera Software develops web browsers for personal computers, mobile phones and other devices like TVs and game consoles (Nintendo Wii, Nintendo DS) [87].The End-User License Agreements (EULA) for Opera browsers can be found on the Opera homepage (`http://www.opera.com/eula/browser/`).

5.3.1. Opera 11

Opera 11 was released on 17 December, 2010 for Windows, Mac OS X, Linux, FreeBSD and Solaris SPARC and is available for download under `http://www.opera.com/browser/`.

It supports most of the HTML5 specification, notably almost all HTML5 form features, because these are based on the XForms specification that was created by Opera. There is no support for the <svg> and the <math> element. The Web Storage API and the Web Workers API are supported. Like in Firefox 4, the Web Sockets API support is disabled by default because of security concerns [6].

5.3.2. Opera Mini and Opera Mobile

Opera Mini is a web browser optimized for mobile phones and tablets. It uses servers from Opera Software as proxy servers that prerenders and compresses the web sites to save processing power and network traffic before sending them to the mobile device. The latest version is Opera Mini 6 and was released on March 22, 2011 [14]. In contrast to Opera Mobile 11, it does not support HTML5 and Adobe Flash. Opera Mobile 11 is the mobile version of Opera 11 and does not use a proxy server for rendering but the same rendering engine as Opera 11 and therefore has the same level of HTML5 support. It is available for the mobile OS Android, Symbian, Windows Mobile, Maemo and MeeGo. Opera Mini is available for nearly all mobile phones that support Java. A full list is available under `http://www.opera.com/mobile/download/pc/` [89].

5.4. WebKit

WebKit is an open source web browser rendering engine. Most of the code is released under the GNU Library General Public License (GNU LGPL license) but there are also components released under the Berkeley Software Distribution Licence (BSD License) [10, 1].

It is developed mainly by employees of Apple, Google and Nokia [145].

Popular browsers that use a WebKit-based rendering engine are Google Chrome and Safari.

More information on WebKit is available on the project homepage (`http://www.webkit.org`.

5.4.1. Google Chrome

Google Chrome is developed by Google Inc. and is released under the BSD-License, an Open Source license. Contribution is possible over the Google Code web site under `http://code.google.com/intl/de-DE/chromium/` [11].

Google has a website called Chrome Experiments (`http://www.chromeexperiments.com`) where users can submit links to web sites that demonstrate the features of open web technologies, like HTML5 and javascript.

Google Chrome 12

Google Chrome 12 is the latest desktop version available since 7 June, 2011 [144]. It can be installed on Windows, OS X, Linux and Chromium OS[1]. The official download link is `http://www.google.com/chrome/`.

As of June 2011, Google Chrome 12 has the highest level of support for the HTML5 specification of all desktop browsers, closely followed by Safari 5.1 (stable releases)[6]. However, some features of the HTML5 forms have not yet been implemented.

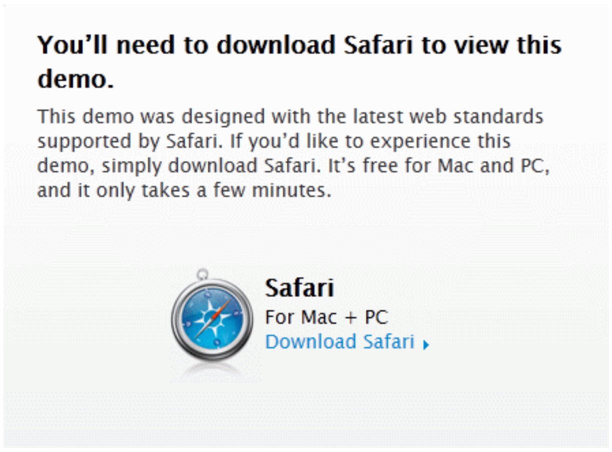
[1]Chromium OS is an OS based on Linux that is developed by Google and currently in beta status. For more information: `http://www.chromium.org/chromium-os`

Google Android Browser

The Google Android Browser is integrated in the mobile operating system Android. Android version 2.2 and 2.3 are mostly used on mobile phones whereas version 3.0 is currently only available on tablet PCs. The browsers of all three versions have roughly the same level of HTML5 support. The Audio element is only supported in versions 2.3 and 3.0. HTML5 forms and MathML are not supported by any version. All versions support SVG, Canvas, Drag and Drop and the application cache.

5.4.2. Safari

Safari is a web browser developed by Apple Inc. The Safari Welcome Page is written in HTML5 and CSS3: `http://www.apple.com/safari/welcome/`. Apple has also some HTML5 demos on its homepage under `http://www.apple.com/html5/`. These demos are only intended for the Safari web browser. Trying to access them with a different browser will often result in an error message[2]. Trying to access the URL of a demo will redirect the browser to the Apple HTML5 page.


You'll need to download Safari to view this demo.

This demo was designed with the latest web standards supported by Safari. If you'd like to experience this demo, simply download Safari. It's free for Mac and PC, and it only takes a few minutes.

Figure 5.1.: Error message when trying to access an Apple HTML5 demo with Firefox 4

Safari 5

Safari 5 is available for Mac OS X and Windows and was released on June 7, 2010. It is available free of charge under `http://www.apple.com/safari/`. The full license agreement can be found here:

- For Windows: `http://www.apple.com/safari/download/terms_win.html`
- For Mac: `http://www.apple.com/safari/download/terms_mac.html`

[2]The demos also work with other WebKit browsers, like Google Chrome

As of June 2011, the current version is Safari 5.1. It does support Video and Audio – but not OGG Theora and OGG Vorbis codecs – and and only some of the new form features. It supports Web Storage, Application Cache, Drag and Drop, Canvas, MathML and SVG [6].

Safari for iOS

Mobile versions of Safari are available only for the mobile operating system iOS (iPhone, iPad, iPod touch) [2, 3, 4]. Compared to the desktop version, the current version 4.3 has a much lower level of support for HTML5. It does not support Drag and Drop, form validation, MathML and SVG. It supports the Web Sockets and Web Storage APIs but not the Web Workers API. [6].

5.5. Frameworks and Libraries for ensuring backwards and cross-browser compatibility

There are several Frameworks and javascript Libraries available that help web authors to implement backwards compatible HTML5 code. Some of them are presented in this section.

5.5.1. HTML5 Boilerplate

HTML5 Boilerplate is a framework for creating websites with HTML, CSS and javascript. The features of the framework include cross-browser compatibility (including older versions), HTML5 tags, mobile browser optimization, progressive enhancement[3] and graceful degradation[4] [75].

It is available for download on the homepage `http://html5boilerplate.com` and published as open-source software.

5.5.2. 52framework

The 52framework can be used to create websites with HTML5, CSS3 and javascript and is available under the open-source Creative Commons license. The URL of the project homepage is `http://www.52framework.com`. It supports Internet Explorer from version 6, Firefox from version 3, Safari from version 4 and Google Chrome from version 4 onwards [8].

[3]`http://en.wikipedia.org/wiki/Progressive_enhancement`
[4]`http://en.wikipedia.org/wiki/Graceful_degradation`

5.5.3. Modernizr

The javascript Library Modernizr is able to detect browser support for HTML5 and CSS3 features and allows to define alternative code that gets loaded if a feature is not supported or prevent browsers from loading resources they can not display. It is open-source (MIT and BSD licenses) and can be downloaded from the project homepage `http://www.modernizr.com` [84].

5.5.4. HTML5 Browser Polyfills

Polyfills are compatibility workarounds written in javascript that emulate HTML5 features in browsers with no native support. A comprehensive list of Polyfills can be found in the Modernizr wiki on:
`http://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-browser-Polyfills`.
Polyfills can be used together with Modernizer to load them when they are needed [83].

6. Examples of HTML5 in use

This section provides a selection of web sites and web applications that already use HTML5 technologies.

6.1. The HTML5 Logo

As part of the marketing effort for HTML5, a HTML5 logo was created by the W3C and presented on 18 January 2011. It is free to use (licensed under the Creative Commons Attribution 3.0) and can be downloaded from the HTML5 logo homepage: `http://www.w3.org/html/logo/` [137].

Figure 6.1.: W3C HTML5 logo [124]

The "Badge Builder 5000" that is also available on the logo homepage allows users to create an HTML5 logo for a web page, that shows which technologies the page uses. The technologies are split in eight classes with their own logo. Included in these classes are CSS3 and javascript APIs like Web Workers, Canvas, WebGL, Geolocation and Web Sockets [123].

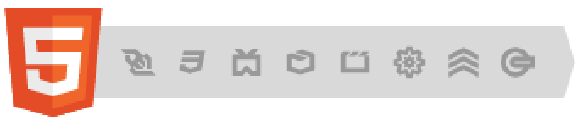


Figure 6.2.: W3C HTML5 logo badge created with the Badge Builder 5000 with all eight class logos in grey [125]

A gallery shows some homepages that use HTML5 technologies and the HTML5 logo: `http://www.w3.org/html/logo/#the-gallery`. Users are also encouraged to post pages that use the HTML5 logo on twitter with the hashtag #html5logo, although a search for the hashtag on Twitter on 11 June 2011 only brought a few results [101].

6.2. Quake II Google Web Toolkit Port

Google developers ported the Jake2 engine (Java port of the Quake II game engine) to javascript using the Google Web Toolkit. This allows playing the video game Quake II in a browser that supports HTML5, WebGL and WebSockets. The code and installation instructions for Mac OS X and Linux are available under `http://code.google.com/p/quake2-gwt-port/` [92].

6.3. YouTube HTML5 player

Most YouTube videos can be viewed in an HTML5 video player without the need for an Adobe Flash browser plugin provided the required codecs are installed. To use the YouTube HTML5 player, users have to log in their Google/YouTube account and go to `http://www.youtube.com/html5` and click on “participate in the HTML5 test”. YouTube uses the Codecs h.264 and WebM[1].

6.4. Google Mail

Google Mail `http://googlemail.com` already uses several HTML5 technologies to enhance the capabilities of the web application. E.g. it is possible to use drag&drop to add an attachment to an e-mail. In addition, the web application has a different user interface when accessed from a mobile device that is optimized for smaller touch screens [12].

6.5. HTML5demos.com

The web site `http://html5demos.com` contains several small examples of HTML5 technologies coded by Remy Sharp. Demos can be filtered by browser support (Internet Explorer, Chrome, Safari or Opera) and by the technologies they use.

6.6. canvaspaint.org

The web site `http://canvaspaint.org` uses the Canvas API to recreate Microsoft Paint as a web application. It was created in 2006.

6.7. kaazing.me

`http://kaazing.me` provides lots of examples for the use of the WebSockets API for real-time updates on web pages, e.g. current stock market rates or news feeds.

[1]Project Homepage: `http://www.webmproject.org`, also see Section 7.1

7. Critique

7.1. Codecs for Video and Audio

There are currently no default codecs specified in the HTML5 specification that all browsers that support the specification are required to support for the <audio> and <video> elements. In previous versions of the specification, the open source codecs OGG Vorbis for audio and OGG Theora for video were listed as default codecs and are also supported by most browsers (Opera, Firefox, Chrome). However, not all browser vendors agreed on implementing these two codecs and no agreement on any other codecs could be reached yet.

Therefore the required codecs have been removed from the specification, although with the hope that in the future an agreement on common codecs can be found [17]. E.g. Google has started developing a new open source video codec called webM (`http://www.webmproject.org/`) because OGG Theora was deemed unsuitable for YouTube. webM is also supported by Mozilla, Opera and Adobe. Until an agreement can be found, it will be necessary to offer several file formats for audio and video playback for maximum browser compatibility.

7.2. Two different HTML specifications

Two specifications for one markup language is one too many. It makes the development process too difficult and requires too much coordination. Differences in the specifications small though they are and a different outline add to the confusion. Parts of the WHATWG HTML Living Standard that are published as individual specifications by the W3C create a mess that is hard to understand.

7.3. Length of the Development Process

As of June 2011, the planned release date for the W3C: HTML5 recommendation is December 31, 2014. However, many web pages already use HTML5 features. It is questionable if the specification will still be relevant at the beginning of 2015. The WHATWG has dropped the version number for its HTML standard and calls it a “living standard”. There are already features in the WHATWG HTML living standard that are not part of the W3C HTML5 specification since the W3C HTML5 specification is feature complete since

May 2011. By the time the W3C HTML5 specification becomes a recommendation it may be already outdated, because Mozilla, Apple and Opera will likely continue implementing features from the WHATWG specification in their browsers. This would also increase the differences between browsers and browser lock-in which harms interoperability. The worst case scenario would be two different HTML standards which would harm interoperability and very likely make both of them irrelevant. HTML5 could share the fate of XHTML2 and die aborning.

7.4. Interoperability and Backwards Compatibility

One goal of the W3C: HTML5 specification is browser interoperability. All browsers that support the specification should render valid HTML5 code the same way. This should eliminate the need for so called browser hacks.

The question is if all browser vendors want that. Microsoft's Internet Explorer had the biggest market share in May 2011 and Internet Explorer 9 has the least support for HTML5 features of all current browsers [99, 98, 6]. Apple does not allow other browsers than Safari to access their HTML5 demos. There are no specific audio and video codecs defined in the specification.

Also most users are only slowly upgrading their browser versions, which makes it necessary to include fall back content for older browsers and browser hacks to ensure maximum compatibility.

Extensive frameworks and JavaScript code are required to ensure full cross-browser and backwards compatibility of HTML5 features. This bloats the HTML code and increases the size and loading times of web pages. Some web authors may decide to continue using proprietary technologies like Microsoft Silverlight or Adobe Flash instead of the Canvas API or the native video and audio playback of HTML5.

7.5. The power of the editor

Ian Hickson is the key player of the HTML5 development process as editor of both the W3C: HTML specification and the WHATWG HTML Living Standard. He has a considerable power of decision, as he is the only one that can edit and change the specification. This somewhat contradicts the community-driven development approach and the goal of creating an open web standard, because the final decision is always made by the editor. A more collaborative approach with more editors would prevent decisions being dependent on the opinions and ideology of one person.

Having only one editor is also a great risk because if the current one must be replaced for whatever reason, it would take very long for a successor to become acquainted with the task and development would be severely hindered.

7.6. W3C vs. WHATWG

After the agreement between the WHATWG and the W3C to develop HTML5 together, the WHATWG could have dissolved itself. That it did not is a reason to assume there are still differences of opinion between the two. This is also apparent when reading the archives of the W3C public HTML mailing list. Concerns are often raised when discussing the joint specification development process. The tone in e-mail conversations is often not very polite, sometimes even insulting [97].

The WHATWG from the outside seems like a one man show by Ian Hickson. The names of the other members are only listed in the group charter and do not surface anywhere else on the homepage, suggesting that the majority – if not all – of the content on the homepage was created by him. He is the editor of all WHATWG specification and also the spokesman of the group which is stated on the bottom of the homepage in very small writing. Compared to the W3C, the WHATWG has a different, more open and less bureaucratic approach to developing specifications. The members are younger and the organization structure is less complicated. The WHATWG argues that forks of the HTML standard should be allowed to create competition with the W3C specifications that might increase the quality of work the W3C delivers [149].

The future will show, how the relationship between the W3C and the WHATWG develops. From the outside, they do not seem to fit well together. The presence of two groups that want to develop global standards for the same technologies creates a tension that is not desirable for good results. Issue-151 of the HTML5 bug tracker is about the removal of all WHATWG references in the W3C HTML5 specification. This issue has been reopened by Sam Ruby in May 2011, after a previous request had been dismissed by Ian Hickson in December 2010. This could mean that the W3C is trying to cut its ties with the WHATWG [93].

7.7. Practical Relevance

Another goal of the HTML5 specification is practical relevance. The specification of HTML4 is full of bugs and implementations in browsers differ greatly from it. Therefore the HTML4 code on web pages has often very little in common with the specification. The HTML5 specification is developed together with web authors and browser vendors to ensure that it reflects the practical use of the language. However, the goal of web authors is to create web pages that most of the Internet users can access. Sometimes HTML or CSS code is not used as intended to ensure browser-compatibility. As the W3C HTML5 specification will not be changed once it becomes a recommendation it is not very likely that it will still reflect the practical use of HTML5. This is also one of the reasons why the WHATWG propagates a living standard [148].

8. Conclusion

The HTML5 specification addresses many of the shortcomings previous HTML versions have because of the rapid advancement of Internet technologies and the development of new Internet devices. It defines HTML as independent markup language and forsakes SGML and XML as fundamentals, making the syntax much more forgiving.

One focus is providing features for web applications. The Application Cache allows web applications to work even without a network connection. Drag and drop support and additional javascript APIs that enable simultaneous execution of scripts, real-time updates and data storage, further diminish the differences between desktop and web applications.

The multimedia capabilities are also increased. SVG and Canvas allow the rendering of graphics and animations. The <audio> and <video> elements allow native audio and video playback in the browser. This reduces the need for additional (proprietary) browser plugins, which is also an advantage for mobile devices, which often can not use these plugins.

HTML5 also caters to mobile devices with additional elements for structuring web sites and creating an outline and additional input types for forms. Those elements allow browsers to better understand the content structure of web pages and to provide better usability through navigational aids and user input validation. In addition, several Frameworks for developing native mobile applications for nearly all current mobile operating systems are available that allow cross-platform development in HTML5, CSS3 and javascript.

Browser support for HTML5 is getting better and better and many web sites already use HTML5 features. Several Frameworks and javascript Libraries already allow the use of HTML5 features with backwards compatibility for older browsers. If the goal of interoperability for all browsers that support the standard will be achieved remains to be seen after the specification has reached W3C recommendation status.

The joint development process by the W3C and the WHATWG with the W3C HTML5 specification on the one side and the WHATWG HTML Living Standard on the other side causes a little uncertainty about the future of HTML5. By the time the specification is completed it may be already outdated compared to the WHATWG standard, that stays constantly in development. Also it is unsure what influence the WHATWG will have on the W3C in the future after it brought the W3C to abandon the development of XHTML2 for HTML5. The tensions between the two groups make future joint efforts not very likely.

As the importance of the Internet further increases, so will the need for a modern language of the web. HTML5 tries to fulfill that role. Numerous projects around HTML5, like mobile application frameworks and HTML5 frameworks indicate that web developers are looking forward to HTML5 and are quick to adopt it. This indicates a promising future for the language.

A. Appendix

A.1. HTML5 Code Examples

All examples in this section (except the PhoneGap manifest.xml) and all examples provided in chapter 3 are also available online on the WU homepage of the author at `http://www.wu.ac.at/usr/h05c/h0551854/html5`.

A.1.1. Structure and Outline

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="utf-8">
 <title>HTML5: Structure and Outline Example</title>

 <style>
 header, footer, nav, article, aside {display:block;}

 aside {
  float: left;
  width: 30%;
  height: 100%;
 }

 #mainContent {
  margin-left: 30%;
  padding: 10pt;
 }

 h1, h2, h3, h4, h5, h6 {
  font-family: Arial,sans-serif;
 }

 body footer {
  text-align: center;
  color: grey;
 }
 </style>
</head>

<body>
 <header>
  <h1>Structure and Outline Example</h1>
  <p>Coded in HTML5</p>
 </header>

 <aside>
  <nav>
  <h1>Navigation</h1>
  <ul>
   <li><a href="#">Link1</a></li>
   <li><a href="#">Link2</a></li>
   <li><a href="#">Link3</a></li>
  </ul>
  </nav>
 </aside>

 <section id="mainContent">
  <header>
   <h1>Articles on this Page</h1>
   <p>This section contains articles published on this page.</p>
  </header>
  <article>
   <h1>Article 1</h1>
   <p>This is an Article.</p>
   <footer>
    <p>Posted on
        <time datetime="2011-05-29">29 Mai 2011</time>,
        by Sebastian Schwarz</p>
   </footer>
  </article>
  <article>
   <h1>Article 2</h1>
```

```html
    <p>This is an Article.</p>
    <footer>
     <p>Posted on
         <time datetime="2011-05-29">29 Mai 2011</time>,
         by Sebastian Schwarz</p>
    </footer>
   </article>
  </section>

  <section id="additional">
   <h1>Additional Information</h1>
   <ul>
    <li>
     <a href="http://html5doctor.com/designing-a-blog-with-html5/">
    HTML5 Doctor</a>
    </li>
    <li><a href="http://www.w3.org/TR/html5/sections.html#sections">
    W3C HTML5 Specification</a>
    </li>
   </ul>
  </section>

  <footer>
   &copy; 2011, Sebastian Schwarz
   <address>
    <a href="mailto:h0551854@wu.ac.at">h0551854@wu.ac.at</a>
   </address>
  </footer>
</body>

</html>
```

A.1.2. Online hotel reservation form

```html
<!DOCTYPE html>

<html>
<head>
<title>Online Hotel Reservation</title>
<style>
```

```
label {
display: block;
float: left;
width: 100%;
padding: 5px;
}

input[type=range]::before {content: attr(min);}
input[type=range]::after {content: attr(max);}
</style>


</head>

<body>
 <form>
 <h1>Online Hotel Reservation</h1>
 <fieldset>
 <legend>Personal Data</legend>
 <label for="fullname">
  full name:
  <input type="text" id="fullname" name="fullname" size="50"
   placeholder="forename, surname" pattern="([a-zA-Z]+ *)+, ([a-zA-Z]+ *)+"
   autofocus required>
   </label>
 <label for="salutation">salutation:
  <input type="text" list="salutations" id="salutation" name="salutation">
  <datalist id="salutations">
   <option label="Mr." value="Mr.">
   <option label="Ms." value="Ms.">
  </datalist>
 </label>
 <label for="email">e-mail:
  <input type="email" id="email" name="email" required>
 </label>
 <label for="phone">phone:
  <input type="tel" id="phone" name="phone">
 </label>
 <label for="hp">homepage:
  <input type="url" id="hp" name="homepage">
 </label>
 </fieldset>

 <fieldset>
 <legend>Reservation Details</legend>
  <label for="doa">date and time of arrival
   <input type="datetime" id="doa" name="doa" required>
  </label>
  <label for="dod">date and time of departure
   <input type="datetime" id="dod" name="dod" required>
  </label>
  <label for="beds">beds
   <input id="beds" name="beds" type="range" min="1" max="4" step="1">
  </label>
 </fieldset>
 <input type="submit" value="send">
 </form>
</body>
</html>
```

A.1.3. Drag and drop with Web Storage

```
<!DOCTYPE html>

<html>
<head>
<title>Drag and drop</title>
</head>

<body>
<h1>Drag and drop</h1>
<nav id="menu1" style="height: 100%; width: 100%;
  background-color: red;padding: 10px;"
  ondragstart="this.style.background = 'orange';"
  ondragend="this.style.background = 'red';">
 <h1>Menu 1</h1>
 <ul>
 <li><a href="http://www.wu.ac.at">WU Wien</a></li>
 <li>
```

```
  <a href="http://www.wu.ac.at/ec/">
 Institute for Management Information Systems
  </a>
 </li>
 <li>
  <a href="http://dev.w3.org/html5/spec/dnd.html#dnd">
  Drag and drop in the W3C HTML5 specification
  </a>
 </li>
 </ul>
</nav>

<section style="height: 100%; width: 100%;
  background-color: green;padding: 10px;"
  dropzone="copy s:URL"
  ondrop="dropHandler(event);this.style.background = 'green';"
  ondragover="this.style.background = 'yellow';return false;"
  ondragenter="return false;">
 <h2>Menu 2</h2>
 <p>Drag and drop links here.</p>
 <ul id="dropzone">
 </ul>
</section>

<a href="javascript:if(localStorage) {
                      localStorage.clear();
                      location.reload(true);
                    };">
Delete Menu 2
</a>

<script type="text/javascript">
 //get nav element of menu2
 var drop = document.getElementById('dropzone');

 if (localStorage && localStorage.getItem('SavedMenu2')) {
  drop.innerHTML = localStorage.getItem('SavedMenu2');
  alert('Saved Menu 2 loaded!');
 }

 function dropHandler(event) {
  if (event.dataTransfer.getData('URL')) {
   drop.innerHTML += '<li><a href="'
                     + event.dataTransfer.getData('URL')
                     + '">'
                     + event.dataTransfer.getData('URL') + '</a></li>';
  }
  //save Menu
  localStorage.setItem('SavedMenu2', drop.innerHTML);
  // Stops Firefox from redirecting.
  if (event.preventDefault) {
   event.preventDefault();
  }
 }
</script>


</body>
</html>
```

A.1.4. Web Workers

```
<!DOCTYPE html>

<html>
<head>
<title>Web Workers Example</title>
</head>

<body>

<section>
 <h1>Countdown</h1>
 <p>Countdown executed by Worker: <span id="countdown"></span></p>
 <script type="text/javascript">
  //create worker
  var worker = new Worker('countdown.js');

  //get current countdown number from worker
```

```
   worker.onmessage = function (event) {
     document.querySelector('#countdown').innerHTML = event.data;
   }
  </script>
  <input type="button" onclick="worker.terminate();"
  value="Stop the worker">
</section>

<section>
  <h1>Push the Button</h1>
  <input type="button" onclick="buttonPushed();"
  value="Push the button.">
  <h2>Button Log</h1>
  <p>Logs how often and when the button was pushed</p>.
  <ol id="buttonPushed">
  </ol>

  <script type="text/javascript">
   function buttonPushed() {
    //get current time
    var date = new Date();
    var hrs = date.getHours();
    var min = date.getMinutes();
    var sec = date.getSeconds();

    //write current time in Log
    document.querySelector('#buttonPushed').innerHTML +=
    '<li>Button Push at ' + hrs + ':' + min + ':' + sec + '</li>';
   }
  </script>
</section>

</body>
</html>
```

A.1.5. PhoneGap manifest.xml for Android

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.phonegap.helloworld"
 android:versionCode="1"
 android:versionName="1.0">

 <!-- begin permissions for phonegap -->
 <supports-screens
  android:largeScreens="true"
  android:normalScreens="true"
  android:smallScreens="true"
  android:resizeable="true"
  android:anyDensity="true"
 />
 <uses-permission android:name="android.permission.CAMERA" />
 <uses-permission android:name="android.permission.VIBRATE" />
 <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
 <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
 <uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
 <uses-permission android:name="android.permission.READ_PHONE_STATE" />
 <uses-permission android:name="android.permission.INTERNET" />
 <uses-permission android:name="android.permission.RECEIVE_SMS" />
 <uses-permission android:name="android.permission.RECORD_AUDIO" />
 <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
 <uses-permission android:name="android.permission.READ_CONTACTS" />
 <uses-permission android:name="android.permission.WRITE_CONTACTS" />
 <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
 <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
 <!-- end permissions for phonegap -->

 <application android:icon="@drawable/icon" android:label="@string/app_name">
 <!-- add attribute android:configChanges="orientation}keyboardHidden"
  to activity element-->
 <activity android:name=".App"
  android:label="@string/app_name"
  android:configChanges="orientation}keyboardHidden">
 <intent-filter>
 <action android:name="android.intent.action.MAIN" />
 <category android:name="android.intent.category.LAUNCHER" />
 </intent-filter>
 </activity>

 </application>
</manifest>
```

Bibliography

[1] Apple: BSD License. http://www.webkit.org/coding/bsd-license.html, viewed 10 June 2011.

[2] Apple: iPad – Safari. http://www.apple.com/ipad/built-in-apps/safari.html, viewed 10 June 2011.

[3] Apple: iPhone – Safari. http://www.apple.com/iphone/features/safari.html, viewed 10 June 2011.

[4] Apple: iPod touch – Safari. http://www.apple.com/de/ipodtouch/features/safari.html, viewed 10 June 2011.

[5] Cotton, Paul: Last Call timeline. http://lists.w3.org/Archives/Public/public-html/2011May/0162.html, viewed 9 June 2011.

[6] Deveria, Alexis: When can I use... Support tables for HTML5, CSS3, etc. http://caniuse.com/#cats=HTML5, viewed 10 June 2011.

[7] Duerst, M. and Suignard, M.: RFC3987 Internationalized Resource Identifiers. http://tools.ietf.org/html/rfc3987, viewed 9 June 2011.

[8] enavu network: 52framework. http://www.52framework.com/, viewed 10 June 2011.

[9] European Commission: Web browser choice for European consumers. http://ec.europa.eu/competition/consumers/web_browsers_choice_en.html, viewed 10 June 2011.

[10] Free Software Foundation: GNU LIBRARY GENERAL PUBLIC LICENSE. http://www.webkit.org/coding/lgpl-license.html, viewed 10 June 2011.

[11] Google: Chromium Terms and Conditions. http://code.google.com/intl/de-DE/chromium/terms.html, viewed 10 June 2011.

[12] Google: Google Mail for mobiles. `http://www.google.com/mobile/mail/`, viewed 10 June 2011.

[13] Google: Web Authoring Statistics. `http://code.google.com/intl/en-US/webstats/`, viewed 9 June 2011.

[14] H., Marian: Opera Mini 6 is here!. `http://my.opera.com/operamini/blog/opera-mini-6-is-here`, viewed 10 June 2011.

[15] Heilmann, Chris: WebSocket disabled in Firefox 4. `http://hacks.mozilla.org/2010/12/websockets-disabled-in-firefox-4/`, viewed 10 June 2011.

[16] Hickson, Ian: Beneath The Surface. `http://ln.hixie.ch/?start=%201115899732&count=1`, viewed 10 June 2011.

[17] Hickson, Ian: Codecs for <audio> and <video>. `http://lists.whatwg.org/pipermail/whatwg-whatwg.org/2009-June/020620.html`, viewed 10 June 2011.

[18] Hickson, Ian: DOM trees. `http://dev.w3.org/html5/spec/Overview.html#dom-trees`, viewed 9 June 2011.

[19] Hickson, Ian: HTML - Living Standard - 1.5 History. `http://www.whatwg.org/specs/web-apps/current-work/multipage/introduction.html#history-1`, viewed 9 June 2011.

[20] Hickson, Ian: HTML5 specification – A quick introduction to HTML. `http://dev.w3.org/html5/spec/Overview.html#a-quick-introduction-to-html`, viewed 9 June 2011.

[21] Hickson, Ian: HTML5 specification – Application caches. `http://dev.w3.org/html5/spec/Overview.html#application-cache`, viewed 10 June 2011.

[22] Hickson, Ian: HTML5 specification – Autofocusing a form control. `http://dev.w3.org/html5/spec/Overview.html#attr-fe-autofocus`, viewed 10 June 2011.

[23] Hickson, Ian: HTML5 specification – Creating an outline. `http://dev.w3.org/html5/spec/sections.html#outline`, viewed 10 June 2011.

[24] Hickson, Ian: HTML5 specification – Date and Time state. `http://dev.w3.org/html5/spec/Overview.html#date-and-time-state`, viewed 10 June 2011.

[25] Hickson, Ian: HTML5 specification – Dimension attributes. `http://dev.w3.org/html5/spec/Overview.html#attr-dim-width`, viewed 10 June 2011.

[26] Hickson, Ian: HTML5 specification – Drag and drop. `http://dev.w3.org/html5/spec/Overview.html#dnd`, viewed 10 June 2011.

[27] Hickson, Ian: HTML5 Specification – Drag and drop – Introduction. `http://dev.w3.org/html5/spec/dnd.html#introduction-7`, viewed 12 June 2011.

[28] Hickson, Ian: HTML5 specification – E-mail state. `http://dev.w3.org/html5/spec/Overview.html#e-mail-state`, viewed 10 June 2011.

[29] Hickson, Ian: HTML5 specification – Form Security. `http://dev.w3.org/html5/spec/Overview.html#security-forms`, viewed 10 June 2011.

[30] Hickson, Ian: HTML5 specification – Forms. `http://dev.w3.org/html5/spec/Overview.html#forms`, viewed 10 June 2011.

[31] Hickson, Ian: HTML5 specification – Headings and sections. `http://dev.w3.org/html5/spec/sections.html#headings-and-sections`, viewed 10 June 2011.

[32] Hickson, Ian: HTML5 specification – Location of the media resource. `http://dev.w3.org/html5/spec/Overview.html#attr-media-src`, viewed 10 June 2011.

[33] Hickson, Ian: HTML5 specification – MathML. `http://dev.w3.org/html5/spec/Overview.html#mathml`, viewed 10 June 2011.

[34] Hickson, Ian: HTML5 specification – Range state. `http://dev.w3.org/html5/spec/Overview.html#range-state`, viewed 10 June 2011.

[35] Hickson, Ian: HTML5 specification – Sectioning Content Elements. `http://dev.w3.org/html5/spec/content-models.html#sectioning-content`, viewed 9 June 2011.

[36] Hickson, Ian: HTML5 specification – Sectioning Root Elements. `http://dev.w3.org/html5/spec/sections.html#sectioning-root`, viewed 9 June 2011.

[37] Hickson, Ian: HTML5 specification – Telephone state. `http://dev.w3.org/html5/spec/Overview.html#telephone-state`, viewed 10 June 2011.

[38] Hickson, Ian: HTML5 specification – The address element. `http://dev.w3.org/html5/spec/sections.html#the-address-element`, viewed 10 June 2011.

[39] Hickson, Ian: HTML5 specification – The article element. `http://dev.w3.org/html5/spec/sections.html#the-article-element`, viewed 9 June 2011.

[40] Hickson, Ian: HTML5 specification – The aside element. `http://dev.w3.org/html5/spec/sections.html#the-aside-element`, viewed 9 June 2011.

[41] Hickson, Ian: HTML5 specification – The audio element. `http://dev.w3.org/html5/spec/Overview.html#the-audio-element`, viewed 10 June 2011.

[42] Hickson, Ian: HTML5 specification – The autoplay attribute. `http://dev.w3.org/html5/spec/Overview.html#attr-media-autoplay`, viewed 10 June 2011.

[43] Hickson, Ian: HTML5 specification – The cache manifest syntax. `http://dev.w3.org/html5/spec/Overview.html#manifests`, viewed 10 June 2011.

[44] Hickson, Ian: HTML5 specification – The canvas element. `http://dev.w3.org/html5/spec/Overview.html#the-canvas-element`, viewed 10 June 2011.

[45] Hickson, Ian: HTML5 specification – The controls attribute. `http://dev.w3.org/html5/spec/Overview.html#attr-media-controls`, viewed 10 June 2011.

[46] Hickson, Ian: HTML5 specification – The datalist element. `http://dev.w3.org/html5/spec/Overview.html#the-datalist-element`, viewed 10 June 2011.

[47] Hickson, Ian: HTML5 specification – The figure element. `http://dev.w3.org/html5/spec/Overview.html#the-figure-element`, viewed 12 June 2011.

[48] Hickson, Ian: HTML5 specification – The footer element. `http://dev.w3.org/html5/spec/sections.html#the-footer-element`, viewed 10 June 2011.

[49] Hickson, Ian: HTML5 specification – The header element. `http://dev.w3.org/html5/spec/Overview.html#the-header-element`, viewed 10 June 2011.

[50] Hickson, Ian: HTML5 specification – The loop attribute. `http://dev.w3.org/html5/spec/Overview.html#attr-media-loop`, viewed 10 June 2011.

[51] Hickson, Ian: HTML5 specification – The manifest attribute. `http://dev.w3.org/html5/spec/Overview.html#attr-html-manifest`, viewed 10 June 2011.

[52] Hickson, Ian: HTML5 specification – The mediagroup attribute. `http://dev.w3.org/html5/spec/Overview.html#attr-media-mediagroup`, viewed 10 June 2011.

[53] Hickson, Ian: HTML5 specification – The muted attribute. `http://dev.w3.org/html5/spec/Overview.html#attr-media-muted`, viewed 10 June 2011.

[54] Hickson, Ian: HTML5 specification – The nav element. `http://dev.w3.org/html5/spec/sections.html#the-nav-element`, viewed 9 June 2011.

[55] Hickson, Ian: HTML5 specification – The pattern attribute. `http://dev.w3.org/html5/spec/Overview.html#the-pattern-attribute`, viewed 10 June 2011.

[56] Hickson, Ian: HTML5 specification – The placeholder attribute. `http://dev.w3.org/html5/spec/Overview.html#attr-input-placeholder`, viewed 10 June 2011.

[57] Hickson, Ian: HTML5 specification – The poster attribute. `http://dev.w3.org/html5/spec/Overview.html#attr-video-poster`, viewed 10 June 2011.

[58] Hickson, Ian: HTML5 specification – The preload attribute. `http://dev.w3.org/html5/spec/Overview.html#attr-media-preload`, viewed 10 June 2011.

[59] Hickson, Ian: HTML5 specification – The required attribute. `http://dev.w3.org/html5/spec/Overview.html#attr-input-required`, viewed 10 June 2011.

[60] Hickson, Ian: HTML5 specification – The section element. `http://dev.w3.org/html5/spec/sections.html#the-section-element`, viewed 10 June 2011.

[61] Hickson, Ian: HTML5 specification – The source element. `http://dev.w3.org/html5/spec/Overview.html#the-source-element`, viewed 10 June 2011.

[62] Hickson, Ian: HTML5 specification – The video element. `http://dev.w3.org/html5/spec/Overview.html#the-video-element`, viewed 10 June 2011.

[63] Hickson, Ian: HTML5 specification – URL state. `http://dev.w3.org/html5/spec/Overview.html#url-state`, viewed 10 June 2011.

[64] Hickson, Ian: HTML5 specification design notes. `http://dev.w3.org/html5/spec/Overview.html#design-notes`, viewed 9 June 2011.

[65] Hickson, Ian: New split-out drafts. `http://lists.w3.org/Archives/Public/public-html/2010Jan/0320.html`, viewed 14 June 2011.

[66] Hickson, Ian: Web Workers - Dedicated workers and the Worker interface. `http://dev.w3.org/html5/workers/#dedicated-workers-and-the-worker-interface`, viewed 10 June 2011.

[67] Hickson, Ian: Web Workers - Introduction. `http://dev.w3.org/html5/workers/#introduction`, viewed 10 June 2011.

[68] Hickson, Ian: Web Workers - Tutorial. `http://dev.w3.org/html5/workers/#tutorial`, viewed 10 June 2011.

[69] Hickson, Ian: The WebSocket API – The WebSocket interface. `http://dev.w3.org/html5/websockets/#the-websocket-interface`, viewed 10 June 2011.

[70] Hickson, Ian: Webstorage API - The localStorage attribute. `http://dev.w3.org/html5/webstorage/#the-localstorage-attribute`, viewed 10 June 2011.

[71] Hickson, Ian: Webstorage API - The storage interface. `http://dev.w3.org/html5/webstorage/#the-storage-interface`, viewed 10 June 2011.

[72] Hickson, Ian: Webstorage API - The Storage interface. `http://dev.w3.org/html5/webstorage/#the-storage-interface`, viewed 10 June 2011.

[73] Hickson, Ian: WHATWG HTML Standard. `http://www.whatwg.org/specs/web-apps/current-work/multipage/`, viewed 9 June 2011.

[74] IE Mobile Team: IE9 Coming to Windows Phone in 2011. `http://blogs.msdn.com/b/iemobile/archive/2011/02/14/ie9-coming-to-windows-phone-in-2011.aspx`, viewed 10 June 2011.

[75] Irish, Paul, Manian, Divya and Shichuan: HTML5 Boilerplate is 1.0!. `http://html5boilerplate.com/#intro`, viewed 10 June 2011.

[76] Lawson, Bruce and Sharp, Remy: Introducing HTML5. New Riders, 2010, pp. 2–5.

[77] Lawson, Bruce and Sharp, Remy: Introducing HTML5. New Riders, 2010, p. 124.

[78] Lawson, Bruce and Sharp, Remy: Introducing HTML5. New Riders, 2010, p. 167.

[79] LeRoux, Brian: phonegap-android-eclipse-quickstart. `http://wiki.phonegap.com/w/page/30862722/phonegap-android-eclipse-quickstart`, viewed 10 June 2011.

[80] Lubbers, Peter and Greco, Frank: HTML5 Web Sockets: A Quantum Leap in Scalability for the Web. `http://websocket.org/quantum.html`, viewed 10 June 2011.

[81] Microsoft: Download Internet Explorer 9. `http://windows.microsoft.com/en-US/internet-explorer/downloads/ie-9/worldwide-languages`, viewed 10 June 2011.

[82] Microsoft: Microsoft Announces Global Availability of Internet Explorer 9. `http://www.microsoft.com/presspass/press/2011/mar11/03-14IE9RTWPR.mspx`, viewed 10 June 2011.

[83] modernizr.com: Polyfills and Modernizr. `http://www.modernizr.com/docs/#polyfills`, viewed 10 June 2011.

[84] modernizr.com: Why use Modernizr?. `http://www.modernizr.com/`, viewed 10 June 2011.

[85] Mozilla: Firefox Home Ű looking to the future. `http://blog.mozilla.com/mobile/2010/09/28/firefox-home-looking-to-the-future/`, viewed 10 June 2011.

[86] Mozilla: Mozilla Public License Version 1.1. `http://www.mozilla.org/MPL/MPL-1.1.html`, viewed 10 June 2011.

[87] Opera: Business solutions. `http://www.opera.com/business/devices/`, viewed 10 June 2011.

[88] Opera: ID attribute list. `http://devfiles.myopera.com/articles/572/idlist-url.htm`, viewed 9 June 2011.

[89] Opera: Opera Mini & Opera Mobile. `http://www.opera.com/mobile/specs/`, viewed 10 June 2011.

[90] PhoneGap: Get Started Guide. `http://www.phonegap.com/start#android`, viewed 10 June 2011.

[91] PhoneGap: PhoneGap/build FAQ. `https://build.phonegap.com/faq`, viewed 10 June 2011.

[92] Ramsdale, Chris: Look ma, no plugin!. `http://googlewebtoolkit.blogspot.com/2010/04/look-ma-no-plugin.html`, viewed 10 June 2011.

[93] Ruby, Sam: ISSUE-151: Remove WhatWG and html5.org references in status section of document. `http://www.w3.org/html/wg/tracker/issues/151?changelog`, viewed 16 June 2011.

[94] Ruby, Sam: Revert request for r6052, moving forward on bug 11828 and issue 164. `http://lists.w3.org/Archives/Public/public-html/2011May/0061.html`, viewed 10 June 2011.

[95] S., Adam: Use HTML5 in Your BlackBerry Web Content!. `http://devblog.blackberry.com/2010/03/use-html5-in-your-blackberry-web-content/`, viewed 10 June 2011.

[96] Sharp, Remy: Native Drag and Drop. `http://html5doctor.com/native-drag-and-drop/`, viewed 12 June 2011.

[97] Stachowiak, Maciej: Re: New split-out drafts vs. modular design. `http://lists.w3.org/Archives/Public/public-html/2010Jan/0422.html`, viewed 14 June 2011.

[98] StatCounter.com: GlobalStats. `http://gs.statcounter.com/`, viewed 10 June 2011.

[99] StatOwl.com: Web Browser Market Share. `http://www.statowl.com/web_browser_market_share.php`, viewed 10 June 2011.

[100] Thompson, Henry S.: The future of applications: W3C TAG perspectives. `http://www.w3.org/2001/tag/doc/IAB_Prague_2011_slides.html`, viewed 9 June 2011.

[101] Twitter: Search results for #html5logo. `http://twitter.com/#!/search/%23html5logo`, viewed 10 June 2011.

[102] van Kesteren, Anne and Pieters, Simon: HTML5 differences from HTML4 – APIs. `http://dev.w3.org/html5/html4-differences/#apis`, viewed 9 June 2011.

[103] van Kesteren, Anne and Pieters, Simon: HTML5 differences from HTML4 – Language. `http://dev.w3.org/html5/html4-differences/#language`, viewed 9 June 2011.

[104] van Kesteren, Anne and Pieters, Simon: HTML5 differences from HTML4 – MathML and SVG. `http://dev.w3.org/html5/html4-differences/#mathml-svg`, viewed 9 June 2011.

[105] van Kesteren, Anne and Pieters, Simon: HTML5 differences from HTML4 – Miscellaneous. `http://dev.w3.org/html5/html4-differences/#syntax-misc`, viewed 9 June 2011.

[106] van Kesteren, Anne and Pieters, Simon: HTML5 differences from HTML4 – The DOCTYPE. `http://dev.w3.org/html5/html4-differences/#doctype`, viewed 9 June 2011.

[107] W3C: Advancing a Technical Report to Recommendation. `http://www.w3.org/2005/10/Process-20051014/tr.html#rec-advance`, viewed 9 June 2011.

[108] W3C: Advisory Committee (AC). `http://www.w3.org/2005/10/Process-20051014/organization#AC`, viewed 9 June 2011.

[109] W3C: Bug / Issue Tracking Service HTML WG. `http://www.w3.org/Bugs/Public/describecomponents.cgi?product=HTML%20WG`, viewed 9 June 2011.

[110] W3C: Call for Review of a Proposed Recommendation. `http://www.w3.org/2005/10/Process-20051014/tr.html#cfr`, viewed 9 June 2011.

[111] W3C: Can I participate "as an individual" in W3C?. `http://www.w3.org/participate/faq.html#individual`, viewed 9 June 2011.

[112] W3C: Consensus. `http://www.w3.org/2005/10/Process-20051014/policies.html#Consensus`, viewed 9 June 2011.

[113] W3C: Current Members. `http://www.w3.org/Consortium/Member/List`, viewed 16 June 2011.

[114] W3C: FAQ for HTML5 Last Call. `http://www.w3.org/2011/05/html5lc-faq.html`, viewed 9 June 2011.

[115] W3C: Frequently Asked Questions (FAQ) about Public Invited Experts in the W3C HTML Working Group. `http://www.w3.org/2007/04/html-ie-faq`, viewed 9 June 2011.

[116] W3C: Groups. `http://www.w3.org/Consortium/activities.html`, viewed 9 June 2011.

[117] W3C: History. `http://www.w3.org/Consortium/facts#history`, viewed 9 June 2011.

[118] W3C: HTML <fieldset> Tag. `http://www.w3schools.com/tags/tag_fieldset.asp`, viewed 10 June 2011.

[119] W3C: HTML <label> Tag. `http://www.w3schools.com/tags/tag_label.asp`, viewed 10 June 2011.

[120] W3C: HTML <legend> Tag. `http://www.w3schools.com/tags/tag_legend.asp`, viewed 10 June 2011.

[121] W3C: HTML WG Deliverables. `http://www.w3.org/2007/03/HTML-WG-charter.html#deliverables`, viewed 9 June 2011.

[122] W3C: HTML Working Group Charter. `http://www.w3.org/2007/03/HTML-WG-charter`, viewed 9 June 2011.

[123] W3C: HTML5 Logo. `http://www.w3.org/html/logo/`, viewed 10 June 2011.

[124] W3C: HTML5 Logo 256px.
http://www.w3.org/html/logo/downloads/HTML5_Logo_256.png, viewed 10 June 2011.

[125] W3C: HTML5 Logo badge with all technologies.
http://www.w3.org/html/logo/badge/
html5-badge-h-connectivity-css3-device-graphics-multimedia-performance-semantics
png, viewed 10 June 2011.

[126] W3C: HTML5 Test Suite Conformance Results.
http://w3c-test.org/html/tests/reporting/report.htm, viewed 13 June 2011.

[127] W3C: Instructions for non-Members (Invited Experts).
http://www.w3.org/2004/08/invexp.html, viewed 9 June 2011.

[128] W3C: Join W3C. http://www.w3.org/Consortium/join, viewed 9 June 2011.

[129] W3C: Maturity Levels.
http://www.w3.org/2005/10/Process-20051014/tr.html#maturity-levels, viewed 9 June 2011.

[130] W3C: Membership Fees - W3C. http://www.w3.org/Consortium/fees, viewed 9 June 2011.

[131] W3C: People of W3C. http://www.w3.org/Consortium/facts#people, viewed 9 June 2011.

[132] W3C: public-html@w3.org Mail Archives.
http://lists.w3.org/Archives/Public/public-html/, viewed 9 June 2011.

[133] W3C: Rejection of a Submission Request. http:
//www.w3.org/2005/10/Process-20051014/submission.html#SubmissionNo, viewed 9 June 2011.

[134] W3C: SVG animate Element. http://www.w3schools.com/svg/el_animate.asp, viewed 10 June 2011.

[135] W3C: Technical Architecture Group (TAG).
http://www.w3.org/2005/10/Process-20051014/organization#TAG, viewed 9 June 2011.

[136] W3C: W3C Advisory Board. http://www.w3.org/2002/ab/, viewed 9 June 2011.

[137] W3C: W3C Introduces an HTML5 Logo.
http://www.w3.org/News/2011#entry-8992, viewed 10 June 2011.

[138] W3C: W3C Process Document.
http://www.w3.org/2005/10/Process-20051014/, viewed 9 June 2011.

[139] W3C: The W3C Team.
http://www.w3.org/2005/10/Process-20051014/organization.html#Team, viewed 9 June 2011.

[140] W3C: W3C Technical Report Development Process.
http://www.w3.org/2005/10/Process-20051014/tr.html, viewed 9 June 2011.

[141] W3C: When can I use HTML5?.
http://www.w3.org/html/wiki/FAQs#When_can_I_use_HTML5.3F, viewed 9 June 2011.

[142] W3C: Why is HTML5 so exciting?.
http://www.w3.org/html/wiki/FAQs#Why_is_HTML5_so_exciting.3F, viewed 13 June 2011.

[143] W3C: Working Groups, Interest Groups, and Coordination Groups.
http://www.w3.org/2005/10/Process-20051014/groups.html, viewed 9 June 2011.

[144] Walker, Adrienne: A new stable release of Chrome: safer and snazzier.
http://chrome.blogspot.com/2011/06/chrome-12-safer-and-snazzier.html, viewed 10 June 2011.

[145] WebKit Open Source Project: WebKit Team.
http://trac.webkit.org/wiki/WebKit%20Team, viewed 10 June 2011.

[146] WebSocket.org: What is WebSocket?. http://websocket.org/index.html, viewed 10 June 2011.

[147] WHATWG: CanvasContexts. http://wiki.whatwg.org/wiki/CanvasContexts, viewed 10 June 2011.

[148] WHATWG: DonŠt browsers need a target to work their implementations towards, even if itŠs a snapshot that is essentially arbitrary?.
http://wiki.whatwg.org/wiki/FAQ#Don.E2.80.99t_browsers_need_a_target_to_work_their_implementations_towards.2C_even_if_it.E2.80.99s_a_snapshot_that_is_essentially_arbitrary.3F, viewed 13 June 2011.

[149] WHATWG: Forking. http://wiki.whatwg.org/wiki/Forking, viewed 10 June 2011.

[150] WHATWG: How does the WHATWG work?. http://wiki.whatwg.org/wiki/FAQ#How_does_the_WHATWG_work.3F, viewed 9 June 2011.

[151] WHATWG: How should tool developers, screen reader developers, browser vendors, search engine vendors, and other implementors interact with the WHATWG?. http://wiki.whatwg.org/wiki/FAQ#How_should_tool_developers.2C_screen_reader_developers.2C_browser_vendors.2C_search_engine_vendors.2C_and_other_implementors_interact_with_the_WHATWG.3F, viewed 9 June 2011.

[152] WHATWG: How should tool developers, screen reader developers, browser vendors, search engine vendors, and other implementors interact with the WHATWG?. http://wiki.whatwg.org/wiki/FAQ#What_are_the_various_versions_of_the_spec.3F, viewed 9 June 2011.

[153] WHATWG: Is participation free?. http://wiki.whatwg.org/wiki/FAQ#Is_participation_free.3F, viewed 9 June 2011.

[154] WHATWG: Is this HTML5?. http://www.whatwg.org/specs/web-apps/current-work/multipage/introduction.html#is-this-html5?, viewed 9 June 2011.

[155] WHATWG: Mailing List. http://www.whatwg.org/mailing-list, viewed 9 June 2011.

[156] WHATWG: Specification annotation system documentation. http://www.whatwg.org/specs/web-apps/current-work/status-documentation.html, viewed 9 June 2011.

[157] WHATWG: Web Hypertext Application Technology Working Group Charter. http://www.whatwg.org/charter, viewed 9 June 2011.

[158] WHATWG: What does "Living Standard" mean?. http://wiki.whatwg.org/wiki/FAQ#What_does_.22Living_Standard.22_mean.3F, viewed 9 June 2011.

[159] WHATWG: What is HTML5?. http://wiki.whatwg.org/wiki/FAQ#What_is_HTML5.3F, viewed 9 June 2011.

[160] WHATWG: What's this I hear about 2022?. http://wiki.whatwg.org/wiki/FAQ#What.27s_this_I_hear_about_2022.3F, viewed 9 June 2011.