Sustainable Technologies for Mobile Applications

Approved: ___________________________ Date: ________________

Prof. Dr. Christoph Wentzel - Committee Chair

Approved: ___________________________ Date: ________________

Prof. Qi Yang, Ph.D. - Committee Member

Approved: ___________________________ Date: ________________

Prof. Dr. Bernhard Kreling - Committee Member
Sustainable Technologies for Mobile Applications

A Thesis presented to the Graduate Faculties

University of Wisconsin Platteville (UWP) and

University of Applied Sciences Darmstadt (h_da)

In Partial Fulfillment of the Requirement for the Degree

Master of Science

By

Simon Heckmann

Committee Chair: Prof. Dr. Christoph Wentzel (h_da)
Committee Member: Prof. Dr. Bernhard Kreling (h_da)
Committee Member: Prof. Qi Yang, Ph.D. (UWP)

Start date: 02/16/2011
End date: 08/16/2011
Statement of Authorship

I hereby declare that this thesis is written on my own solely by using the sources quoted in the thesis body. All contents (including text, figures, and tables), which are taken from sources, both verbally and in terms of meanings, are clearly marked and referenced.

The thesis has not been previously submitted to any other examination board, neither in this nor in a similar form.

___________________________________
Simon Heckmann
Acknowledgments

First of all I would like to thank my advising professors, Dr. Wentzel, Dr. Yang and Dr. Kreling for helping me with this thesis. The great discussions about my ideas, results and findings helped me a lot during the entire creation process of this document.

Secondly, I owe special thanks to my supervisors and colleges at Merck for their great suggestions and support. Especially, I would like to thank Dr. Michaela Bühler and Matthias Feldmann for their professional guidance and continuous motivation to tackle the challenges of this thesis. Furthermore, I want to thank Matthias Schwab, who never got tired of answering my continuous questions about the current Merck Toolbox implementation.

Furthermore, I want thank everyone who spent their valuable time proofreading and correcting this thesis.

Last but not least, I would like to thank my family in particular my parents and my fiancé Christine for their great support. Your love, open ears and continuous motivation kept me going. Thank you for seeing the best in me and being there for me no matter what!

You all made it happen! Thank you!
English Abstract

With the continuously increasing availability of powerful mobile devices such as smartphones, tablets and notebook computers it is now possible to provide sophisticated software applications that can be used anywhere and anytime. However, when developing such mobile applications, developers find themselves confronted with a broad variety of different operating systems and hardware platforms. Unfortunately, each of these comes with support for different programming languages, development frameworks and libraries. An application that runs on one device can often not be easily run on another. Therefore, the process of bringing an application to many different mobile devices is often a time consuming and expensive endeavor.

As this thesis shows, using standardized and open web technologies such as the HyperText Markup Language, Cascading Style Sheets and JavaScript provides a feasible way to develop mobile applications for a broad variety of different mobile platforms in a cost-efficient way. Nevertheless, until today such open standard web applications are not yet entirely capable of competing with native applications or web applications with plugins as they still lack some capabilities. Fortunately, this is about to change as many new upcoming web standards and specifications are about to address these limitations.

This thesis discusses why such web applications support nearly every available platform and how the upcoming standard candidates will allow open web applications to compete with native and plugin-based approaches and thereby save money by reducing development, deployment and maintenance costs.
German Abstract


In dieser Arbeit wird ausführlich diskutiert, wie es Webanwendungen auf eine einfache Art und Weise gestatten, plattformübergreifend zu entwickeln. Weiterhin wird erläutert, wie die angehenden Standards webbasierte Anwendungen ermöglichen, die in der Lage sind, es mit nativen oder Plugin-basierten Ansätzen aufzunehmen und wie gleichzeitig Entwicklungs-, Bereitstellungs- und Wartungskosten reduziert werden können.
# Table of Contents

## 1. Introduction
1.1. Goal
1.2. Differentiation
1.3. Thesis Overview

## 2. Mobile Application Development
2.1. Classification of Mobile Computing Devices
2.2. Development platforms
   2.2.1. Operating Systems
   2.2.2. Programming Languages and Frameworks
2.3. Summary

## 3. Web Applications
3.1. Architecture of Web Applications
3.2. Standardization Organizations
   3.2.1. World Wide Web Consortium
   3.2.2. Web Hypertext Application Technology Working Group
   3.2.3. Khronos Group
   3.2.4. ECMA International
3.3. Terminology
3.4. Web browsers
3.5. Future Web Standard Candidates
   3.5.1. Presentation tier
   3.5.2. Logic tier
   3.5.3. Data tier
   3.5.4. Featured Technologies
3.6. Prefixing
3.7. Compatibility Tests and Ratings
# Table of Contents

3.8. Summary ................................................................................................................. 48  

4. Feasibility Analysis ......................................................................................... 50  
   4.1. Merck Toolbox ............................................................................................... 50  
   4.2. Implementing the Prototype ........................................................................ 51  
   4.3. Open Issues .................................................................................................... 56  
   4.4. Platform Independence .................................................................................. 57  
   4.5. Web Applications on iOS ............................................................................. 60  
   4.6. Summary ......................................................................................................... 61  

5. Conclusion and Outlook .................................................................................. 63  
   5.1. Unfinished Standards .................................................................................... 63  
   5.2. Microsoft Internet Explorer .......................................................................... 64  
   5.3. Roadmaps and Previews .............................................................................. 66  
   5.4. Security concerns .......................................................................................... 68  
   5.5. Application Stores ......................................................................................... 69  
   5.6. Improving the Prototype ............................................................................... 71  
      5.6.1. Disk Space Quota Management .............................................................. 71  
      5.6.2. Data Encryption .................................................................................... 73  
      5.6.3. Alternative Document Format .............................................................. 74  
   5.7. Summary ......................................................................................................... 76  

List of Abbreviations ............................................................................................ 78  

Appendix ................................................................................................................. 80  
   Installing and Running the Prototype ................................................................. 80  
   Merck Toolbox ..................................................................................................... 81  

Bibliography .......................................................................................................... 88
Table of Figures

**Figure 1:** High-level comparison of performance versus mobility of mobile devices ............... 7
**Figure 2:** Different architectures for client-server applications ........................................... 20
**Figure 3:** High-level overview of the W3C standardization process ..................................... 23
**Figure 4:** The ECMA International standardization process (ECMA International, 2011) ...... 26
**Figure 5:** Public perception of the term HTML5 in contrast to the actual definition .......... 28
**Figure 6:** Popular web browsers and their for web applications relevant core components. 32
**Figure 7:** Current architecture of the Adobe Flash version of the Merck Toolbox ............... 51
**Figure 8:** Prototype architecture of HTML version of the Merck Toolbox ............................. 53
**Figure 9:** Feasibility rating of new web technologies support on notebooks ...................... 59
**Figure 10:** Feasibility rating of new web technology support on smartphones and tablets ... 60
**Figure 11:** Comparison of web browser features and market share ..................................... 65
**Figure 12:** Merck Toolbox welcome screen with animated clock (Flash) ............................ 81
**Figure 13:** Customizable favorites on the left and category browser on the right (Flash) ..... 81
**Figure 14:** File browser with documents and video files in different languages (Flash) ....... 82
**Figure 15:** View videos and documents directly within the application (Flash) .................... 82
**Figure 16:** Merck Toolbox welcome screen with animated clock (HTML) .......................... 83
**Figure 17:** Customizable favorites on the left and category browser on the right (HTML) .... 83
**Figure 18:** File browser with documents and video files in different languages (HTML) ....... 84
**Figure 19:** View videos and documents directly within the application (HTML) ................. 84
**Figure 20:** Download cart for downloading selected files to the local hard drive (HTML) ....... 85
**Figure 21:** Monitoring the download progress with pop-up notification on success (HTML) 85
**Figure 22:** Offline mode showing only local files if no connection is available (HTML) ........ 86
**Figure 23:** Merck Toolbox running in full screen mode on an Apple iPad (HTML) ............... 86
**Figure 24:** Merck Toolbox prototype running on a Motorola Xoom (HTML) ........................ 87
List of Tables

Table 1: Comparison of popular hardware devices................................................................. 8
Table 2: Estimated market share of smartphone operating systems for the year 2011 ........... 10
Table 3: Estimated market share of notebook operating systems for the year 2011 ........... 11
Table 4: Overview of available development environments on various platforms.............. 16
Table 5: Candidates for future standards and specifications .................................................. 45
1. Introduction

The Merck Group is a global-operating chemical, pharmaceutical and life science company with its headquarter in Darmstadt, Germany. As of December 2010 Merck had approximately 40,000 employees and subsidiaries for production and distribution in 67 countries worldwide. In 2010 the global revenue was nearly 10 billion euro. The Merck Group is divided into two distinct divisions: Merck Chemicals and Merck Pharmaceuticals. (Merck KGaA, 2011)

In the past Merck strictly controlled the kind of hardware devices, which were allowed for usage. By enforcing policies that only permitted a certain manufacturer for company notebooks and phones, the number of platforms could be easily restrained. Every software product and every application was developed for and maintained on a very small number of different systems, which helped to keep costs under control. In those days sales representatives were equipped with a standard notebook and a simple mobile phone.

However, in the past five years the segment of mobile hardware has rapidly grown, which has led to an increasing demand for additional mobile hardware. According to the research company Gartner in the second quarter of 2006 approximately 230 million mobile phones were sold (Gartner, Inc., 2007). Now, five years later, more than 428 million devices could be shipped within the same timeframe (Gartner, Inc., 2011a). In addition the tablet market has increased by 45.1% during last year (IDC, 2011b).

For a company like Merck, which relies on field sales as primary distribution channel, the growing importance of new mobile devices is significant. Sales representatives, once only equipped with a notebook, now strive to use other more convenient mobile technologies such as different smartphones or the newly introduced tablet computers, such as Apple’s iPad. Today Merck is confronted with the business needs to support a much broader range of hardware platforms and operating systems.
1. Introduction

Unfortunately, designing software for a broader variety of platforms is not trivial as every platform comes with a unique eco system for application development. As each vendor tries to manifest their market position these different eco systems are usually not compatible and require using different APIs and maybe even different programming languages (see section 2.2.2).

Of course this dilemma is not limited to business applications. The continuous growth of the smartphone and tablet market has also led to changes in the consumer market. Consequently, a software product has to be developed on more different devices and platforms to reach the same number of customers it did reach five years ago.

When trying to make an application available to the majority of users, it has to be developed for many different platforms and operating systems (compare section 2.2.1). This rapidly increases the development costs, the deployment overhead and the maintenance expenses. Not only does developing a mobile application require a great number of developers specializing in different hardware platforms and programming languages, it also requires several different development tools and development hardware. Hence, keeping the application consistent and fully featured across all major platforms is a time consuming and expensive task (Maxwell, 2011; Brown, 2009). It is therefore very important for application developers and software vendors to find a cost-efficient and convenient way to develop mobile applications for all relevant mobile platforms.

1.1. Goal

This thesis will analyze and compare different technologies for mobile application development, and will show that web applications built using only standardized and open technologies such as HTML, CSS and JavaScript promise to provide a cost-efficient and convenient way to develop mobile applications that run on all major mobile platforms and operating systems.
1. Introduction

The focus of this thesis is to analyze the capabilities of current and future web standard candidates and discuss whether they are capable of replacing native applications as well as web applications that require additional plugins. To examine the potentials and risks of this new web platform a prototype re-implementation of a current Merck application will be used. This application is currently only available on a limited set of operating systems, but should be brought to a much broader range of devices. Re-implementing this prototype with only open and standardized web technologies helps to demonstrate to which extent open web applications are able to replace other approaches today and in the future.

The conclusion of this thesis provides an estimate for the future of open standard web applications and summarizes open questions that might need to be addressed before such web applications are fully applicable for current and future business uses.

1.2. Differentiation

Although on the first glance it might occur that the future of the web strongly depends on when all browser manufacturers will have implemented the proposed features, the considerations within this thesis go beyond that. Even though the adoption speed of browser vendors could have an impact on the development speed of the web as a platform, it seems that eventually all browsers will agree to implement the standardized features. It is therefore far more interesting to see the chances and insufficiencies of the actual standards than the chances and insufficiencies of the current browser implementations. Nevertheless, it might be necessary to touch the topic of browser compatibility within this thesis, but the main focus is not on browser features, as these change on a daily basis. The emphasis of this thesis therefore is not on providing compatibility charts, but on a detailed analysis of current web standards and future web standard candidates.
1. Introduction

1.3. Thesis Overview

The rest of this thesis is divided into the following chapters:

**Mobile Application Development:** This chapter introduces the reader to the terminology and concepts that are needed to understand this thesis. It presents a classification of mobile devices and provides a detailed market analysis on mobile platforms, operating systems and available programming languages and frameworks for mobile application development.

**Web Applications:** In this section the concepts of web applications are described. This chapter also lays the foundation for understanding the potential of standardized and open web technologies by outlining upcoming standards and summarizing the essential standardization organizations and their processes.

**Feasibility Analysis:** After the general concepts behind web applications and future standards have been introduced, this chapter discusses the implementation of a prototype application using some of the new web technologies and outlines advantages and disadvantages of the new technologies.

**Conclusion and Outlook:** The last chapter of this thesis provides a summary of the previous chapters and evaluates the results of the feasibility analysis. The chapter concludes with an outlook on the future of the web as a platform.
2. Mobile Application Development

Before diving into the topic it is important to deliver a suitable classification of mobile computing devices and perform a detailed market analysis of frameworks and platforms for mobile application development.

2.1. Classification of Mobile Computing Devices

To achieve a common understanding of mobile applications it is required to analyze and characterize the different classes of mobile computing devices. A mobile application is considered to be software that can be run on such devices.

The main aspect that distinguishes mobile computing devices from regular stationary computing devices is the aspect of permanent mobility, which means “users can conduct business anytime and anywhere” (Nah, Siau, & Sheng, 2005). Rajeswari Malladi and Dharma Agrawal provide a similar definition, which states: “Mobile computing means continuous accessibility to the user” (Malladi & Agrawal, 2002).

Based on these definitions a mobile application has to provide all its functionality on the device itself or, in case it is designed as client-server software, has to provide communication mechanisms and has to allow using the application even if no network connection is available. To fulfill this, a mobile device has to host the application, hold the required data and provide wireless communication to a remote network. Devices, which can run custom software applications and have methods for user interaction and data storage, can be classified as follows:

**Smartphone:** *Smartphones* are the combination of regular phones and Personal Digital Assistant (PDA). They combine the communication abilities of standard mobile phones and the possibility to browse the web, read and write emails and run a broad variety of custom applications from the PDA. Smartphones are controlled using a build in
2. Mobile Application Development

keyboard or a touch-screen. (Charlesworth, 2009; Phone Factor, LLC, 2011; TechTerms.com, 2010)

Tablet: According to a glossary provided by Gartner a tablet is a lightweight computing device that “is operated by direct screen contact via a pen or touch interface” (Gartner, Inc., 2011b). Those tablets are relatively new category popular since the introduction of Apple’s iPad and now adopted by many other hardware manufacturers (Ogg, 2010).

However, the idea of tablets has been around before the iPad (TechTarget, 2011b). Before that, those devices where referred to as tablet-PCs, which “meet all criteria for mobile PCs but are equipped with a pen and on-screen digitizer and are configurable into a tablet format” (Gartner, Inc., 2011b).

Notebook: A notebook or laptop is “a computer system designed for portability” (Gartner, Inc., 2011b). They share a very similar architecture as regular desktop computers, but due to their compact construction, offer a much higher mobility then stationary personal computers. Notebooks are controlled through mouse resp. touchpad and keyboard, which are built directly into the device. Furthermore notebooks do not feature a touch-screen. (TechTarget, 2011a)

A category of devices which is very similar to notebooks are netbooks. Both belong to the same device family, but netbooks have a smaller form factor to increase the aspect of mobility.

Of course not all mobile devices can be clearly assigned to one individual class. There are hybrid devices that tend to fit in more then one category, or can be seen as connector or transition between two different categories. However, the above-mentioned classification provides a decent categorization of the spectrum of mobile devices.
2. Mobile Application Development

**Figure 1** shows the four main characteristics of mobile computing devices and how the already mentioned three device classes perform.

![Figure 1: High-level comparison of performance versus mobility of mobile devices](image)

**Mobility:** Smartphones are small, lightweight devices that fit in every pocket. It is therefore very easily brought everywhere (e.g. Apple iPhone, approx. 140g). Tablets are usually bigger than smartphones, but are still convenient to carry (e.g. Apple iPad, approx. 600g). Notebooks are the biggest and heaviest mobile computing devices (e.g. Dell Latitude, approx. 2kg).

**Storage capacity:** Due to their limited size, smartphones have only limited persistent storage capacity (e.g. Apple iPhone, up to 32GB). Although tablets exceed the storage limitations of smartphones (e.g. Apple iPad, up to 64GB) they still have far less capacity than notebooks with nearly unlimited persistent storage (e.g. Dell Latitude, up to 500GB).
2. Mobile Application Development

**Screen size:** As already mentioned the small size of smartphones is also responsible for their screen size (e.g. Apple iPhone, 3.5 inch). Notebooks, which have a full-size keyboard, host much larger screens (e.g. Dell Latitude, up to 15.6 inch). Tablets are usually ranked between smartphones and notebooks (e.g. Apple iPad, 9.7-inch).

**Computational power:** Modern Notebooks are equipped with multi-core processors with high clock speed and large amounts of internal memory (e.g. Dell Latitude, up to 4 cores at 3.3 GHz with 8GB RAM). Smartphones have much more limited capacities (e.g. Apple iPhone, single-core at 1 GHz with 512MB RAM). Tablets again rank somewhere between the two (Apple iPad, dual-core at 1GHz with 512MB RAM).

Battery runtime of notebooks, tablets and smartphones is no longer an issue due to today’s high capacity batteries and power energy efficient CPUs devices can be powered for eight and more hours.

The following table summarizes the paragraphs above and provides a direct comparison of the different hardware devices:

<table>
<thead>
<tr>
<th></th>
<th>Smartphone (e.g. Apple iPhone)</th>
<th>Tablet (e.g. Apple iPad)</th>
<th>Notebook (e.g. Dell Latitude)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mobility</strong></td>
<td>approx. 140g</td>
<td>approx. 600g</td>
<td>approx. 2000g</td>
</tr>
<tr>
<td><strong>Storage capacity</strong></td>
<td>up to 32GB</td>
<td>up to 64GB</td>
<td>up to 500GB</td>
</tr>
<tr>
<td><strong>Screen size</strong></td>
<td>3.5 inch</td>
<td>9.7 inch</td>
<td>15.6 inch</td>
</tr>
<tr>
<td><strong>Computational power</strong></td>
<td>single-core at 1 GHz with 512MB RAM</td>
<td>dual-core at 1 GHz with 512MB RAM</td>
<td>up to 4 cores at 3.3 GHz with 8GB RAM</td>
</tr>
</tbody>
</table>

**Table 1:** Comparison of popular hardware devices
2. Mobile Application Development

2.2. Development platforms

When starting development of an application or software system one of the key decisions that has to be made is to specify the hardware platforms and operating systems on which the final software shall be executable. Depending on this choice, the number of potential programming languages or frameworks might be limited. Certain compilers, runtime environments or APIs are not available on any operating system or device. Therefore, this chapter will give an overview of possible platforms, programming languages and related frameworks which are relevant for mobile application development.

2.2.1. Operating Systems

As outlined in chapter 2.1 mobile devices can be roughly divided into three groups: notebooks, tablets and smartphones. While smartphones and notebooks work with individually developed operating systems, tablets currently run on modified or customized versions of either smart phone or notebook operating systems.

According to a report published by Gartner (Gartner, Inc., 2010), the most important players for smart phone and tablet operating systems in 2011 are Nokia’s Symbian OS with 34.2% market share, Google’s Android with 22.2%, Apple’s iOS with 17.1%, Research in Motion’s (RIM) Blackberry OS with 15.0% and Microsoft’s Windows Phone (formerly Windows Mobile) with 5.2%. All other smart phone operating systems reach a share of 6.3%.

As these values are calculated based on statistical projection and estimated sales numbers, an analysis of the worldwide smart phone market released by the International Data Corporation (IDC) infers slightly different market shares for 2011 (IDC, 2011a): Google’s Android with 39.5%, Nokia’s Symbian OS with 20.9%, Apple’s iOS with 15.7%, RIM’s Blackberry OS with 14.9%, Microsoft’s Windows Phone with 5.5% and others with 3.5%.

- 9 -
2. Mobile Application Development

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Gartner</th>
<th>IDC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nokia Symbian OS</td>
<td>34.2%</td>
<td>20.9%</td>
</tr>
<tr>
<td>Google Android</td>
<td>22.2%</td>
<td>39.5%</td>
</tr>
<tr>
<td>Apple iOS</td>
<td>17.1%</td>
<td>15.7%</td>
</tr>
<tr>
<td>RIM Blackberry OS</td>
<td>15.0%</td>
<td>14.9%</td>
</tr>
<tr>
<td>Microsoft Windows Phone</td>
<td>5.2%</td>
<td>5.5%</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>93.7%</strong></td>
<td><strong>96.5%</strong></td>
</tr>
</tbody>
</table>

*Table 2: Estimated market share of smartphone operating systems for the year 2011*

Although, both reports come to different rankings, the top five operating systems with a cumulated market share of 93.7% to 96.5% stay the same. All operating systems listed as others are individually seen as to unimportant and will therefore not be explicitly covered within this thesis.

The market share of notebook operating systems (Table 3) is not as heavily researched as the smartphone market. This makes it difficult to get a decent indicator on the usage share distribution for notebook devices. However, analyzing the user agent string of web browsers can give an approximation about the underlying operating system in use.

Net Applications, a software development company specialized on applications for webmasters and eMarketers, continuously releases usage share statistics for Internet technologies, which can be used to determine the usage share of notebook operating systems. In March 2011 the market share is presented as follows (Net Applications, 2011a): With almost 94% Microsoft’s Windows family is still the clear market leader. Apple’s Mac OS X comes in second with roughly 5%. All available Linux distributions combined barely exceed 1%.
2. Mobile Application Development

A similar distribution can be seen when taking the browser statistics for March 2011 provided by W3Counter, a widely deployed statistic tool for web administrators, as base for calculations (Awio Web Services LLC, 2011): Microsoft Windows reaches 89%, Apple’s Mac OS X brings approximately 10% to the books and all available Linux distributions sum up to a bit more then 1%.

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Net Applications</th>
<th>W3Counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Windows</td>
<td>94%</td>
<td>89%</td>
</tr>
<tr>
<td>Apple Mac OS X</td>
<td>5%</td>
<td>10%</td>
</tr>
<tr>
<td>Linux or similar</td>
<td>1%</td>
<td>1%</td>
</tr>
<tr>
<td>Total</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

*Table 3:* Estimated market share of notebook operating systems for the year 2011

Based on this data the most important operating systems in the mobile device market are Microsoft Windows, Apple Mac OS X, Linux or similar, RIM Blackberry OS, Apple iOS, Google Android, Microsoft Windows Phone and Nokia Symbian. Therefore, only these eight operating systems are considered relevant and will be analyzed as base for mobile application development.

2.2.2. Programming Languages and Frameworks

After introducing the list of all relevant operating systems, this chapter will provide an overview of potential programming languages and frameworks and their availability on the listed platforms.

**C/C++:** When developing application for notebooks or desktop computers, C and C++ are supported by a broad variety of different frameworks and compilers. A suite com-
2. Mobile Application Development

Monly used is the GNU Compiler Collection capable of compiling on Windows, Linux (Free Software Foundation, Inc., 2011) and Mac OS X (Apple Inc., 2004) or Microsoft’s Visual C++ (Microsoft Corporation, 2011h) designed especially for Windows. Unfortunately, C and C++ do not directly include a possibility to write applications with graphical user interface directly. To accomplish this a dedicated framework or toolkit has to be used. While frameworks like the GIMP Toolkit (GTK+) (The GTK+ Team, 2011) or Qt (Nokia, 2011a) are available for all three operating systems, others such as Microsoft’s Visual C++, which uses .NET or MFC, or Apple’s Cocoa (Apple Inc., 2011d) are only available for Windows and respectively, Mac OS X.

For smart phones and tablet operating systems, the development support for C/C++ is even more limited. Although Nokia’s Symbian or Apple’s iOS applications can be implemented using C/C++, both platforms require different GUI frameworks. C/C++ applications for Symbian can utilize the already mentioned Qt toolkit, whereas iOS applications can only use Cocoa Touch, a framework similar to Apple’s Cocoa for Mac OS X. For Google’s Android, RIM’s Blackberry OS and Windows Phone C or C++ development is not supported at all.

It can be seen that although some of the platforms support C and C++ code for their application the platforms are only interoperable to a certain extent. While code parts might still be reused as soon as it comes to frameworks and toolkits, each platform requires individual adjustment.

C# and .NET: Because C# and .NET are both technologies developed by Microsoft, the only platform that provides full support for these technologies is Windows. Nevertheless, the huge success of the programming language C# has inspired third party developers to start Mono, a free and open-source project aiming to bring C# and the .NET runtime environments to other platforms such as Mac OS X, Linux and even Android (Novell, Inc., 2011a). With MonoTouch (Novell, Inc., 2011b) there is also a possibility
2. Mobile Application Development

to develop C# applications for iOS, but due to Apple’s restrictions (Apple Inc., 2011c, Section 3.3.2) the final C# code has to be compiled to machine code, which is unusual for .NET applications. On Windows Phone Microsoft only provides support for the programming language C#. A .NET runtime is not available. RIM’s Blackberry OS is currently not supported at all.

Although with the Mono project there is a broader support for .NET and C#, it has to be kept in mind that Mono is only tolerated by Microsoft but not actively supported. Therefore, there is no guarantee that Mono will provide exactly the same functionality as .NET does for Windows. Similar holds through for iOS, because MonoTouch applications must be compiled against native iOS APIs. As soon as Apple changes this API, MonoTouch has to be updated as well.

**Objective C:** Thanks to the GNU Compiler Collection there is an Objective C compiler that can generate executable applications for Windows, Linux and Mac OS X. Unfortunately, similar to C and C++, Objective C does not come with any GUI toolkit or framework. Such frameworks are currently only available for Mac OS X (Cocoa) and iOS (Cocoa Touch). While both frameworks share a common base, they are not fully interoperable. An application designed for iOS cannot directly run on Mac OS X, although larger parts of the source code can be reused. For any other mobile platform Objective C is not supported.

**Java:** On Microsoft Windows, Apple Mac OS X and Linux systems the Java Runtime Environment (JRE) provides a feature-rich and high performing virtual machine for convenient and interoperable application development (Oracle Corporation, 2011). Any Java application relying on standard features of the Java virtual machine can be executed on any of the above three operating systems without any adaptations.

Developing applications for BlackBerry OS, Android or Symbian OS can also be done using Java technologies. RIM (Research In Motion Limited, 2011b), Google (Google
Inc., 2011i) and Nokia (Nokia, 2011b) provide comprehensive APIs and a Java runtime environment directly inside their operating systems. However, these APIs are completely vendor specific and do not allow simple Android applications to run on BlackBerry OS or Symbian and vice versa. On top of that, the provided APIs are also not available on the desktop and notebook implementations of the JRE, which is why Android, BlackBerry and Symbian Java applications cannot directly run on Windows, Mac OS X or Linux.

**Silverlight:** Microsoft Silverlight is web browser extension intended to provide additional APIs and functionality not directly available in every web browser. Currently, Microsoft maintains a Silverlight implementation for Windows and Apple’s Mac OS X (Microsoft Corporation, 2011b). Due to the fact the Microsoft does not provide a Silverlight implementation for Linux, the Mono project has created a Silverlight compliant implementation called Moonlight (Novell, Inc., 2011c). Similar to the Mono .NET implementation, Moonlight is a third party implementation and might therefore not always be guaranteed to be compatible with the latest Silverlight release.

Beyond the use in web browsers Microsoft has also implemented Silverlight for Windows Phone (Microsoft Corporation, 2011c) & Symbian OS (Microsoft Corporation, 2011g) to serve as application development platform. Silverlight implementations are at the moment not available for any other operating system.

**Flash:** The Adobe Flash Platform, similar to Microsoft Silverlight, is a technology designed to deliver APIs for rich Internet applications. To accomplish this the Adobe Flash Player is installed as a browser plugin. On notebooks or desktop computers Adobe Flash is available on all three operating systems (Adobe Systems Inc., 2011c). On smartphones and tablets however some compromises have to be made:

For Android devices a fully compatible Flash Player is only available for Android version 2.2 and above. As of June 2011 over a quarter of all installed Android platforms
2. Mobile Application Development

still runs a version smaller then 2.2 (Google Inc., 2011f) and are therefore incapable of running Flash applications. Additionally Adobe lists some minimal hardware requirements to run their Flash Player. A list of all supported mobile devices and tablets is available on the Adobe website (Adobe Systems Inc., 2011a).

On iOS Flash applications are not supported directly. However, Adobe provides a packager application that allows transforming Flash applications into native iOS applications (Bansod, 2011). This procedure is similar to MonoTouch and provides the same drawbacks.

For RIM’s BlackBerry OS devices Adobe has officially announced full-featured Flash support. However, Adobe currently lists no single Blackberry smartphone on which Flash is available. Only the BlackBerry playbook, RIM’s first tablet device has the Adobe Flash Player preinstalled (Research In Motion Limited, 2011c).

Devices running Symbian OS are only equipped with Adobe’s Flash Player Lite (Adobe Systems Inc., 2011b), a version od Adobe Flash with reduces featured and capabilities. Not all current Flash applications might be able to run with the limited feature set provided by the lite version.

**HTML, CSS and JavaScript:** Due to the high standardization of HTML, JavaScript and CSS and the availability of at least one web browser on every of the discussed platforms, web technologies are currently the only way to develop cross-platform applications running on all relevant modern operating system. A detailed description of web applications built using HTML, CSS and JavaScript is provided in chapter 3.

To summarize the results of the mobile platform analysis Table 4 gives a high-level overview of the available programming languages respectively frameworks:
## 2. Mobile Application Development

<table>
<thead>
<tr>
<th></th>
<th>C/C++</th>
<th>C# and .NET</th>
<th>Objective C</th>
<th>Java</th>
<th>Silverlight</th>
<th>Flash</th>
<th>HTML, CSS, JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple iOS</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
</tr>
<tr>
<td>Apple Mac OS X</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
</tr>
<tr>
<td>RIM BlackBerry OS</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
</tr>
<tr>
<td>Google Android</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
</tr>
<tr>
<td>Microsoft Windows</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
</tr>
<tr>
<td>Microsoft Windows Phone</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
</tr>
<tr>
<td>Nokia Symbian OS</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
</tr>
<tr>
<td>Linux or similar</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
<td>〇</td>
</tr>
</tbody>
</table>

〇 Fully supported 〇 Partially supported 〇 Not supported

### Table 4: Overview of available development environments on various platforms

#### 2.3. Summary

The main aspect of mobile applications is the permanent availability of its functionality to its user. A mobile device is therefore a piece of hardware that can conveniently be carried around. These devices can be grouped into three different categories: notebooks, tablets and smartphones. Each of these classes defers in its weight, its screen size, its computational power and its storage capacity.

When preforming a detailed market share analysis of smartphone and notebook operating systems eight relevant platforms can be identified. Microsoft Windows, Apple Mac OS X and Linux systems are the most important notebook operating systems. Microsoft Windows
2. Mobile Application Development

Phone, Apple iOS, Google Android, RIM Backberry OS and Nokia Symbian OS are most commonly used on smartphones. Tablets run on either notebook or smartphone operating systems.

For each platform and operating system there is a set of different programming languages and development frameworks that can be used to develop mobile applications. However, not many of these development environments are compatible to each other or are designed for development on more then one individual platform. As it turns out, only the open web technologies HTML, CSS and JavaScript are broadly supported on every major operating system and platform.
3. Web Applications

As shown in chapter 2.2 targeting multiple hardware platforms or multiple operating systems at the same time is not a trivial task as many technologies are limited to only a subset of hardware or software platforms. To overcome these limitations, open and standardized technologies such as HTML, CSS and JavaScript are currently the only way to develop applications across all major platforms.

While these technologies are currently mainly used in the context of client-server applications exchanging data over the Internet and using a web browser as client software, they are commonly referred to as web-based applications or just web applications. However, the term web application is currently not limited to the core web technologies HTML, CSS and JavaScript but also involves proprietary technologies such as Microsoft’s Silverlight or Adobe’s Flash platform. The disadvantage with these proprietary technologies however is that they are not supported by the web browser directly, but require a special plug-in which has to be individually developed for every web browser, every hardware architecture and every operating system. Therefore, this thesis will only focus on those web applications that solely utilize open standards such HTML, CSS and JavaScript and do not require proprietary addons or plugins to be installed. Hence, in this thesis the term web application is used as a short form for an open standard web application.

In addition to supporting a nearly unlimited set of different operating systems and hardware platforms, open standard web applications bring a number of other advantages. As web applications and their data are hosted on central servers and delivered over the Internet, development, maintenance and deployment of the application and its data and can be conveniently done in one central spot. This is therefore much less cost intensive than administration of locally installed applications. Furthermore, the central storage of data can help to eliminate time consuming and error-prone data replication and synchronization.
This way, web applications do not only provide a cost efficient form of cross-platform application development, but they also help to reduce deployment, administration and maintenance costs. A more detailed analysis of the benefits of web applications can be found in the Bachelor’s thesis “Online vs. Offline” (Heckmann, 2008).

This chapter provides an introduction into the general architecture of web applications and elaborates on the different standardization organizations and their processes, which are required to understand the different web standards and their origins. Additionally, all future standard candidates are introduced to show the new possibilities of web applications.

3.1. Architecture of Web Applications

To understand the new concepts and possibilities that are currently discussed for future HTML revisions, it is important to gain a common understanding of the architectural concepts of web applications. While there are many architectural patterns for client-server software application development, this chapter is not intended to go into detailed discussions about software designs and architectures but aims to give a high level overview. A commonly used design principle for client-server applications is the three-tier architecture (Tanenbaum & Van Steen, 2007):

The presentation tier is often also called user interface and is responsible for managing the user's interaction with the application. To accomplish this, it gives the users mechanisms to navigate through the application and offers functionality to enter, view and maintain all required data.

The logic tier provides the application’s actual functionality by processing the data based on certain rules or algorithms.

The data tier focuses on the persistent storage of the data used within the application. It provides mechanisms to store and share diverse types of information.
These three tiers can be conveniently distributed between the client and the server. Figure 2 illustrates the three most common possibilities.

In the simplest scenario the client application is only responsible for providing the user interface. All application logic is processed on the server side and only completely computed results are returned to the client for displaying. To use such an application, a permanent connection has to be available, as all required data and logic are only available on the server. Client-server applications following this design principle are referred to as **thin client**.

In contrast to thin clients **fat clients** implement all three tiers, which allows them to operate completely independent from the server as long as all required data is available in the local data tier. Even new data can be entered and cached on the client. As soon as data is missing or new data has to be distributed to other clients, a connection to the server has to be established and the local and the remote data storage has to be synchronized.

**Figure 2**: Different architectures for client-server applications
A third possible distribution of the applications components is to extend a thin client with the ability to perform a limited set of calculations without contacting the server. In such cases the amount of client-server interaction could be reduced, but for more complex operations or any read or write access to persistent data a connection to the server is still required.

With the first version of HTML it was not possible to execute any application logic within the browser, nor was it possible store any data on the client. Therefore with HTML 1.0 only thin client architectures were possible (Berners-Lee & Connolly, 1993). When HTML 4.0 was introduced, the new specification for the first time included the possibility to add JavaScript code to web pages (Raggett, Hors, & Jacobs, 1999). This way the thin client architecture could be extended to execute some rudimentary application logic within the browser. As it will be shown in later chapters now with the new version of HTML and many new JavaScript APIs on the horizon, for the first time it will be possible to model the entire range from thin to fat client with web applications.

### 3.2. Standardization Organizations

One of the reasons for the large interoperability of web applications is the comprehensive standardization of nearly every technology involved. The standardization organizations and standardization processes have a huge impact on the future development of web applications because they lay the foundation for all future possibilities. To understand the risks and potentials of web applications it is therefore very important to know the four relevant standardization organizations and their standardization process.

#### 3.2.1. World Wide Web Consortium

The most important organization in the area of web standards is the World Wide Web Consortium (W3C), led by the inventor of the Internet, Tim Berners-Lee. The organization currently consists of more then 300 members, including the vendors of all relevant web brows-
3. Web Applications

er such as Apple Inc., Google Inc., the Microsoft Cooperation, the Mozilla Foundation and Opera Software (W3C, 2011a). A full time staff has devoted itself to developing high quality web standards. To achieve this, the W3C brings members, staff and community contributors together to promote the development of consensus-based web standards (W3C, 2009).

Over the last two decades the W3C was responsible for well-known standards such as the Extensible Markup Language (XML), the Document Object Model (DOM), Web Services and the previous versions of HTML.

Before a new idea can be developed into a standard, the W3C defines a detailed standardization process that each proposal has to go through. Each potential standard, or in W3C terminology, a technical report, is maintained by a working group and has to go through a finite set of process steps called maturity levels at the W3C. A high-level overview of this process is outlined in Figure 3. Currently there are four different maturity levels defined. The following paragraphs provide a short overview over the maturity levels and the W3C standardization process. A detailed explanation of the individual process and process requirements can be found in the official W3C Process Document (Jacobs, 2005).

**Working draft (WD):** A working draft is the initial maturity level for any potential standard maintained at the W3C. Its purpose is to have the proposal publicly available for review by the community and other interested parties. When a new technical report enters this maturity level it has to be announced as First Public Working Draft. It then undergoes an arbitrary number of iterations until the responsible working group believes all relevant technical requirements are satisfied. The Last Call Announcement indicates that if no further objections are raised the technical report can pass over to the next maturity level.

**Candidate Recommendation (CR):** If the working group and interested parties believe that a technical report is ready for implementation it publishes a Call for Implementations. Thereby the technical report is from now on considered a Candidate Recommenda-
3. Web Applications

tion. When in this maturity level, the report only undergoes a small set of changes, for example removing parts that turn out to be very difficult to implement. This level can be skipped if the quality of a Working Draft is already high enough and at least two distinct implementations of every aspect of the technical report are available.

**Proposed Recommendation (PR):** After continues feedback from all involved parties and at least two mature and comprehensive implementations of the proposed standard, the technical report’s maturity level is called a Proposed Recommendation. The responsible working group can then Call for Review one last time, requesting significant support for the technical document.

**Recommendation (REC):** A technical report can emerge into a Recommendation as soon as it is significantly supported by the W3C team, the working group and the public. The W3C then proceeds with the Publication of a W3C Recommendation. From this point on the underlying technical report is considered a standard.

![Diagram](image)

**Figure 3:** High-level overview of the W3C standardization process

Naturally not all proposals or working drafts make their way into a Recommendation. Whenever a working group is unable to bring, e.g. due to serious implementation issues, or

- 23 -
3. Web Applications

sees no further benefit in taking a proposed standard to the next maturity level, it can always cancel the standardization process and release a Working Group Note stating the reason for the discontinuation.

3.2.2. Web Hypertext Application Technology Working Group

In 2004 the Web Hypertext Application Technology Working Groups (WHATWG) was founded after several employees of Apple Inc., the Mozilla Foundation and Opera Software were in disagreement with the W3C’s future roadmap for HTML (Pilgrim, HTML5: Up and Running, 2010). While the W3C’s plan was to continue the development of HTML all by themselves, the newly formed WHATWG wanted to focus on developing standards for web application development in close cooperation with real-world authors. While working on different standards for the next two years the W3C finally smoothed the way for a reunion by deciding to drop their own future HTML standard in favor of the version developed by the WHATWG.

Until today both organizations are working closely together on new web standards but still with slightly different approaches. While as already explained the W3C develops standards in a very democratic approach searching for consensus among all participating parties, the WHATWG has a distinct editor who ultimately decides what to include and what to leave out. Currently, Ian Hickson is the editor of both the W3C’s (Hickson, HTML5, 2011) and the WHATWG’s (Hickson, HTML Living Standard, 2011) future HTML standard.

Although the WHATWG standards cover many aspects of standards developed by the W3C, there are important differences in the standardization process (WHATWG, 2011). While the W3C has a set of individual technical reports for each distinct application areas, the WHATWG currently only maintains two different documents combining many different application fields. These two documents are the WHATWG HTML specification (Hickson, HTML Living Standard, 2011) and the Web Application specification (Hickson, Web Applications 1.0,
3. Web Applications

2011), which includes the full WHATWG HTML standard but also covers a much broader range of concepts from the area of web application development. Instead of defining the maturity level for the whole document altogether, each section or subchapter can have discrete states. The WHATWG takes account of this by calling its incremental model of standardization Living Standards.

Each section or chapter of a WHATG Living Standard can have one of the following states: idea yet to be specified, first draft, working draft, last call for comments, awaiting implementation feedback and implemented and widely deployed. As the names suggest these individual states can be loosely mapped to the corresponding W3C states.

3.2.3. Khronos Group

In contrast to the WHATWG or the W3C, the standards of the Khronos Group are not exclusively related to web technologies but cover a much broader range from the area visual computing (Khronos Group, 2011a). Presently only one standard maintained by the Khronos Group is related to web applications. This standard is detailed in chapter 3.5.

To contribute in the group’s standardization process, there are currently three different levels of participation (Khronos Group, 2010): Promoters, Contributors and Academic Contributors. Promoters are the most powerful members of the organization. They establish the work groups, participate in the technical specification and have the right to ratify the final standard. Contributors can only participate and vote within their working groups but have no voting rights aside from that. Finally, academic contributors, similar to ordinary contributors, can provide input for the technical specification but get no vote in any stage of the standardization process.

From the top five web browser developers only Apple Inc. is enrolled as promoter. The Mozilla Foundation, Google Inc. and Opera Software are currently listed as contributors while Microsoft is not a member at all.
3.2.4. **ECMA International**

ECMA International, formerly known as European Computer Manufacturers Association (ECMA), is a standardization organization, which similar to the Khronous Group, is responsible for a broad set of standards (ECMA International, 2011).

Currently only a small subset of its specifications is related to web applications or web technologies. Along with many others, the list of members includes Apple Inc. as *Associate Member*, Google Inc. and the Microsoft Cooperation as *Ordinary Members* and the Mozilla Foundation as *Not-for-profit Member*. Opera Software however is currently not listed as member of ECMA International.

The standardization procedure, as shown in *Figure 4*, is far more simple compared to the W3C’s process. After a closed working group called *Technical Committee* has outlined and agreed on a standard proposal, it can submit its final draft to the *General Assembly*, the highest authority of the association. Only Ordinary Members get a vote and each proposal must get at least a two-thirds majority to be approved as a new standard.

![Ecma Process](image)

*Figure 4*: The ECMA International standardization process (ECMA International, 2011)

3.3. **Terminology**

As this thesis focuses on future web standards and technologies, it is important to clarify some misinterpretation of some of the important terms in the area of web applications. Unfortunately, since its introduction the term HTML5 or HTML 5 has become an ambiguous
3. Web Applications

term, which has lead to a set of imprecise conclusions. To avoid confusion, this chapter will provide a suitable definition for all required terms related to web application technologies and standards.

Strictly speaking, the term HTML5 references a technical report that is currently maintained by the W3C as a Working Draft (Hickson, HTML5, 2011). Although this standard is still under development, the last call announcement has been made and the purpose of the standard has been clearly defined:

“This specification is limited to providing a semantic-level markup language and associated semantic-level scripting APIs for authoring accessible pages on the Web ranging from static documents to dynamic applications.” (Hickson, HTML5, 2011)

HTML5 does therefore not cover the definition of media-specific customization of presentation, which is usually done with CSS. Although the HTML5 specification describes mechanisms to integrate CSS styling into HTML files, CSS is defined by a set of completely independent technical reports.

Similar principles also apply to the programming language JavaScript. While the HTML5 provides mechanisms to include JavaScript files, the programming language itself is completely independent.

However, the related JavaScript APIs, which provide the interface between the JavaScript language and the web browser’s functionality are treated different. While most of the APIs are similar to CSS defined in completely separate specifications, a small number of APIs are directly included in HTML5. Usually, these included APIs are directly and solely related to a feature or tag in HTML5 and cannot exist without them. Some examples are given in chapter 3.5.

In contrast to the W3C’s HTML5, HTML 5 – with blank as delimiter between the letters and the version number – is the former name of the WHATWG’s HTML – Living standard
3. Web Applications

(Hickson, HTML Living Standard, 2011). While, as already stated, both standards are very similar HTML – Living standard does contain a set of features, which are not part of the HTML5 specification. Although HTML 5 has a received a new, more distinguishable name, older documents might still refer to it with its old name, which can lead to all kinds of confusion and provides a huge source for errors.

Unfortunately, the confusion and errors caused by the very similar spelling are not the only problem with the terminology. HTML5 has especially become a synonym for all kinds of future web standards even if they are not part of the real HTML5 standard described above. Therefore, the public perception of HTML5 is very different than the actual definition, and HTML5 is often understood as an umbrella term. Even the W3C promotes this ambiguity by including CSS styling and actually independent JavaScript APIs into their HTML5 Logo (W3C, 2011b).

To illustrate the different standards and interpretations Figure 5 provides an overview of the individual components.

Figure 5: Public perception of the term HTML5 in contrast to the actual definition
While everything within the surrounding shape with the dashed line in the public perception is often considered to be HTML5, in fact only the centered gray shape represents the actual HTML5 standard. The yellow shape represents the WHATWG’s HTML – Living standard, which includes all features listed in HTML5 and also provides additional concepts, which could potentially provide the base for a future revision of HTML (e.g. HTML6). Both HTML5 and HTML – Living standard define mechanisms to include or interact with related technologies such as Scalable Vector Graphics (SVG), the Mathematical Markup Language (MathML), and as already mentioned JavaScript and CSS. Additionally, there are large sets of JavaScript APIs maintained by different organizations.

Even it were legitimate to use HTML5 as a synonym for future web standards, it will not be used as such in this thesis. The term HTML5 only describes the real W3C specification with this name. In all other cases the phrase future web standards will be used to describe the combination of HTML5 and all other related specifications.

Another term that is often used together with web development is the eXtensible HyperText Markup Language (XHTML). From the beginning of the web HTML was based on the Standard Generalized Markup Language (SGML). However, with the growing success of XML the desire of an XML compatible version of HTML grew, which finally lead to XHTML, a reformulation of HTML 4 (Pemberton, et al., 2002). While the W3C tried to continue the development of XHTML, the WHATWG focused on regular HTML development. As the name HTML5 indicates, today the W3C and the WHATWG have agreed to focus on HTML instead of XHTML. Nevertheless, there also is a XHTML5 specification available, but this is not the primary standard (Hickson, HTML5, 2011, Section 9).

3.4. Web browsers

One of the most essential parts of any web application is the web browser. A web browser, or just browser, is a software application that serves as a gateway to any web page or web
3. Web Applications

application. To display a web page the web browser connects to the server and downloads the required resource files. The rendering engine, one of the core components of the browser, is then responsible for generating the visual appearance of the web page based on the HTML and CSS files. A dedicated JavaScript Engine executes script files included in the web page. It is important to note that these JavaScript Engines do not provide the JavaScript APIs available to web applications developers. These APIs are implemented as a subpart of the rendering engine.

For many of the relevant platforms and operating systems mentioned above, there is a broad variety of different web browsers available. However, these browsers vary in market share and spread. The following web browsers fulfill the criteria of being available on most of the relevant platforms and have gained a market share of at least 1% according to StatCounter (StatCounter, 2011a; StatCounter, 2011b). Although there are many other statistics about web browser market share, StatCounter is the only one with freely available information on mobile browsers.

Firefox: Firefox and Firefox Mobile are two open source browsers developed by the Mozilla Foundation. Firefox is currently available on Microsoft Windows, Apple Mac OS X and Linux (Mozilla Foundation, 2011a). Mobile Firefox can be installed on Android devices and Nokia’s mobile phone N900 (Mozilla Foundation, 2011d).

Chrome & Android browser: Chrome is a web browser created by Google and developed based on the open source browser Chromium. Chrome releases are available on Windows, Linux and Mac OS X (Google Inc., 2011e). As the Android operating system is also developed by Google, Android’s preinstalled browser is very similar to Google Chrome.

Safari: Safari and Mobile Safari are two similar browsers developed by Apple (Apple Inc., 2011b). While the desktop and notebook version of Safari runs on Windows and Mac OS X, the mobile version is only available on iOS devices.
3. Web Applications

**Internet Explorer:** Microsoft’s web browser named Internet Explorer is available on Microsoft platforms such as Windows and Windows Phone. Microsoft’s Internet Explorer is proprietary software and the source code is not publicly available.

**Opera:** Opera Software ASA currently distributes three different browsers: Opera for Windows, Mac OS X and Linux (Opera Software ASA, 2011a), Opera Mobile for Android and Symbian OS and Opera Mini for Android, Symbian OS, iOS and BlackBerry OS. While Opera and Opera Mobile are fully featured browsers, Opera Mini has only limited functionality. Instead of loading a web site or web application directly it is routed over a proxy server that pre-renders and compresses the website before sending it to the device (Opera Software ASA, 2011b). This provides a fast web browsing experience on devices with limited hardware resources but also restrains the web applications from using certain features.

**BlackBerry and Symbian browser:** As the names already suggest, the BlackBerry browser and the Symbian browser are only available on BlackBerry OS respectively Symbian OS.

Although these web browsers might differ in their user interface, privacy features, browsing comfort or configuration possibilities, some of them are based on the same JavaScript Engine or the same rendering engine. **Figure 6** illustrates this behavior. While the Firefox family (Mozilla Foundation, 2011c), the Internet Explorer family (Microsoft Corporation, 2011e) and the Opera family (Opera Software ASA, 2011d; Lindström, 2009) rely on their individual rendering and JavaScript engines all other major browsers utilize WebKit as their rendering engine (Apple Inc., 2011a; Research In Motion Limited, 2011a; Carson, 2007) and except for Google also as their JavaScript engine. Only in Google’s Chrome and Google’s Android browser the JavaScript code execution is done by the external V8 Engine (Google Inc., 2011g; Google Inc., 2011b) and not by WebKit’s built in SquirrelFish.
It is important to note that the actual standard compliance of a web browser is not so much defined by the web browser itself but by the rendering and JavaScript engine it uses. If many browsers share the same engine they often support a similar set of web technologies and features. This has to be kept in mind especially when comparing market share and availability of different browsers.

3.5. Future Web Standard Candidates

To understand and evaluate the maturity and capabilities of the web as a platform, it is important to introduce all future standards that are currently under development. Therefore, this chapter provides an overview on all major new standards and specifications related to web technologies and designed to evolve standard web sites to fully featured web applications. This chapter focuses only on new technologies and only includes already finalized and widely deployed standards when they are required for better understanding. In addition, it does only focus on relevant standards and technologies.

3.5.1. Presentation tier

Providing a rich and powerful Graphical User Interface (GUI) is a very important aspect when trying to break even with native applications or web applications that required plugins. Since the introduction of personal computers the user interaction has emerged from simple command line access to fully featured graphical interfaces, which provide a broad set of interactive features. Only when web applications are able to provide a similar graphical user expe-
3. Web Applications

Experience as native applications, they have the potential to coexist or supersede them. The following standards strive to bring such a graphical user experience to web applications:

Graphics: With the introduction of the HTML5 canvas tag (Hickson, HTML5, 2011, Section 4.8.11), web applications are now capable of performing customized 2-dimensional drawing operations directly on a resolution-dependent bitmap canvas. In contrast to having to pre-render graphs, game graphics or visual images on the server with the sophisticated JavaScript Canvas 2D Context API (Hickson, HTML Canvas 2D Context, 2011), specified as Working Draft at the W3C, it is now possible to draw lines, shapes and text directly on the client. Furthermore, the Canvas API provides mechanism to load, manipulate and export bitmaps images right within the web browser. For the first time is it possible to perform sophisticated 2-dimensional image processing, graphical data reporting and sophisticated game graphics solely on the client.

In addition to the possibility of utilizing custom 2-dimensional graphics through JavaScript, the Khronos Group is currently standardizing a specification which allows hardware accelerated 3-dimensional graphics using a subset of OpenGL directly through JavaScript. This new standard is entitled WebGL (Marrin, 2011) and likewise uses the HTML5 Canvas element as a base for rendering 3-dimensional scenes and models. This enables developers to bring state of the art computer games and other 3-dimensional visualizations directly and plugin-free to web applications.

Audio and Video: With the success of the web video platform YouTube, using the web to watch videos and listen to music has become a standard use case on the Internet. Unfortunately, many web sites require plugins to be installed on the system. To allow plugin free playback of different media types directly within the browser, HTML5 defines a new audio and a new video tag, which allow embedding media files directly into web pages (Hickson, HTML5, 2011, Sections 4.8.6 to 4.8.10). A dedicated JavaScript API can be used to control the media playback and customize the player ap-
3. Web Applications

appearance. Due to political discussions the HTML5 standard does not yet define a set of audio and video codecs that have to be supported by the browsers. Therefore, different browser vendors support different audio and video types.

**Advanced Layout Managers:** One of the huge disadvantages of web technologies was always the fact that creating an adequate user interface layout was nearly impossible. From the beginning of HTML it was intended to layout documents and not user interfaces. This was one of the reasons why many web developers chose HTML table tags to position their GUI elements. As HTML tables were never meant to be misused for layout purposes, the CSS box model was introduced. Unfortunately, the CSS box model still left several things to be desired. With the *Flexible Box Layout Module* (Atkins Jr., Mogilevsky, & Baron, 2011) and the *Grid Positioning Module* (Mogilevsky & Mielke, 2007) these shortcomings are about to be addressed. While the Flexible Box Model allows the browser to dynamically size and position elements according to special pre-defined rules, the Grid Positioning Module brings back all the benefits from the early HTML table layouts but in a clear structure without misusing any differently intended features.

**Device-dependent Layout:** As already outlined in chapter 5 there is a huge variety of different screen resolutions and sizes, which require individual adoption of the user interface. Although layout managers as listed above provide mechanisms to compensate for different resolutions, there might still be a need to specify different layouts for different devices. To address this the W3C has created *Media Queries* (Lie, Çelik, Glazman, & van Kesteren, 2010), a CSS extension, which enables web applications to include different styles depending on the properties of the device. Dedicated CSS rules can be assigned for different screen sizes, screen resolutions, dive orientations or available screen colors. Media Queries have already reached the Candidate Recommendation stage.
User Interaction: A further area in which web applications still have room for improvement is intuitive user interaction. To enable advanced user interfaces the HTML5 specification introduces a flexible Drag and Drop API (Hickson, HTML5, 2011, Section 7.7), which consist of a set of new attributes to specify draggable objects and drop zones and JavaScript interface to react on drag and drop events.

Together with drag and drop, another API for advanced GUI design is the Clipboard API (Steen, 2011). Similar to native applications or plugin-based web applications, this API specifies how web applications can utilize the clipboard to enable copy and paste of custom application data. This API is still in a very early Working Draft stage and the specification leaves a lot of open questions.

Another API that tries to maximize the flexibility for user interaction is the Touch Events Specification (Schepers, Moon, & Brubeck, 2011) currently maintained as Working Draft at the W3C. As more and more devices can be controlled using a touch screen, it is important to directly integrate such functionality into modern web applications. Similar to the drag and drop API the Touch Events specification introduces a set of events and listeners for JavaScript to react and control single- and multi-touch interaction.

Although forms and input fields have been available in html for quite a while, there was no input type that allowed rich text formatting. Only plain text could be entered. To overcome this limitation Microsoft introduced the contentEditable attribute and a corresponding JavaScript API to bring rich text editing capabilities to its web browser. Inspired by this idea, Mozilla invented a similar concept for its web browser lineup. Today many modern web browsers implement their own slightly different version of Microsoft’s idea (Pilgrim, The Road to HTML 5: contentEditable, 2009). To streamline these efforts and to come to with unified API the W3C has now included the contentEditable specification into HTML5 (Hickson, HTML5, 2011, Section 7.8).
One of the most confusing problems when dealing with web applications is the usage of the browsers back and forward buttons. In the beginning, those buttons were simply for navigational purposes. Today many users misinterpret them as undo buttons. In addition web pages can dynamically download additional content in the background without loading another page. Therefore, in many cases the back button does not always have the effect the user expects. To solve this, HTML5 introduces an advanced History object, which allows the web application to manually add additional steps to the browsing history (Hickson, HTML5, 2011, Section 5.4.2). This way an application can map its internal state directly to the back and forward buttons of the browser and get notified if the user uses them.

Notifications: Web Notifications (Gregg, 2011) provide web applications with the possibility to show small popup windows on the users desktop or the device’s notification system. Those notifications, e.g. when receiving a new email or chat message, can be displayed even when the browser runs in the background or is minimized. Clicking on the notification will directly take the user back to the corresponding web application.

Media Capturing: Today many mobile devices are equipped with cameras and microphones, making it possible to conveniently capture audio and video or to take photos. To bring such capture capabilities to web applications the W3C has initiated two new specifications, currently both Working Drafts, which enable direct interaction with microphones and cameras through JavaScript. While The Media Capture API (Tran, Oksanen, & Kliche, 2010) aims to define a comprehensive API to help developers interacting with capture devices directly, the HTML Media Capture API (Oksanen & Hazaël-Massieux, 2011) extends the HTML file input dialog to enable quick capturing of new media files for direct upload.

Speech Input: The Speech Input API Specification (Sampath & Bringert, 2010) is an Editor’s Draft proposed by Google Inc. Although the documentation is hosted by the W3C it
3. Web Applications

has not yet reached the state of a technical report. Nevertheless, a dedicated work
group, the HTML Speech Incubator Group, has been formed and will continue to
work on Google’s draft and transfer it into a technical report. The Speech Input API
itself is designed to provide access to speech recognition facilities that are either
hosted on a remote server or embedded directly into the web browser. Developers
can utilize this API to either continuously monitor the user’s spoken requests or to
define certain input fields which can be filled through speech input.

Animations: Using Animations in web applications is not an entirely new concept. Currently,
there are many web sites that animate objects using JavaScript requiring complex
source code and a lot of computational power. To solve this, the new CSS specification
contains a Transitions Module (Jackson, Hyatt, Marrin, & Baron, CSS Transitions
Module Level 3, 2009) and an Animation Module (Jackson, Hyatt, & Marrin, CSS Ani-
mations Module Level 3, 2009) to support hardware accelerated and smooth anima-
tions, which can be easily integrated into a web site. Together with the 2D and the 3D
Transforms Module (Jackson, Hyatt, & Marrin, CSS 2D Transforms Module Level 3,
2009) (Jackson, Hyatt, & Marrin, CSS 3D Transforms Module Level 3, 2009) there is
now a convenient way to add many different kinds of animations to web applica-
tions.

Advanced Forms: With HTML5 and the new CSS standards, web developers get a much more
convenient way of designing forms. HTML5 does not only add many new form ele-
ments including date, time or color input fields, but also allows setting default values,
placeholders and data formats (Hickson, HTML5, 2011, Section 4.10). Especially for
web applications running on smaller screens these pre-defined data formats come in
handy because they allow the browser to select the best suited on-screen keyboard.
If the data format is set to numeric values the device can display a numeric keyboard
layout, if the data format is set to text the device can display a regular keyboard lay-
out. Whenever the user enters invalid or insufficient values, CSS can be used to au-
3. Web Applications

tomatically style the improper filled fields and give the user direct feedback (Çelik, 2004).

**New HTML Elements:** In addition to the markup elements already mentioned, HTML5 introduces many more new tags and attributes. In contrast to the audio, the video or the canvas tag, most of these additional elements do not have major features associated with them. They are merely supplementary tags that were designed to provide more semantic information throughout the document. Chapter 4 of the HTML5 specification (Hickson, HTML5, 2011) provides a complete list of all available elements and attributes.

**Advanced Styling:** Similar to the animations mentioned above, the new CSS specifications outlines many more features to simplify the styling of web applications. Even though styling features like drop shadows, rounded borders (Bos, Etemad, & Kemper, 2011), gradients (Etemad & Atkins Jr., 2011), or custom fonts (Daggett, 2011) were achievable by using pre-built images, it is now possible to add all these styles with simple CSS code. For example, instead of adding special images for every rounded border of a box this can all be done with one single line of CSS.

Beyond these already mentioned improvements, the new CSS specifications cover a lot more enhancements. It is out of scope of this thesis to cover every single new CSS feature. Therefore only relevant changes to CSS are described.

**Other Markup Languages:** As already mentioned in chapter 3.3, MathML and SVG are two independent XML-based standards that can both be included into HTML documents to enhance the visual appearance of web applications:

SVG (Dahlström, et al., 2011) is vector-based image format, which allows graphics that can be resized without quality loss along with a small file size. Similar to CSS animations and transitions, SVG files can also contain pre-defined animations.
MathML (Carlisle, Ion, & Miner, 2010) provides a comprehensive list of tags and attributes that can be used to conveniently layout complex mathematical terms and equations. Thanks to MathML, mathematical formulas can be processed and displayed directly by the web browser.

### 3.5.2. Logic tier

To develop fully featured web applications it is important that web browsers provide rich and flexible APIs to support a variety different use cases. The following sections will introduce these new methods to bring the capabilities of web applications to a whole new level.

**Multi-threaded applications:** While the principle of performing several computational tasks at parallel using threads has been available for a long time on many platforms and systems, web applications were not capable to do so. Although many JavaScript APIs where designed based on callback mechanisms to keep the application responsive until the result of an API call was calculated, regular JavaScript code could not be executed in parallel. With the newly introduced *Web Workers* (Hickson, Web Workers, 2011), a Working Draft maintained by the W3C, such parallel computing comes to web applications. Thanks to Web Workers both API calls and regular JavaScript code can be executed without locking the browser and without requiring complex callback mechanisms.

**Geographic location:** Especially on mobile platforms the importance and spread of so-called location-based services has significantly increased. Thanks to built-in Global Positioning System (GPS) chips and mechanisms to calculate the users location based on surrounding cell phone towers, nearby wireless networks or the device’s Internet Protocol (IP) address, it is possible to provide the users with distinct services and specially tailored information based on their location. With the *Geolocation API Specification*
3. Web Applications

(Popescu, 2010), which has already reached the state of a Candidate Recommendation, web applications can now also provide location-based-services.

**Enhanced Communication:** One of first successes in the chapter of web applications was the introduction of the XMLHttpRequest (van Kesteren, XMLHttpRequest, 2010), which was the foundation for what later became known as AJAX web applications. For the first time it was possible from within a web application to exchange data between client and server without needing to reload the page or to restart the application. Unfortunately the XMLHttpRequest specification was only capable of transmitting a very limited number of data types. To address these shortcomings the W3C is currently working on a revision of the XMLHttpRequest entitled XMLHttpRequest Level 2 (van Kesteren, XMLHttpRequest Level 2, 2010). Along with a much broader field of different file types and formats, including binary data, this new iteration of the specification also adds more convenient upload mechanisms for form data and binary files and allows the web application continuous monitoring of the up- and download process through Progress Events (van Kesteren, Progress Events, 2011). While the XMLHttpRequest has reached the status candidate recommendation, Level 2 of the specification is still in the Working Draft stage.

The disadvantages with XMLHttpRequest Level 1 and 2 is the fact it strictly follows the request and response scheme by which the web browser sends out a request, which the server answers with a single response. A server can never send multiple responses in return for a single request nor can the server send a request itself. To overcome these limitations the W3C has provides a more sophisticated communication API called WebSocket API (Hickson, The WebSocket API, 2011). WebSockets, similar to Berkeley sockets, provide an end-to-end bi-directional full duplex communication channel for binary data exchange. With WebSockets it is simple and much more convenient to develop two way real-time communication in JavaScript because it
3. Web Applications

does not rely on continuous polling, which XMLHttpRequest-based web applications did. The WebSockets API is currently a Working Draft.

Although WebSockets and XMLHttpRequests provide a broad variety of data exchange between client and server there is still a scenario left uncovered. In both cases the communication has to be initialized by the client. A server can never send out data before the client has triggered the setup of the communication channel. A solution to this problem is introduced with the W3C’s Server-Sent-Events (Hickson, Server-Sent Events, 2011), which allows push notifications to the client. There does not have to be a manual communication channel. The browser handles all message exchanges and the web application developer only has to register an event handler to receive the pushed data. There is no polling required, which saves bandwidth and especially on mobile devices increases battery life. The specification is currently a working draft.

A fourth communication API, which serves a slightly different purpose than the above three, is the HTML5 Web Messaging (Hickson, HTML5 Web Messaging, 2011). Instead of providing various kinds of client-server communication this API provides communications channels between different web applications running on the same client. This API is especially intended for web applications willing to exchange data but hosted on different domains, because web browsers forbid cross-site interaction for security and privacy reasons. HTML5 Web Messaging was designed with these restrictions in mind and therefore provides a secure and safe channel to exchange data between applications from different origins. Although this specification has HTML5 in its name it is a completely independent standard and is at the moment a working draft.

**Extended Messaging:** The W3C Messaging API (Hazaël-Massieux, Chitturi, Froumentin, Oteo, & Widell, 2011), not to be confused with HTML5 Web Messaging, is a generic programming interface allowing web applications to send various type of messages
3. Web Applications

including e-mail, SMS or MMS. Similar to the already widely used URL modifier mail-to, this API allows sending pre-composed messages either directly or through the platforms corresponding built in messaging application.

**Calendar and Address Book:** With the *Contacts API* (Tibbett, Contacts API, 2011) and the *Calendar API* (Tibbett & Chitturi, Calendar API, 2011) the W3C is currently working on two specifications to enable the information exchange with other contacts and calendar applications or services such as the local address book or the device’s local calendar. Both APIs define mechanisms to read, filter, update and create contact data or calendar dates. The Contacts API as well as the Calendar API are currently still in an early stage of the standardization process and are only available as working draft.

**System and Device Information:** It is also important to enable web applications to retrieve detailed information about the hardware and software they run on. To satisfy this, the Device APIs and Policy Working Group has published a First Public Working Draft entitled *The System Information API* (Tran & Froumentin, 2010). This API provides information about the power supply, the battery status and load and temperature of the Central Processing Unit (CPU). It also covers any attached hardware sensors including ambient light, proximity, ambient temperature, ambient noise or atmospheric pressure. Furthermore, it offers information about any output and input device such as the display, the audio output, the keyboard, the pointing device, the camera and the microphone. Beyond this there are also API calls to check the existing storage units, network properties and available audio and video codecs.

In addition to the System Information API the W3C also defines the *Network Information API* (Chitturi & Berjon, 2011) and the *Battery Status Event Specification* (Kostiainen, 2011). On top of providing the information already mentioned in the section above, these two specifications also provide mechanisms to receive continuous
3. Web Applications

events whenever the status of the network connection or the battery have changed. Both API are also published as Working Draft.

3.5.3. Data tier

As already outlined in chapter 3.1 the data tier provides mechanisms to store and receive data persistently. In the scope of web browsers this means that even when the web browser is closed data persisted by the web application remains stored and can be accessed as soon as the web application or website is launched again. Currently, there are four different kinds of local persistent storage:

**Local File System:** With the *File API* (Ranganathan & Sicking, 2010), the *File API: Writer* (Uhrhane, File API: Writer, 2011) and the *File API: Directories and System* (Uhrhane, File API: Directories and System, 2011) the W3C maintains three Working Drafts, which allows accessing the local files system through JavaScript. The APIs provide mechanisms to read and write files bitwise from and to a dedicated directory structure managed by the web browser. Additionally, the user can explicitly select files from outside the browser’s managed directory to be read and processed by a web application. Possible use cases for the File APIs include preview or manipulation of files before an upload to the web server or storing documents, settings and resources persistently on the local hard drive.

**Local Database:** In contrast to the File API, which is designed for larger binary objects and files, the W3C has also specified a storage facility for structured data referred to as *Indexed Database API* (IndexedDB). This API provides mechanisms to store and retrieve JavaScript objects to and from a database layer provided by the browser. The IndexedDB specification (Mehta, Sicking, Graff, Popescu, & Orlow, 2011) is currently a Working Draft.
3. Web Applications

In addition to the IndexedDB API the W3C also hosts a technical report entitled *Web SQL Database* (Hickson, Web SQL Database, 2010). This specification describes a JavaScript API designed query a local database hosted by the browser directly with SQL statements. Currently, this specification is no longer actively maintained because all implementers have used the same SQL backend, which is not suitable for a standard in development. Nevertheless, Web SQL has been implemented by three of the five major web browser vendors and is therefore actively used by web developers.

**Key-Value-Storage:** The simplest way to store data persistently in the browser is provided by the *Web Storage API* (Hickson, Web Storage, 2011). Similar to the well-known Hyper-text Transfer Protocol (HTTP) session cookies this API allows maintaining key-value-pairs. Both pair and value have to be stored as a string and only provide one-to-one relationships. The Web Storage API, currently a Working Draft, defines two different storage types: *Local Storage* and *Session Storage*. Session Storage provides the storage mechanisms for each browser window or tab individually and only as long as the window is opened. Local Storage on the other hand allows accessing the storage from any window or tab from the same domain and persists the values even if the web browser is closed.

**Offline Application Cache:** The *Offline Application Cache* is one of the few APIs defined directly in the HTML5 specification (Hickson, HTML5, 2011). Its purpose is to make the entire web applications available for usage even if there is no connection to the server. To accomplish this a so-called *manifest* file has to be referenced from any HTML file of the web application. This manifest is a text file containing a list of all resources including HTML, CSS, JavaScript and image files required by the application to run. As soon as a web browser detects such a manifest it downloads and caches all listed files on the local hard drive. If the user tries to access a website or web application and the browser cannot reach the server the cached files loaded instead and the application remains operational. In contrast to the regular browser cache the offline applica-
3. Web Applications

Application does not expire automatically and is controlled by the web application and not by the browser. When the connection to the server can be reestablished the web browser automatically checks updates the locally cached files if there were any changes to the manifest since the last access.

3.5.4. **Featured Technologies**

As discussed above there are currently many new candidates for future web standards in the standardization process. *Table 5* summarizes the already outlined specifications:

| Presentation tier | Canvas 2D Context API, WebGL, HTML5 Audio & Video, CSS Flexible Box Layout Module, CSS Grid Positioning Module, CSS Media Queries, HTML5 Drag and Drop API, Clipboard API, Touch Event Specification, HTML5 Editable Content, HTML5 Browsing History, Web Notifications, Media Capture API, Speech Input API Specification, CSS Transition Module, CSS Animation Module, HTML5 Forms, HTML5 Markup Elements, CSS Backgrounds and Border Module, CSS Image Values and Replaced Content Module, CSS Fonts, SVG, MathML |
| Logic tier | Web Workers, Geolocation API Specification, XMLHttpRequest, Progress Events, Web Socket API, Server-Sent-Events, HTML5 Web Messaging, Messaging API, Contacts API, Calendar API, System Information API, Network Information API, Battery Status Event Specification |
| Data tier | File API, File API: Writer, File API: Directories and System, Indexed Database API, Web SQL Database, Web Storage API, HTML5 Offline Application Cache |

*Table 5*: Candidates for future standards and specifications
3. Web Applications

3.6. Prefixing

As outlined in 3.2.1 the W3C requires any standardization proposal to be implemented by several parties before it can emerge into a final standard. However, depending on the maturity level of the specification these proposals are still subject to fundamental changes. Additionally, as shown in Figure 5, standards might be initialized or suggested by other parties such as browser vendors who have already implemented their proposal. Every time a standard proposal changes, the behavior of the web browser might have to change. In the long run a different web browser version might require different syntax or markup for the same feature. To avoid that a later standardized functionality works different in different browser version, although referred to with the same name, the main browser vendors have started to prefix features that are not yet stable. This is usually done be prepending a short letter combination to the standardized name. Once the standard proposal is solid enough the prefix can be dropped.

```css
-moz-transform: rotate(10deg); /* Firefox 3.5+ */
-o-transform: rotate(10deg); /* Opera 10.5+ */
-webkit-transform: rotate(10deg); /* Chrome & Safari3.1+ */
-ms-transform: rotate(10deg); /* Internet Explorer 9 */
```

**Listing 1**: Prefixing of unstable features in modern web browsers

**Listing 1** shows a prefixing example for the CSS function *rotate()* As soon as a function or markup attribute starts with a leading hyphen all other web browsers ignore it. As the example shows, it has to be kept in mind that developing a web application that uses new unstable features might require significant overhead to bring it to all browsers due to the prefixing.
3. Web Applications

3.7. Compatibility Tests and Ratings

Although every web browser provides basic functionality to interpret and display web pages and execute JavaScript code, not all of the new features listed in chapter 3.5 are available in every one of them. Depending on the release date of the browser or political decisions there might always be web browsers that do not provide designated APIs or support certain markup and styling. To provide users as well as web developers with a reliable indicator on which web features are available in which web browser, different organizations or private persons have created compatibility tests and ratings to measure maturity and performance of web browsers. The following tests and listings provide a good overview on web browser standard compliance.

**When can I use:** The website caniuse.com provides a detailed “compatibility tables for support of HTML5, CSS3, SVG and more in desktop and mobile browsers” (Deveria, 2011). This site is especially handy when trying to compare different web browser or browser versions directly. On top of that all compatibility ratings are automatically applied against the individual browser’s market share. This gives a rough estimate on how widely deployed each individual feature is. The listed browsers include previous, current and future releases of the most important vendors.

**HTML5 Test:** Similar to the website When can I use, the HTML5 calculates an individual store for the visitor’s current browser. The store itself “is an indication of how well your browser supports the upcoming HTML5 standard and related specifications” (Leenheer, 2011). It includes HTML markup and JavaScript APIs but does not list support for CSS features.

When working with these tests one thing must always be kept in mind: Although two browsers may reach the same score or percentage that does not mean that they both support exactly the same features. In fact two browsers with the same score could in theory not have a single feature in common. However, some standards and specifications are more sta-
ble then others, which effects that some standards are more likely to be implemented then others.

Of course there are many more compliance tests and listings around. However, for this thesis and the covered topics these two tests provide the most comprehensive results.

3.8. Summary

Hosting all application logic and data on a server and providing only a user interface on the client is considered thin client architecture. This is the most commonly used architectural principle for web applications. However, sometimes it might still be useful to store parts of the data or the application logic on the client. This is especially useful when for some reason an Internet connection is not available. As soon as application logic and application data are stored on the client, this is referred to as fat client. Thanks to the upcoming web standards future web applications can conveniently be developed as thin as well as fat clients.

When talking about web applications there is a common misunderstanding about the term HTML5. In its original definition, the term only defines the W3C HTML specification, which is focused on describing markup elements and their related JavaScript API. In the public reception, HTML5 has emerged as an umbrella term for any further web standard or web technology. Reading articles about HTML5 has therefore always to be done with a grain of salt.

With the upcoming standards and specifications web developers can soon use a broad set of new features that will allow developing sophisticated applications. The upcoming standards include advanced and simpler styling and layout of GUIs, plugin-free multimedia support, sophisticated 2D and 3D graphics, advanced access to the device’s hardware and data, enhanced client-server communication methods and much more. Additionally, concepts like SVG, Media Queries and CSS Layout Managers make it very convenient to develop web applications for different device types with different screen resolutions.
3. Web Applications

This huge platform interoperability is based on the high standardization of these technologies, which allows a single web application to run in many different browsers. However, not all standards are yet supported by any browser, as the standardization process of many upcoming specifications is still ongoing. This causes frequent changes of the standardization candidates and has always to be considered when developing web applications that use the latest features.

Fortunately, there are some solutions to the problem of unfinished standards and insufficient web browser support. Web sites with compatibility tests provide a detailed overview on which browser supports which individual web standard. Additionally, web browser vendors can implement even incomplete standards by applying a vendor specific prefix to every function and feature. Although this allows using new web standards from an early stage on, developers might be required to frequently update their applications and add special code for each individual browser.

When comparing different web browser it must also be kept in mind that although web browsers are developed by completely different vendors they might still share the same rendering or JavaScript engine. This means that two different browsers might still have a very similar feature set. Especially on smartphone and tablets the rendering and JavaScript engine WebKit is widely used, which leads to a very homogenous development of the API and standards support within all these browsers.
4. Feasibility Analysis

To gain some experience of the advantages and risks of the new web technologies and specifications, a prototype web application was implemented. This chapter provides a description of the desired functions of the prototype and how they were implemented using the new web standard proposals.

4.1. Merck Toolbox

The Merck Toolbox is a web-based application developed with the Adobe Flash platform, which provides a centralized information platform for Merck’s field representatives. This information about the company, its products and competitor’s positioning is available in the form of documents, presentations and videos. Field representatives can conveniently search or browse through them by category and group their favorite files in folders using drag and drop interaction.

All existing videos are delivered as Flash Video files. Documents and presentations are stored in the FlashPaper format. The FlashPaper format, similar to PDF, is a lightweight method to supply different document types in a simple, slick and easy to display way. It also allows documents to be controlled by Digital Rights Management (DRM) and printed directly from within the application.

The Merck Toolbox is a client-server based application. While the user interface and application logic are downloaded to the client all documents and required Meta information, such as document category or keywords, remain on the remote server. To search or browse through the document repository the application sends a request using a web service. The server then returns a list of document names that match the desired criteria. Once the user tries to open a document the application downloads the corresponding file from the repository through a web server. Figure 7 provides a simplified overview on this procedure.
One of the main ideas behind the toolbox is to provide a **single point of truth**. This means that field representatives will find all the information they need using only one single application. It is not possible to download any data or files from the document repository to the local hard drive. This way it is ensured that all users are always working with the most recent up-to-date copy of a document or video directly from the server. Unfortunately, this also means that the current Merck Toolbox implementation does always require a connection to the server to display any document.

To get a better understanding on the functionality of the Merck Toolbox, screenshots of the applications are provided in the appendix.

### 4.2. Implementing the Prototype

The main focus when developing the prototype was to re-develop the Merck Toolbox using only open web technologies. The look and feel of the original application was reused wherever possible to show the new possibilities and flexibility of the new web standards.
4. Feasibility Analysis

In addition to covering all the old functionality, the prototype implementation has been extended with a so-called offline mode. With this additional feature the users can download dedicated files to their hard drive and view them even if they have no connection to the server. To not break the idea of providing a single point of truth and to prevent field representatives to work with old versions of the documents, the offline persisted files can only be accessed from within the web application and are automatically updated as soon as a connection to the server is available again.

In contrast to the prototype the general client-server architecture of the application is different. This was mainly because the old web service and the old document repository were designed to deliver FlashPaper documents and Flash Video files, which could not be re-used. Additionally, setting up a new repository and a new web service on Merck’s IT infrastructure was seen as too complex and time-consuming for a proof-of-concept and therefore out of scope for this thesis. The new architecture used for the prototype is composed as follows:

A standard web server is responsible for handling the communication between the server and the client. Additionally, the server also holds parts of the application logic such as the user authentication and the management and aggregation of the repository contents. The server-side application logic was written using the PHP: Hypertext Preprocessor (PHP) simply because it was already well known to the author of this thesis. In general the programming language of the server logic is not so much of interest. With the Common Gateway Interface (CGI), many different languages can be used without any influence on the client side development. The client software including the user interface and all the application logic required for the offline mode were developed with HTML, CSS and JavaScript. Only the user authentication implemented in the prototype re-uses the existing old web service to validate usernames and passwords. **Figure 8** provides a structured overview on the used architecture.
4. Feasibility Analysis

As the FlashPaper and the Flash Video files require the proprietary Adobe Flash platform to be displayed it was required to find a suitable alternative. Due to the fact that Adobe PDF document has gained a much broader acceptance and support it was chosen to provide the documents and presentations in the form of PDF files. To play the video files directly within the browser using the HTML5 video tag (compare chapter 3.5.1) it was required to re-encode the video files to a format that all browsers support. Unfortunately, as already mentioned, not all browsers have agreed on a common format. Therefore the videos were converted to both MPEG-4 (Microsoft and Apple) and WebM (WebKit, Opera and Mozilla) video files. When displaying a video, that application automatically detects and loads the proper video format.

The development of the client required to use many of the new web technologies outlined in chapter 3.5. The following paragraphs will give a short overview of how these technolo-
4. Feasibility Analysis

gies can be used to develop comprehensive and sophisticated applications directly in the web browser:

The graphical user interface uses many of the new styling features introduced with CSS. First of all, the flexible box layout module was used to position the individual components on the screen. A huge advantage in contrast to the conventional box modules is that contents can be configured to automatically divide available space among them. As soon as a new element is placed into a flexible box container, all other elements shrink in size. This for example was used to automatically align the category navigation and to layout the favorite sections. Another benefit of the flexible box layout module is it ability to center content vertically which was not conveniently possible before. In addition new features like drop shadows, rounded borders and gradient were used to provide a similar experience as the original Toolbox. Finally, the new CSS font module was used to make the Merck default font available for rendering even if not installed on the particular system.

To provide a similar convenient and intuitive user experience as seen on the Flash version of the Toolbox the HTML5 drag and drop API was also used. This way documents and videos can be added to a favorite section or put into the download cart. The original Merck Toolbox also provides an animated clock on the welcome screen, which shows the current time and date. In the prototype this animated clock has been implemented using the HTML5 canvas element. The clock hands are animated using JavaScript. Although this clock is not crucial for the functionality of the application it provides a good example on how HTML and JavaScript can now be used to provide complex graphical GUI elements.

Instead of using many different HTML files, as was common in the past, to provide the user interface only one single HTML document is loaded when the application is started. This start page with its related JavaScript files is then responsible for downloading, managing and displaying all additional content. Instead of browsing through a set of individual HTML files, this way the users get the feeling of a single consolidated application. This concept of down-
4. Feasibility Analysis

loading additional content asynchronously in the background instead of navigating through distinct files is not new since this has been able to be done with XMLHttpRequests (XHR) for quite a while now, but over all this is a large step from single web pages towards a compact web application.

While with the above mentioned new web technologies it was possible to re-implement the Merck Toolbox the newly added offline functionality required to use a even broader set of new specifications. As already stated for the offline functionality of the application, selected files can be downloaded, stored and later viewed offline. In the prototype implementation the download of the files is done using the XHR level 2. XHR level 1 was not suitable for this, because it has no support for binary files, which the videos and documents used in the application are. Additionally, XHR level 2 allows to continuously monitor the download’s progress by firing continuous progress events. This download progress is displayed in the GUI using the new HTML5 progress element.

As soon as the file is downloaded the File APIs are used to store the binary files on the local hard drive totally hidden from the user. To allow the application to conveniently process the local file during the offline mode, it is required to store some additional Meta data on the client side such as the file’s language, its version, its resolution and its mime-type. This is done using the IndexedDB API, which stores a record related to each individual file persisted on the hard drive. Because downloading and writing the file to the hard drive may take some time, the complete process uses multiple threads by utilizing the WebWorker specification. This way the user interface stays responsive even if several downloads are in progress. Once the download is completed the application displays a notification pop up window. These pop-ups are created with the new Web Notifications. This allows notifications to be visible even if the web browser is running in the background or is minimized.

The offline availability of the application is implemented using the HTML5 Offline Web applications. As soon as the web page is loaded for the first time, the browser automatically
downloads all specified resource files that are required for the application. From now on the user can enter the URL of the application at any time. If a connection to the server is available, the full functionality of the application and all hosted documents are available. If no connection can be established a minimized version of the prototype is loaded, listing only the available files. The user can only access the application by typing the URL into the web browser. The browser then either displays the online or the offline view. As soon as the online view is loaded the application automatically checks for updated versions of the local files and replaces the local copies with the new files. This update progress has been implemented manually with JavaScript and the XHR API. If the web application itself (HTML, CSS, JavaScript and other resources) is changed thanks to the Offline Web applications specification the applications updates itself automatically.

A set of screenshots illustrating the functionality and look-and-feel of the HTML Toolbox prototype can be found in the appendix of this document.

4.3. Open Issues

Although the described prototype is fully operational, there are still some open issues that could or should be addressed or improved:

Local Files: When saving files on the local hard drive this can quickly become a security risk. Even though using the File API allows storing the files encapsulated in the web browser’s sandbox there are still possibilities to retrieve the files even without using the web applications. To solve this problem, an attempt was made to encrypt the files directly on the server with the individual user’s name and password as the encryption key and decrypt them on the client right before they are displayed. Without the valid username and password the files would be useless. While the decryption on the server side was easily accomplished, the decryption on the client was rather complex. On the server, PHP allows to conveniently access fast and efficient crypto-
4. Feasibility Analysis

graphic algorithms. This way the documents and videos could be encrypted in only a few seconds. On the client however JavaScript does not provide any decryption routines. Therefore the decryption algorithms had to be completely written in JavaScript. Although in the end it was possible to successfully encrypt a file on the server and decrypt it on the client, it took over five minutes on a reasonable fast computer to decrypt a 30 megabyte video file. The idea was therefore no longer pursued during the prototype development.

Another idea that was just not implemented due to time constraints would be to assign an expiration date to locally stored files. If either a file is on the hard drive for a specific amount of time or the application was unable to check for updates in a longer time period. The prototype application could automatically restrict access to the local files. This would ensure that the field representative has to connect to the server from time to time.

Document Format: With the implementation of the prototype is has been verified that the Merck Toolbox could be implemented using new web technologies. However, PDF files are not considered to be part of any web standard and require therefore an additional plugin or viewer application. Fortunately due to the wide distribution of PDF almost all platforms are capable of displaying such documents. Nevertheless, the Merck Toolbox could be improved by using a different format for documents and presentations.

4.4. Platform Independence

When trying to run a web application in a different browser one of the following three cases might appear: The applications runs without any or just minor adjustments, some features have to be modified or in some cases the application cannot be run at all.
This behavior can be easily demonstrated when looking at the Merck Toolbox prototype. Due to the fact that Google Chrome offered the most advanced support for new web specifications and technologies the prototype was first developed to be fully functional with that browser. This included the support for an offline mode, which required a comprehensive availability for local data storage. Once this was successful, it was attempted to run the prototype on the iPad. Although the application was almost usable on the tablet device there where still some barriers to overcome.

One example is the absence of drag and drop support on many tablet browser such as the Apple iPad or the Motorola Xoom. To still provide the possibility to drag and drop files to the user’s favorite sections instead of the HTML5 drag and drop API the W3C touch events specifications was used to recreate the touch experience. Even in cases where some features or APIs are not supported by the browser it is possible to find workarounds. Similar is true for the flexible box layout module. While this provides great convenience for GUI designers it still could be implemented using JavaScript. Of course this would complicate development but it extends the availability of the application to more platforms.

Unfortunately, there are also cases where some APIs are crucial for the feasibility of an application and the project could not be realized without them. The current Internet Explorer for Windows Phone for example does not provide support for plugin free HTML video playback. It is therefore not possible to bring the Merck Toolbox prototype to the windows phone platform as a web application. Concerning the offline mode of the Toolbox prototype there currently is no full support for the File APIs except for Google Chrome, which means that the offline functionality could only be implemented there.

To summarize this it can be said that there is always an optimal set of APIs that should be available and a minimal set that must be available. This is illustrated in Figure 9 and Figure 10. The orange line indicates the features required for the offline version. The blue area marks the technologies required for the online mode. The closer a browser comes to the
4. Feasibility Analysis

upper end (best) of the blue box, the more simple and convenient it is to support this browser. The closer a browser scores to the lower end (minimal) the harder it is and the more special browser dependent adjustments have to be made.

Two things have to be kept in mind when working with the following two images. First of all they only provide an approximation. As already outlined in chapter 3.7 the browsers with similar percentage values do not automatically provide the same features. The figures are therefore only a rough indicator. Additionally, upcoming releases of the browser will support more specifications then the current one. This means that in the future more and more browsers will be capable of running the application without any limitations. For the prototype implementation of the Toolbox not all adjustments required for each browser have been made. It is therefore only available on a subset of these browsers.

The browser compatibility data used in the following figures is based on the results of the browser compliance test hosted on html5test.com (see chapter 3.7).

![Figure 9: Feasibility rating of new web technologies support on notebooks](image)
4. Feasibility Analysis

![Feasibility Analysis](image)

**Figure 10:** Feasibility rating of new web technology support on smartphones and tablets

4.5. **Web Applications on iOS**

One other interesting aspect when developing web applications is the way web applications can be used on iOS devices (Apple Inc., 2010). To provide a more native feeling of web applications Apple has equipped its iOS platform with a method to seamlessly integrate web applications into the system. This can be done by adding Meta tags to the application’s HTML pages. These Meta tags allow to specify an application icon and to hide the browser controls such as the address bar. The result is that the web application can use the entire screen and it is invisible to the users that the application they are running is actually a web application instead of a native one. **Listing 2** shows the additional configuration used in the prototype to be run as an embedded web application.
4. Feasibility Analysis

Although this whole concept is limited to the Apple’s iOS devices it provides an interesting outlook on how web applications can be integrated and on par with native applications without an obvious difference to the user.

4.6. Summary

The Merck Toolbox is a support application for sales representatives, which provides a central source for information about the company and its products. Sales representatives use this information when they prepare for customer meetings. The application that is currently in use is based on the Adobe Flash platform and requires Adobe Flash to be installed in order to run. To use the application the user has to always have a connection to the server. The Merck Toolbox does not feature an offline mode.

To show how the new web standard candidates are capable of supporting the development of sophisticated applications and at the same time work on many different platforms and operating systems, an HTML based prototype was implemented that offers the same functionality as the existing Merck Toolbox. Additionally, to illustrate the advanced capabilities of the upcoming standards, the HTML based version is equipped with offline support. This allows sales representatives to download files to their local hard drive and use them even when no Internet connection is available. This demonstrates how web applications are now capable of supporting the full spectrum of client-server architectures.
4. Feasibility Analysis

Although it was possible to develop and run the prototype for various platforms there are still some open issues that leave room for further improvements. Additionally, not every browser supports all upcoming standards. In the case of the Toolbox prototype only Google Chrome provided all required features to implement the full application functionality. Fortunately, with future browser versions, the support for modern web standards will increase and the implemented prototype will gain a much broader support.

For Apple’s iOS the prototype was extended with some additional functionality solely available on Apple’s iOS platform, which allows web applications to mimic the look and feel of regular binary iOS applications. This way an iOS web application can hide the browser controls and add an icon on the desktop. This shows how the border between native applications and open standard web applications blurs more and more.
5. Conclusion and Outlook

After the chances and possibilities of the new specifications and standards have been outlined in the previous chapters, this section will now outline potential risks that always have to be considered when deciding to develop a web application using the new web technologies.

5.1. Unfinished Standards

First of all it has always to be kept in mind that most of the new specifications are still in the stage of working drafts. This means that although there are implementations available, these specifications are still potentially changed or even discontinued. An example for such a discontinuation of a standard is the already described WebSQL. Due to the fact that it was implemented by many browser vendors using the same database engine, the W3C working group decided not to make it a standard although it was already used by web applications.

A similar issue is currently discussed for the Web Storage specification (Barstow, 2011). As it turns out there are unresolved issues related to this technical report where no common agreement could be found. Additionally, each browser vendor has implemented a slightly modified version of the specification. The working group now has to examine whether to address the open issues or just cancel the specification efforts. It is not yet clear what the decision will look like.

A third example for problems when dealing with incomplete standards is the specification of Web Sockets. Although Opera and Mozilla implemented this specification, they later revoked the support in their web browsers due to security concerns (Heilmann, 2010; van Kesteren, Disabling the WebSocket protocol, 2010). Today these issues have been addressed and both vendors have decided to re-enable the support for Web Sockets. The whole issue gives an example how quickly incomplete standards can render a web application, that already relies on them, useless.
Another widely discussed issue is the supported video format for the HTML5 video element. Currently Microsoft and Apple favor the proprietary MPEG-4 video format while Google, Opera and Mozilla rely on the open-source video container WebM. The struggle on which format will gain the most acceptance has been going on for quite a while and has emerged into a more political then a technical dispute. Google, who originally supported both formats, decided to continue to only support WebM in Chrome (Jazayeri, 2011). They started converting all available videos on YouTube into WebM (Zern, 2011) and even announced to develop a WebM plugin for the Internet Explorer (Google Inc., 2011h). Microsoft countered this by announcing an MPEG-4 plugin for Google’s Chrome (Microsoft Corporation, 2011f). At the moment it is not clear which format will succeed. This means that a web application currently has to host different versions of the same video file. And even when a unified format will be chosen for backwards compatibility’s sake, the two formats will still have to be supported for some time.

5.2. Microsoft Internet Explorer

When looking on the already quoted StatCounter (StatCounter, 2011b) browser usage statistics, it is evident that on smartphones and modern tablet devices Microsoft’s Internet Explorer (IE) does not play a significant role.

However, on notebook computers and netbooks, the market share is completely different. When combing all different versions with over 50% Microsoft’s Internet Explorer is still the market leader (Net Applications, 2011b). Nevertheless, Microsoft’s web browser is the one with the lowest standard compliance ratings. This dilemma is illustrated in Figure 11.
As the figure shows, a decent support for the new standards requires at least Internet Explorer 9. Unfortunately, the latest version of Microsoft’s web browser is not available for the still widely deployed Windows XP. This means that using the newest web technologies will expel all Internet Explorer users with versions prior to 9 and by that over 45% of all users. Fortunately, Microsoft has started to catch up on the most recent web standards with Internet Explorer 10 but this release will also not support Microsoft XP. Additionally, the newer release broke a lot of web sites that were especially developed for IE6. Particularly in companies like Merck, a shift from IE version 6, 7 or 8 to 9 or 10 is therefore made with great precaution.

Although recent versions of browsers like Chrome, Opera and Firefox support much more of the new web standard proposals they only have a cumulated market share of 25%. The impact of Microsoft’s Internet Explorer has always to be kept in mind when developing web applications with the latest technologies.
5. Conclusion and Outlook

While private web users can easily switch between different web browser vendors, in the business environment this is a more complex endeavor. As already stated many business web applications are only compatible with older versions of the Internet Explorer. To solve this problem Google has introduced the Google Chrome Frame plugin for Internet Explorer (Google Inc, 2011a). This plugin is available for IE 6 onwards and brings Chrome’s rendering and JavaScript engines to the Internet Explorer. The great benefit of this plugin is, that each web site can individually be configured to run using the Internet Explorer engines or the Chrome engines by adding a simple HTTP header (Listing 3) or an HTML Meta tag (Listing 4) for the relevant page. This way newly developed web pages can be run using the Chrome Frame, but older web applications stay untouched and still use old the IE engine. The advantage for the users is that they only need one browser. The selection of the proper rendering engine is done automatically and unnoticed by the user.

```
X-UA-Compatible: chrome=1
```

Listing 3: Adding the additional tag to the HTTP header will cause it to be displayed using the Chrome Frame

```
<meta http-equiv="X-UA-Compatible" content="chrome=1">
```

Listing 4: Adding the additional tag to a HTML page will cause it to be displayed using the Chrome Frame

5.3. Roadmaps and Previews

One important aspect when planning to develop or extend a web application is to know when a standard feature will be implemented in each browser and be widely available. To achieve a certain planning security it is therefore important to have open development roadmaps from each browser vendor. This way it could always be estimated if and when dedicated features are available.
5. Conclusion and Outlook

Unfortunately, except for Mozilla, developer of the Firefox web browser (Mozilla Foundation, 2011b), no vendor has released a detailed feature roadmap. However, many browser vendors have at least established mechanisms for web application developers to test previews and unstable versions of their upcoming browser releases. But before the release of a preview build, there is no or little information available on what features will be implemented and especially when they will be available. As both WebKit and Gecko are open source browser engines, it of course is always possible to monitor the changes in the source code directly, but this also does only provide a fairly limited preview. The only other way to stay informed about the upcoming features is to closely monitor the unstable preview releases.

Both, Firefox and Chrome have established a multilevel release cycle that gives very early access to unstable versions. Mozilla provides four channels that are regularly and automatically updated and to which a user can subscribe (Legnitto, 2011). The stable channel holds the latest public release of their browser. The beta channel contains fairly stable but not yet fully tested builds. The aurora channel provides continues unstable and only sparsely tested code. Finally, the nightly channel offers daily updated builds that might still contain severe bugs and error. Google Chrome’s model is similar they also have a stable channel and a beta channel. Their counterpart for the aurora stage is called developer channel. Only for nightly builds Google does not provide any automatic release mechanisms. Those daily updated builds have to be downloaded manually. (Google Inc., 2011c)

Apple does not provide any continues previews for their browser UI. However, nightly builds of their rendering engine WebKit can be downloaded and the most recently added features to the engine can be tested (Apple Inc., 2011e). For their iOS version of Safari, Apple provides beta versions of upcoming iOS releases only sporadically and only to paying developers.
5. Conclusion and Outlook

Opera has just established a browser release entitled Opera Next (Opera Software ASA, 2011c). This continuously updated release channel always holds the pre-release version of their desktop browser. For Opera mobile, previews are only occasionally provided.

Microsoft has created two different ways of previewing upcoming features to the user. For specifications that will be implemented in their upcoming releases of their web browser, Microsoft offers so called platform previews for download (Microsoft Corporation, 2011a). Those pared-down versions to not provide any web browser UI but simply contain the rendering engine. In addition to those unstable releases Microsoft also provides a set of installable plugins that bring additional web technologies to the browser. These plugins are a playground for Microsoft and web developers to test potential web specifications even though Microsoft considers them as too incomplete for adding them to their platform preview (Microsoft Corporation, 2011d).

The Android and the BlackBerry web browsers are directly included in the operating system. Therefore no individual previews are available. Depending on the vendor’s decision, preview releases of upcoming operating systems can be tested, which also involve new versions of the web browsers.

5.4. Security concerns

With the upcoming web standards, web applications will be able to use a whole new set of features to provide reach and sophisticated functionality. However, in contrast to locally installed applications, web applications bear a much higher security risk. Without proper precaution, a webpage visited either on purpose or by accident, could write to the local hard disk or access the address book, the camera and other device information. It is therefore very important to give the user detailed privacy and permission controls.
5. Conclusion and Outlook

A second aspect is the protection of the locally stored content. There are currently no methods to encrypt and secure locally saved data. Either through the web browsers data inspector or sometimes even through the local file browser locally stored data is accessible.

How permission management has to be implemented is currently not specified anywhere. Web browser vendors are free to find their own way of protecting the user’s privacy. Additionally, browser bugs or inadequacies in the specification may allow malicious web sites to overcome the privacy settings e.g. by using cross-site-scripting. It remains to be seen how these issues will be addressed. Currently very few of the new features are in extensive use, which is why no detailed analysis is possible at the moment, but security and privacy of web applications and the underlying standards have to be closely monitored.

5.5. Application Stores

When looking at some of the websites with very high visit rates like Amazon, Google, eBay or Facebook, it is apparent that nearly all of them do not only offer a specialized version of their web site optimized for smartphone screens, but also a native application on Android or iOS. The question that now arises is why companies that run their business through a web site and do not offer any native applications on Windows, Mac OS X or Linux, develop individual applications for each smartphone platform although they have already optimized their web site for smartphone devices.

One of the reasons might be that, for example, the Amazon application uses the smartphone’s camera to let the customer scan barcodes of products in a local store to quickly find the same product on Amazon. As already explained in chapter 3.5.1 there are specification efforts which allow accessing the camera directly through the browser, but until now no smartphone browser has implemented this API. With this in mind one possible reason why some companies have developed individual applications for each smartphone operating system is that some features are just not available through web technologies.
However, there are many other web sites that also maintain a specially adjusted application that does not provide any additional features than the small screen version of their web site. At first it seems confusing that a company would invest money to develop native applications for each individual platform although their website with the same functionality is already available on all these devices. A possible explanation for this overhead is the huge success of application stores that are primarily present on smartphones and tablets. Such application stores provide a centralized platform where users can easily search, buy, download and install applications for their individual device. These application stores also host regularly updated lists of most recently added or top selling applications, introducing new ways for customers to discover new software. In addition, it is possible to rate and comment applications, which helps other customers with their purchase decision. Because application stores are such a great way to distribute mobile applications, many companies might just transform their web site into a native application to be listed in an application store and gain additional attention.

One of the problems with web applications is that they are currently not listed in any of the application stores available on iOS or Android. This means that there currently is no convenient way to discover and buy web applications on mobile devices. However, to compete with native applications, web applications should not only provide the same user experience within the application but also during the purchasing process. A way to push web applications would therefore be to establish suitable web application stores. Currently Mozilla and Google are trying to lay the base for web applications:

**Mozilla’s Open Web Applications:** Instead of providing a dedicated web application store Mozilla currently only specifies a standard proposal on how any web application can be hosted by any store (Mozilla Foundation, 2011e). To accomplish this, the web application developers have to generate a set of additional Meta data documents that describe type, functionality and requirements for their web applications. Any web site can then use this Meta data to make web applications searchable and installable.
5. Conclusion and Outlook

by the user. Unfortunately, the installation process requires the browser to implement and provide a special open web applications API, which is currently only provided in Mozilla’s Firefox web browser.

**Google Chrome Web Store:** In contrast to Mozilla, Google takes a different approach. Instead of trying to establish a common format that can be used by any website, Google has introduced the Chrome Web Store, a web application store that is only available in Google Chrome (Google Inc., 2011d). To publish an application in the store, similar to Mozilla’s approach, the developer has to add metadata describing the application. The web application can then be browsed and installed by the user. It is even possible to post comments and reviews. In theory it is also possible to host paid apps on the web store, but currently all applications are freely available. Some however might require a special account from the web application’s operator, which is not free.

It remains to be seen if and how these web application stores can bring a unified application store experience to all major platforms. Nevertheless, well-made comprehensive web application stores might be an important aspect for a rosy future of web applications.

5.6. Improving the Prototype

Although it was possible to develop the Merck Toolbox application using only open and standardized web technologies as shown in chapter 4.3, there are still open issues left that leave room for further improvement.

5.6.1. Disk Space Quota Management

One problem that occurred during the development of the offline functionality of the Toolbox is a missing quota management indicating how much disc space a web application can use for local databases, local files and the offline application cache (compare chapter
5. Conclusion and Outlook

3.5.3). Such control mechanisms are especially required because it would be a huge security risk if every web site or web application a user visits could download unlimited amount of data to the user’s hard drive. Although the HTML5 specification explicitly requires web browsers to applying constraints on disk usage it does not define how quotas should be handled:

“How quotas are presented to the user is not defined by this specification. User agents are encouraged to provide features such as allowing a user to indicate that certain sites are trusted to use more than the default quota, e.g. by asynchronously presenting a user interface while a cache is being updated, or by having an explicit whitelist in the user agent’s configuration interface.” (Hickson, HTML5, 2011, Section 5.6.8)

Unfortunately the lack of a detailed specification does currently drive browser vendors to building their own solution and to assign their individual quota limitations for each site. As there is no standardized API to query the remaining disk quota through JavaScript building web applications that require a large amount of local data. The offline functionality of the prototypic implementation of the Merck Toolbox does currently only run in Google Chrome builds if the quota control is disabled. This, on the other hand, does allow any web site to store unlimited data, which is a high security risk.

To solve this problem the Google Chrome team has started working on a unified quota management throughout all local storage APIs to allow convenient configuration of the quota on a site-by-site basis (Chromium Development Team, 2011). On top of that the Chrome developers are currently implementing a JavaScript API that allows a web application to query the disk space it currently occupies, how much disk space it has still available and call methods to ask for a quota increment. If the Merck Toolbox prototype would utilize this API, it would be possible to conveniently increase the application’s disk quota based on how much content the user would want to make available offline.
5. Conclusion and Outlook

At the moment this unified disk quota management and especially the quota API are only becoming a part of WebKit-based web browsers. It remains to be seen if other browser vendors will adopt this API or if it will be standardized by the W3C or a different standardization organization. However, WebKit-based web browsers have gained quite a market impact, especially on smartphones and tablets, which is why Google’s proposal might gain broad acceptance. The Merck Toolbox prototype could therefore be improved by adding support for this API giving the user transparent control of the offline files and their disk space usage.

5.6.2. Data Encryption

Another issue that could not be addressed during the prototype development is the security aspect of locally persisted data. Although there is no possibility to encrypt IndexedDB or LocalStorage data in sight, there are some proposals that could improve the decryption performance of files stored using the File API.

The first approach is to continue to use decryption routines that are entirely written in JavaScript. As working with binary file formats in JavaScript required a new type of binary data types (Herman & Russel, 2011) those new types have not yet undergone the same optimization efforts than regular type-less JavaScript arrays. Over time browser vendors will continue to optimize the processing of the new array types, which might lead to a significant performance boost.

A second way to improve the JavaScript performance is the introduction of WebCL (Khronos Group, 2011b). WebCL, based on OpenCL, is a JavaScript API that allows applications to run code directly on a device’s Graphics Processing Unit (GPU), which in case of graphical or cryptographic algorithms can offer a huge performance improvement. OpenCL is not yet specified but a work group has started creating a specification hosted by the Khronous Group. With Samsung’s WebKit port there is already a very early implementation of WebCL available (Samsung Group, 2011).
5. Conclusion and Outlook

Another performance bottleneck is that the files have to be completely decrypted before they can be displayed. This is due to the fact that the current File API specifications require the whole file to be available in the browser memory before it can be used. To address this, a Streaming extension for the File API is currently discussed (Malcontenti-Wilson, 2011), which would allow accessing the file data while it is still loaded or decrypted. The advantage would be that the Merck Toolbox could start playing a video file while the end of it is still being decrypted.

A completely different approach is to specify a cryptographic JavaScript API directly within the browser. Instead of porting every cryptographic algorithm to JavaScript the web browser could use well-tested and highly efficient native implementations and simply provide a JavaScript wrapper to make them available to web applications. Thanks to their secure HTTP (HTTPS) support many web browsers do already come with internal support cryptographic standards. With a convenient JavaScript API, these internal methods could be made available in web applications for efficient en- and decryption of any larger data. David Dahl, a Firefox developer, has proposed such a wrapper API entitled DOMCrypt (Dahl, 2011) and already implemented a Firefox extension as proof-of-concept. His specification is not implemented into Firefox directly nor has it been ported to any other browser. However, there is quite some discussion on WHATWG and W3C mailing lists about his proposal. It will be interesting to see if this idea turns into a standard eventually and if it does it could be used to efficiently handle decryption in the Merck Toolbox prototype.

5.6.3. Alternative Document Format

Especially on tablets and smartphones, displaying PDF files is not always convenient and simple because they are often displayed using a dedicated PDF viewer application. To view a PDF hosted on a web application the user usually has to switch to the PDF application and then back to the browser to continue working with the web application. As shown in chapter 4 on devices like the iPad or Notebooks it is also possible to display PDF documents inside of
an HTML iframe-element or new web browser window but this still might break the user experience or alter the look and feel of the application.

A possible solution to this problem would be to not rely on PDF documents but use native document formats. In this case a native document format is considered a file structure that utilized standard web technologies to layout and display its content. Instead of converting a company presentation into a PDF file, it could instead directly be created as an HTML document that holds the same information but in a way that can conveniently be integrated into the application and displayed by the browser. The document format S5 provides exactly that: A slide show system that supports creating HTML presentations, which can be seamlessly displayed directly in the browser (Meyer, 2011). Unfortunately, S5 and similar formats have not gained much attention throughout the web and there are no conversion tools from standard applications like Microsoft PowerPoint. To port the already available Merck presentations from PowerPoint to S5, every presentation would have to be recreated. Although, this is a larger investment at first, in the long run it would allow direct integration and quick and simple display of the company presentations within the web browser.

But even when sticking with PDF files the user experience could still be significantly improved by getting rid of the external PDF viewer. Andreas Gal and his team have developed a full-featured PDF viewer in JavaScript (Gal, 2011). Instead of relying on an external viewer or a browser plugin, the JavaScript PDF viewer could be integrated into the Merck Toolbox prototype, which would allow the display of PDF files directly within the browser. This way existing PowerPoint presentations, Word documents or any other printable format could conveniently be converted to PDF and from there displayed within the Merck Toolbox prototype. It has to be kept in mind that this approach is highly experimental at the moment, but it offers an insight on how powerful the new HTML, CSS and JavaScript technologies are and how the Merck Toolbox user experience could be further improved.
5. Conclusion and Outlook

5.7. Summary

In chapter 2 it was shown that developers of mobile applications are currently confronted with a very heterogeneous landscape and that developing mobile applications for all these different platforms and operating systems is a time consuming and expensive task. It was stated that currently only web applications based on open and standardized web technologies such as HTML, CSS and JavaScript are the only way of developing cross-platform applications in a cost efficient way. Unfortunately, until now such web applications were not entirely sufficient to cover the whole spectrum of client-sever architectures. Only web applications working as thin clients were possible. In addition, many features of native applications or plugin-based web applications were not possible to implement with current web standards.

However, in chapter 3 it was described that with the new upcoming web technologies that are currently getting standardized and implemented in web browsers, these shortcomings of web applications are about to be addressed. Many use cases that could not be done before can now be implemented. Thanks to upcoming concepts such as SVG, Media Queries and CSS Layout managers, web applications are not only a very convenient way to develop sophisticated applications for a broad set of different operating systems and platforms, but also favor different screen sizes and device types.

To demonstrate that soon open standard web applications will be able to compete with native applications or web applications with plugins and on top of that provide nearly unlimited platform support, in chapter 4 a prototype using many of the new web technologies was implemented. Although there are still some open questions concerning the prototype implementation, it was demonstrated that sophisticated platform-independent mobile applications could be realized using open web standards.

Finally, in chapter 5 some concerns were raised that might hinder the success of web applications as sustainable technology for mobile application development. Common problems include the fact that most of the new web technologies are not final standards, and that
browsers with significant market share do not support many of the new concepts. Open security concerns and missing roadmaps are still potential risks when orienting towards web technologies. Fortunately, in the future more and more browsers will support all of the presented standards, which will make web applications the only way to cost-efficiently develop, deploy and maintain mobile applications.
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CGI</td>
<td>Common Gateway Interface</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading Style Sheets</td>
</tr>
<tr>
<td>CR</td>
<td>Candidate Recommendation</td>
</tr>
<tr>
<td>DOM</td>
<td>Document Object Model</td>
</tr>
<tr>
<td>DRM</td>
<td>Digital rights management</td>
</tr>
<tr>
<td>ECMA</td>
<td>European Computer Manufacturers Association</td>
</tr>
<tr>
<td>GB</td>
<td>Gigabyte</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processing Unit</td>
</tr>
<tr>
<td>GTK+</td>
<td>GIMP Toolkit</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>HTTPS</td>
<td>Hypertext Transfer Protocol Secure</td>
</tr>
<tr>
<td>IDC</td>
<td>International Data Corporation</td>
</tr>
<tr>
<td>IE</td>
<td>Microsoft Internet Explorer</td>
</tr>
<tr>
<td>IndexedDB</td>
<td>Indexed Database</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>JRE</td>
<td>Java Runtime Environment</td>
</tr>
<tr>
<td>JS</td>
<td>JavaScript</td>
</tr>
<tr>
<td>MathML</td>
<td>Mathematical Markup Language</td>
</tr>
<tr>
<td>MB</td>
<td>Megabyte</td>
</tr>
<tr>
<td>MPEG</td>
<td>Moving Picture Experts Group</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>PDA</td>
<td>Personal Digital Assistant</td>
</tr>
<tr>
<td>PDF</td>
<td>Portable Document Format</td>
</tr>
<tr>
<td>PHP</td>
<td>PHP: Hypertext Preprocessor</td>
</tr>
<tr>
<td>PR</td>
<td>Proposed Recommendation</td>
</tr>
<tr>
<td>RAM</td>
<td>Random-access memory</td>
</tr>
<tr>
<td>REC</td>
<td>Recommendation</td>
</tr>
<tr>
<td>RIM</td>
<td>Research in Motion</td>
</tr>
<tr>
<td>SGML</td>
<td>Standard Generalized Markup Language</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>SVG</td>
<td>Scalable Vector Graphics</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
<tr>
<td>WD</td>
<td>Working Draft</td>
</tr>
<tr>
<td>WebCL</td>
<td>Web-based Computing Language</td>
</tr>
<tr>
<td>WebGL</td>
<td>Web-based Graphics Library</td>
</tr>
<tr>
<td>WHATWG</td>
<td>Web Hypertext Application Technology Working Group</td>
</tr>
<tr>
<td>XHR</td>
<td>XMLHttpRequest</td>
</tr>
<tr>
<td>XHTML</td>
<td>Extensible HyperText Markup Language</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
Appendix

Installing and Running the Prototype

The following steps are required to run and test the prototype implementation discussed in this thesis. There are two packages available: One for Microsoft Windows and one for Apple Mac OS X. The packages were tested on Windows XP, Windows 7 and Mac OS X 10.6.

1. If not already available, install Google Chrome (at least version 11) on the target system. Installers for both Windows and Mac OS X can be found on CD number 1 labeled “Thesis, Bibliography, Code, Software, Screenshots and Demo Video” in the folder “Installer”.
2. Insert CD number 2 labeled “Prototype Packages”.
3. On Microsoft Windows operating systems extract the contents of the archive file “toolbox_package_win.zip” to the base directory of the root partition (usually “C:\”). On Apple Mac OS X extract the contents of the archive file “toolbox_package_mac.tar.gz” to the applications folder (usually “/Applications”).
4. Now browse to “C:\xampp” (Windows) or “/Applications/XAMPP” (Mac OS X) and run the XAMPP Control application. Within the application use the “Start” button to launch both the MySQL and the Apache server.
5. Start the web browser Google Chrome (at least version 11) with the following two parameters:
   --unlimited-quota-for-files
   --unlimited-quota-for-indexeddb

   Note: For both Microsoft Windows and Apple Mac OS X the folder “Scripts” on CD number 2 contains launch scripts to execute Google Chrome with these parameters.
6. Type the URL http://localhost/toolbox into the address field of Chrome. The login screen of the prototype application should appear.
7. Use the username “guest” and an empty password as login credentials. Some initial test data is already available.
Appendix

Merck Toolbox

Figure 12: Merck Toolbox welcome screen with animated clock (Flash)

Figure 13: Customizable favorites on the left and category browser on the right (Flash)
**Figure 14:** File browser with documents and video files in different languages (Flash)

**Figure 15:** View videos and documents directly within the application (Flash)
Figure 16: Merck Toolbox welcome screen with animated clock (HTML)

Figure 17: Customizable favorites on the left and category browser on the right (HTML)
Appendix

Figure 18: File browser with documents and video files in different languages (HTML)

Figure 19: View videos and documents directly within the application (HTML)
Figure 20: Download cart for downloading selected files to the local hard drive (HTML)

Figure 21: Monitoring the download progress with pop-up notification on success (HTML)
Appendix

Figure 22: Offline mode showing only local files if no connection is available (HTML)

Figure 23: Merck Toolbox running in full screen mode on an Apple iPad (HTML)
Figure 24: Merck Toolbox prototype running on a Motorola Xoom (HTML)


Apple Inc. (2011c, June 06). iOS Developer Program License Agreement. Cupertino, California, USA.


Bibliography


Bibliography


Bibliography


Bibliography


Bibliography


Bibliography


Bibliography

https://apps.mozillalabs.com/


http://monotouch.net/


Bibliography


http://searchmobilecomputing.techtarget.com/definition/notebook-computer

http://searchmobilecomputing.techtarget.com/definition/tablet-PC


http://www.w3.org/TR/contacts-api/

http://www.w3.org/TR/calendar-api/

Bibliography


Bibliography
