Ashish Thapa

HTML5 AS A CROSS-PLATFORM FOR MOBILE APPLICATIONS DEVELOPMENT

Ashish Thapa
Bachelor's Thesis
Winter 2012
Degree Programme in Information Technology
Oulu University of Applied Sciences

PREFACE

The following research was commissioned by Comvise Oy, Oulu. It provides Software development and Integration Services. Consultation is another major expertise of Comvise.

This topic came from Comvise to do an extended research on the evolving technology HTML5, mainly targeting at multimedia and storage. The research was made about the features and support of multimedia and storage of HTML5 for the mobile operating systems. An application was built as a proof of concept. I would like to express my sincere gratitude to my thesis supervisor, Mr Veikko Tapanien. He has been extremely encouraging and he has provided me with his invaluable support from the beginning.

Mr Binay Guragain, the company representative, has been one of the key contributors to my thesis. I would like to thank him, for providing guidelines about the topic and support throughout the research process.

I would like to thank Mrs Kaija Posio, for checking the language of my thesis. She has always been very helpful. Another person I must mention is Mr Jarmo Karppelin, for approving the thesis topic, assigning such understanding and being an active supervisor. Last but not the least, I would like to thank Comvise, for giving this wonderful topic to work with, OAMK for being very supportive, my family and friends for their forever support.

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Degree Programme in Information Technology

Tekijä: Ashish Thapa
Opinnäytetyön nimi: HTML5 AS A CROSS-PLATFORM FOR MOBILE APPLICATIONS DEVELOPMENT
Työn ohjaaja: Veikko Tapaninen
Työn valmistumislukukausi ja -vuosi: Talvi, 2012 Sivumäärä: 73

Viimeinen puolivuosikymmen on ollut erittäin eloisa netti- ja mobiiliteknologioiden alalla. Ala on nähnyt monta mullistavaa innovaatiota, ja HTML5 on yksi niistä. Comvise Oy pyysi tutkimaan sen kahta odotetuinta ominaisuutta, Multimedia ja Storage. Tavoitteena oli tutkia niiden ominaisuuksia ja kuinka ne ovat tuettuina tärkeimmissä mobiilikäyttöjärjestelmissä. Tutkimus tehtiin siten, että lukija saa käsityksen niiden ala-aiheista, ohjeita toteutukseen ja niiden yhteensopivuudesta.

Toinen tavoite oli tehdä PhoneGap-ohjelma käyttäen nettiteknologioita, HTML5:ttä, CSS:ää ja JavaScriptiä, jotka voidaan portata tärkeimmille mobiilikäyttöjärjestelmille jotta voidaan näyttää toteen sen cross-platform-luonne. Se oli mielenkiintoista koska nettiteknologiat käyttivät puhelimien natiiveja toimintoja. Käyttöliittymän tekemisessä käytettiin Sencha Touch 2:ta Android- ja iOS-käyttöjärjestelmissä ja jQuery Mobileä Windows Phone -käyttöjärjestelmässä.

Ohjelma antaa käyttäjän ottaa kuvia tai tallentaa videota Secure Digital (SD) -muistikortille. Otetut kuvat lähetettiin palvelimelle. Ylimääräistä ominaisuutta, Facebook Graph Application Programming Interface (API), testattiin myös ohjelman jatkokehitystä varten. Samaa koodia käytettiin jokaisessa kolmessa mobiilikäyttöjärjestelmässä, mutta testiympäristöt luotiin erikseen jokaiselle käyttöjärjestelmälle.

Asiasanat: HTML5, Cross-platform, Multimedia, Storage, Hybrid, PhoneGap, Application

ABSTRACT

Oulu University of Applied Sciences
Degree programme in Information Technology

Author: Ashish Thapa
Title of Bachelor's thesis: HTML5 AS A CROSS-PLATFORM FOR MOBILE APPLICATIONS DEVELOPMENT
Supervisor: Veikko Tapaninen
Term and year of completion: Winter, 2012 Number of pages: 73

The last half a decade has been extremely vibrant in the field of web and mobile technologies. The industry has seen some revolutionary innovations, HTML5 is one prospect. Comvise wanted me to do the research on two of the most awaited features of HTML5, multimedia and storage. The objective was to study about the features of Multimedia and Storage, and their support on major mobile Operating Systems. The research was made so that it provides the reader, information about the sub-topics, guidelines for implementing the features and their compatibility.

Another objective of the research was to build a PhoneGap application using the web –technologies, HTML5, CSS and JavaScript, which could be ported to major mobile operating systems to prove its cross-platform nature. It was interesting because, the native functionalities of the phone were being accessed by web technologies. Sencha Touch 2 was chosen to build the user interface for Android and iOS operating systems and jQuery Mobile was used for Windows Phone.

The application allowed users to capture images and record video which are then stored locally in Secure Digital (SD) card of the phone. The images captured were uploaded to the server. An additional feature, implementing Facebook's Graph Application Programming Interface (API) was tested in the application for further development of the application in future. The same code base was used in all three major mobile operating systems but the environment was set-up individually for each operating systems.

Keywords: HTML5, Cross-platform, Multimedia, Storage, Hybrid, PhoneGap, Applications

CONTENTS

1 SYMBOLS AND ABBREVIATIONS

Listed below are the abbreviations used in the whole document and their respective full form.

Abbreviation	Detail
2D/3D	Two Dimensional / Three Dimensional
.apk	Android Application Package
ADT	Android Development Tools
API	Application Programming Interface
BLOB	Binary Large Object
CSS	Cascading Style Sheet
CPU	Central Processing Unit
DTD	Document Type Definition
DOM	Document Object Model
GPU	Graphical Processing Unit
HTMLWG	Hypertext Markup Language Working Group
HTML	Hypertext Markup Language
HTML 4.0	Hypertext Markup Language version 4.0
HTML5	Hypertext Markup Language version 5
HTTP	Hypertext Transfer Protocol
ID	Identity
IDE	Integrated Development Environment
iOS	iPhone Operating System
JSON	JavaScript Object Notation
KB	Kilobyte
MATHML	Math Markup Language
MIT	Massachusetts Institute of Technology
MP3/MP4	Media Player version 3 / version 4
MPEG	Motion Pictures Experts Group

MSDN	Microsoft Developer Network
PHP	Hypertext Preprocessor
SGML	Standard Generalized Markup Language
SD	Secure Digital
SDK	Software Development Kit
SVG	Scalable Vector Graphics
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal Serial Bus
WHATWG	Web Hypertext Application Technology Working Group
WWW	World Wide Web
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language

2 INTRODUCTION

Comvise is a telecommunications software company, which provides a software development and integration service and a consultation service to world's technology innovators. Comvise has been actively providing software services for a decade. It has been closely associated with the mobile phone industry. The company details can be found at http://www.comvise.com.

Comvise has always been an innovative company. One of the key features that make the company brilliant is company's constant search for new areas to develop. Comvise is constantly looking for future development opportunities and responses to it with a great deal of research and analysis. Recently, there has been a lot of discussion about "can HTML5 stand out as a cross-platform technology for different operating systems available?" No doubt, W3C is trying to make all the specifications as a stable standard, but there is not yet a definite answer to this question.

The company therefore wanted to carry out a research about the multimedia and storage of HTML5, and the support over different mobile operating systems. I was always curious if there could be one platform to develop an application for different mobile operating systems. I got the opportunity to carry out research for Comvise. The company wanted me to use the PhoneGap framework with Sencha Touch and jQuery Mobile UI frameworks to create a PhoneGap hybrid application. The PhoneGap application was built and tested over three major mobile operating Systems, Android, iOS and Windows Phone as a proof of concept.

To extend the PhoneGap application to the next level, one of the major functionality of Facebook was also implemented in the application

3 HTML5

HTML5 is the fifth version of HTML, a language used to structure and present content for the World Wide Web. It aims to improve HTML4 by adding new features and support to it. HTML5 is a response to various incompatibility issues and syntax errors from previous HTML versions. (2Expert 2012, date of retrieval 22.4.2012).

3.1 The Origin and Evolution of HTML

In 1989, Tim Berners Lee realized the need for a language to share and link the text pages remotely, and he proposed an Internet based hypertext system. To back up this proposal, Lee wrote a browser that could interpret the page and server software to store the original file. He applied SGML to create HTML. It was the same markup language, he had used to create a similar software named ENQUIRE. The difference from SGML was the concept of hyperlink. In HTML, Lee suggested to link text files within themselves through hyper linking the text. The idea was to create a cross reference among the text files to an immediate display of the related files which were to be the basics of the WWW. Dan Connolly, one of the founders of W3C, along with his colleagues collected, analyzed and drafted the widely used HTML tags. He also wrote the document type definition for HTML 2. In 1995, HTML was extended with more HTML tags and published as Internet Draft HTML 3 instead of the standard. The reason behind this change was that, the draft was way too large to be a standard and included a large number of new proposals. W3C's HTMLWG negotiated and altered the draft according to the need and released it as HTML 3.2 standard in 1996. The markup for mathematical formula was completely dropped from the HTML standard in the HTML 3.2 version and later it was standardized in MATHML. (Ragget, lam, Alexander & Kmiec 1998, 17-34).

In December 1997, W3C published HTML4.0 as a recommendation, with three different variations Strict, Transitional, and Frameset. In 1998, a minor editing was made to HTML 4.0 but the same version number was used. In 1997 HTML 4.01 was published. After that W3C took an XML-based markup Language XHTML over HTML. XHTML 1.0 was released in January 2000. It was a mere reformulation of HTML 4 document in the XML version. The next recommendation of XHTML 1.1 was released in May 2001. Despite the issues like implementation, compatibility and interoperability, W3C continued XHTML as a future of markup which created a rebellion inside the W3C. The representative from Apple, Opera, and Mozilla proposed to choose HTML over XHTML which W3C declined. In 2004, the unhappy members created a working group called WHATWG which was independent of W3C. W3C continued to work on XHTML whereas WHATWG started to work on HTML5 .In 2006 the founder of HTML, Tim Berners Lee admitted that an XML-Based Markup was not delivering as expected, and finally in 2007 W3C accepted the new proposal from WHATWG to develop HTML5. Since then W3C has been in charge of HTML5. Currently two groups are working on HTML5 namely WHATWG and HTMLWG. (Ragget, Iam, Alexander & Kmiec 1998, 17-34).

3.2 The Existence of HTML5

HTML5 certainly has additional features over the earlier versions of HTML. The real reason behind HTML5's existence was the complexity and confusion of HTML 4.1 and XHTML, their interoperability and compatibility issues.

XHTML 1.0 was no different compared to HTML 4.0.1, except that its syntax was strict and XML based. XHTML 1.1 was completely based on XML and did not support the mime type "text/html". The bigger problem was XHTML 2.0, which was not backwards compatible and the support for the existing web content was very poor. Since, the web seemed to go nowhere, the representatives from Opera, Apple and Mozilla were not happy. Ian Hickson an employee from Opera proposed revival of HTML, which was rejected by W3C. The unhappy members then formed Web Hypertext Application Technology Working Group (WHATWG). The group started to work on two different specifications Web Forms 2.0 and Web Apps 1.0, later to be merged and evolved as HTML5. The proposal of HTML5 was made to W3C, which was accepted in 2007. Currently the two working groups HTMLWG and WHATWG are working on HTML5. WHATWG is creating HTML5 specification which is then reviewed by HTMLWG and published as a HTML standard. (Keith 2010, 1-8).

3.3 Delta from Original Versions

W3C has stated that “HTML5 is still a draft” and its content is continuously evolving but it is clear that it will replace HTML4 which had an issue of interoperability and backward compatibility with HTML and critical mass of deployed content. Same applies to XHTML 1.1, which is “an XML serialization” of HTML4 and DOM level 2 HTML. HTML5 is compatible with both HTML 4 and XHTML 1.1 but offers no compatibility with more esoteric SGML features of HTML4. The Doc-type is case insensitive in HTML5 and not in HTML4 because HTML4 was based on SGML. It also required a reference to the DTD. The declaration is <! DOCTYPE html > with an option for XML. With the goal “once released should be usable” the specification will not be considered complete until there are at least two implementations. The completeness of the specification is tested by the test suite. In the previous versions, the committee used to recognize the final specification before implementation. HTML5 redefines the parsing rule including an error handling and a processing model, making it largely compatible with other popular implementations. It uses an XML compatible “text/html” mime type for a usual text and “text/htmlsandboxed” for suspicious and sandboxed content. HTML5 defines 3 ways to encode characters, using an HTTP content-type header at the transport level, using a Unicode Byte Order markup at the beginning and using a Meta char-set encoding within the first 1024 bytes. Various new elements like SVG and MATHML are introduced in HTML5 and some existing elements and features are absent. HTML5 introduces a number of APIs to create better web applications and “HTML5 Change-Logs” to let the users know about the changes. (W3C 2012a, date of retrieval 16.1.2012).

3.4 Desktop Browser Support

Most of the major desktop browsers like Internet Explorer, Mozilla Firefox, Google Chrome, Apple Safari and Opera fully support “contentEditable” attribute, “hashChange” event, new semantic elements, Canvas, Video, Audio, inline SVG elements and “getElementByClassName()” method when accessing the DOM elements. All the major browsers except Opera have been supporting and will support Drag and Drop, a defer attribute. All browsers and also Internet Explorer 10+

versions support the Offline Web Application. The session history management is currently supported by Firefox, Chrome and Opera. IE 10.0 announced a complete support and Opera will continue its partial support for this feature. Currently Opera supports fully the form features. Firefox, Chrome and Safari partially support HTML5 Form Features, IE has a partial support in version 10.0 and afterwards. Browsers are trying to maximize their HTML5 support now and then. It is a good idea to identify the target browser and test its support of each individual feature before developing, it further. (@Fyrd 2010, date of retrieval 16.1.2012).

4 INTRODUCTION OF HTML5 IN MOBILE DEVICES

Mobile Phones have changed a lot from being a mere device, with a calling and messaging functionality. Nowadays, Smart Phones offer users a large number of functionalities and varieties of applications. Let us look at some major smart phone operating systems.

4.1 Mobile Operating Systems

iOS:

iOS is Apple's proprietary operating system for mobile devices. Objective-C is used to create mobile applications and Cocoa Touch framework for the User Interface. It provides multi-touch gestures for rich user experience and serves gaming purpose. The native browser is Web Kit based and supports most of the HTML5 features. (Apple Inc 2012a. date of retrieval 18.1.2012).

Android OS:

The Android operating system was initially developed by Android Inc., which was later purchased by Google in 2005. It was then released under the Apache License, which means it was made open source and anybody could use it for free and add extensions to it. The application developed for Android Operating systems, if compatible, runs over the entire devices which uses Android OS. It provides features like multi-touch and multitasking. However, the device used is responsible for the performance. The native Android browser is based on Web Kit and chrome's V8 JavaScript Engine. (Lee 2011, 1-3).

Windows Phone:

Initially, it started as Windows Mobile in 2003 and was replaced by Windows Phone in 2010. Windows Phone is another innovation of the leading OS provider in PC, Microsoft. It is based on the Windows CE kernel and has a Silverlight framework for an application and the XNA framework for a game development. Windows Phone has unique Metro UI with tiles as its components. The application can be downloaded from and released to the Windows Phone

Marketplace. It uses Internet Explorer which previously did not support HTML5 features. It supports HTML5 since version 9 (Zhou, Zhu, Zheng & Yang 2011, 2 - 8).

Blackberry OS:

It was developed by Research In Motion (RIM) for Blackberry devices. BlackBerry devices have smartphone functionalities like a touch screen and a multitasking but they are better known as corporate phones. The latest operating system ships with a Web Kit based native browser. (Hill 2010, date of retrieval 30.1.2012).

Along with the operating system, the hardware that can run the operating system, input and output methods, the processor, memory and screen functionalities are smarter than ever. Smart phone comprises of these components to provide a way to use an application effortlessly. In other words, the “Application is the heart of Smart Phone” and different body parts provide the support to keep it alive. (Buzzle 2012, date of retrieval 30.1.2012).

4.2 HTML5 in Mobile Devices

Apple’s decision not to support flash on iPhone and iPad stirred the existence of plug-ins. In 2011 Adobe decided to discontinue its mobile version of flash and to support HTML5. Flash for a mobile will eventually disappear. However, it is less likely that browsers will completely replace the support for flash over HTML5 in the near future for a desktop. It is recommended to use a flash format in an audio and video element as a fallback option since HTML5 is yet to be a standard and a lot of existing content is flash based. (Adhikari 2011, date of retrieval 9.2.2012).

Not a long ago, most of the applications running on smart phones used to be native. The biggest drawback of a native application is interoperability. An application of one platform does not work on another platform. Leaving aside different platforms, developers sometimes had to create the same application for one platform more than once due to the lack of backwards compatibility. It means a huge resource loss and nightmare for developers to learn different languages. W3C has adopted HTML5 as the future of the web and has started creating a specification targeted at web and mobile devices. HTML5 is a good replacement for content driven native app, where a client can connect to a server, which does not have to store large amount of data and which is platform independent.

There are also frameworks to create hybrid apps, through which a developer can create native-like app from a web app using the APIs. HTML5 apps or web-apps are usually written using HTML, CSS3 and JavaScript and supported by most modern web browsers. To stay connected to the Internet all the time is a big concern. That need was eradicated by HTML5 by providing support for offline web application. This means, apps accessed when online can be accessed later even if there is no connectivity. The offline application is actually an online-offline application, which is cached and the device can access it later. The new HTML5 has a capacity of up to 5MB offline storage in comparison to cookies which could store a maximum of 5 KB and were vulnerable to security threats. Other significant HTML5 features ported to mobile are Canvas, Video and Audio streaming, Geo-location and Local Storage. (Kravchick 2011, date of retrieval 31.1.2012).

It's a challenge to pinpoint every feature that will be supported by all mobile browsers and how long the support will last. The browsers are constantly improving their support and W3C is trying to keep the size of the standard limited by keeping the most commonly used and generic features in the standard and creating APIs for the features that are browser dependent. The other reason for creating APIs is the non-uniformity among the browsers. The related parties who are willing to create HTML5 apps should determine what features a browser would support.

There are JavaScript libraries such as Modernizr, which are able to provide information about the availability of features during implementation phase. (Modernizr 2009-2012, date of retrieval 31.1.2012).

4.3 The Features and Additions in HTML5

“HTML5 Changelogs” introduces new features that are added to HTML5 .To meet today’s requirements, various new elements are added to achieve a better structure, a better media content and a better form handling.

TABLE 1. Some New Inline Elements (Refsnes Data 1999-2012a, date of retrieval 16.1.2012)

HTML5 Elements					
<article>	<aside>	<command>	<summary>	<figure>	<figcaption>
<header>	<footer>	<nav>	<section>	<wbr>	<audio>
<video>	<source>	<embed>	<track>	<canvas>	<svg>

<article>, <header>, <footer>, <nav>, <section>, <time>, <progress> are included for better structure. Elements like <audio>, <video>, <source>, <embed>, <track> provide a new standard for media content and <canvas> offers ways to create graphics with the script. Html5 provides more form elements and different new values for the element's type attribute such as tel, search, URL email, date-time, time, color, and placeholder.

The <video> element combined with <source> tags and control attributes like play, pause, volume, preload, height, width provide ways to embed a movie/video in a web page without using a plug-in from third parties. Similarly, <audio> element specifies a standard way to embed audio file in a webpage without using third party plug-ins like flash. HTML5 has provided <canvas> element, as a container to be used with Scripting (Usually JavaScript) to draw graphics on the fly. Canvas cannot draw anything by itself unless a built-in object is available, "getContext("2d")" object, which can invoke methods to draw different shapes. (Refsnes Data 1999-2012a, date of retrieval 16.1.2012).

Not everything used in the HTML document may be a first class HTML5 citizen. Some may be W3C specifications, such as SVG. Some may be APIs in the WHATWG HTML specification, but they may not be in the W3C standard. It will be hard to say what the standard will exactly look like. Some APIs are neither part of the WHATWG HTML specification nor the W3C HTML5 specification. They rather have their own specification.

TABLE 2. Some APIs in HTML5 (Wikipedia 2011, date of retrieval 16.1.2012)

WHATWG Specification	**Own API**
Drag and Drop API	Web SQL Database API (Depreciated)
Offline Web Application API	Indexed Database API
Document Editing API	File API
History API	Directories and System API
MIME-type API	File Writer API

4.4 The Features Lacking in HTML5

HTML5 is not a silver-bullet that many people are expecting it to be. HTML5 is an initiative to create a standard, for a consistent deployment among the devices. The implementation degree of the HTML5 features varies from not implemented to fully implement. In the near future, it is very less likely that HTML5 will be a high performance environment to run a heavy audio/video processing. HTML5 is not ideal for heavy database queries or heavy 2D/3D graphics where it needs an extreme CPU or GPU processing. It still assumes a connected environment, and many apps and “Offline APIs”, which HTML5 provides are not ready to work smoothly on multiple platforms. The other important issues are App discovery and Monetization. The app provider stores do not support web applications. The reason is that they generate revenue selling the native apps from their stores. Hence it is hard to find the web-apps. In addition, the third party developers have to rely on an advertisement for the revenue whereas in the store they can get the revenue according to the number of application downloads. (Closs 2011, date of retrieval 1.2.2012)

The gap between HTML-based apps and native apps is huge and it is not closing anytime soon. Developers are still developing native apps and big players like Apple, Google and Nokia are supporting them. In fact, very few, if any, are going to an HTML5-only way. (Siegler 2011, date of retrieval 1.2.2012).

“HTML5 offers the promise of write once run anywhere, but really it is a fallacy. To get the most out of HTML5 you need to write once, customize everywhere.” (thebeebs 2011, date of retrieval 1.2.2012).

5 MULTIMEDIA

“Multimedia comes in different formats. It can be almost anything you can hear or see. Examples: Pictures, music, sound, videos, records, films, animations and more. Modern Web Pages have often embedded multimedia elements, and modern browsers have support for various multimedia formats.” (Refsnes Data 1999-2012b, date of retrieval 8.2.2012)

Before the evolution of HTML5, text and image were native citizens of the markup but audio and video streaming had to rely on a third party plug-in to deliver multimedia contents. The situation was even worse before, when the Internet speed was slow and people could not imagine an audio and video streaming through the web. A Media player was the only choice. Major tech-giants Microsoft, Apple, Real Networks, and Macromedia had their proprietary players. Microsoft shipped a Media Player with Windows. Apple’s QuickTime was used as a Mac-only software, later released for Windows. Real networks first introduced its Real Audio Player with a .Ra and .Ram format and later allowed the video streaming using the H.263 video format. Eventually, these audio and video formats were bundled together under the name Real Player which was delivered as an option in Windows 98. Macromedia had its Shockwave Flash Player using an SWF format to exchange data, audio and video. It also supported simple animations. Macromedia was later acquired by Adobe and this technology was renamed Adobe Flash which later became the de facto standard for browser plug-ins with more than three fourth of market share among the plug-in.

Flash made the static web pages dynamic and it provided a more interactive web, with a better multimedia experience. Microsoft and Apple also have their plug-ins Silverlight and QuickTime. Vendors had their way of delivering multimedia but there was no standard. Users needed to have the appropriate players and plug-ins installed on their machine. The plug-ins needed constant updating and the existence of different vender specific plug-ins could conflict with other plug-ins creating instability and sometimes crashing the player. Another known issue was security, since plug-ins run on the client side. It was an easy target of malware. Vendors provided the remedy as next-release but it was quite a hassle for end users to wait for latest versions of plug-ins and upgrade them frequently. (Devlin 2012, 24-28).

5.1 Video

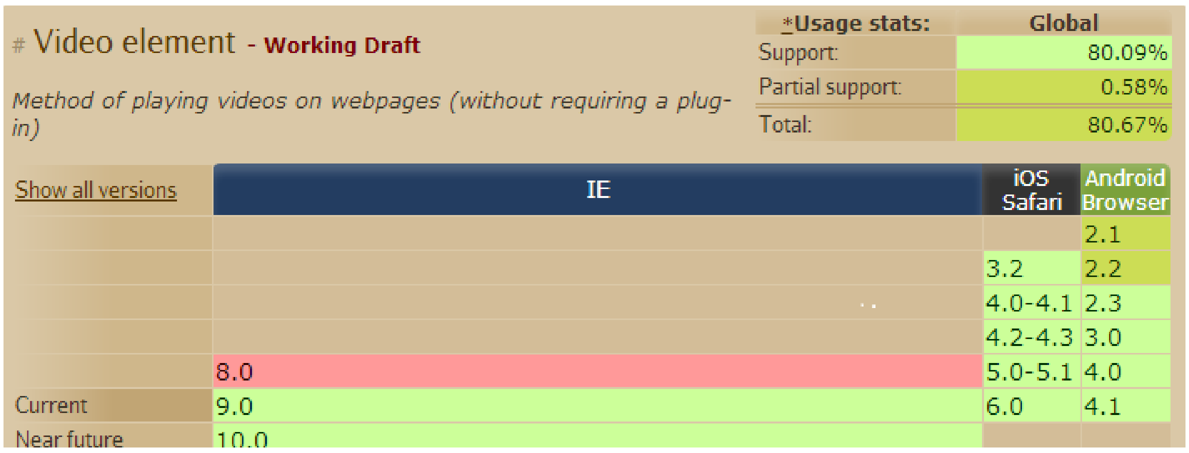
HTML5 provides a standard way to embed a video/movie within the web page with <video> element. The Video element is stable, but still evolving. Different attributes and functionalities are constantly added and upgraded.

A very minimum absolute version of embedding a video element inside a webpage instance could be:

```
<video src = “video.webm”></video>
```

The Video element supports a global attribute and has its own optional attributes. (Refsnes Data 1999-2012c, date of retrieval 8.2.2012).

Almost all the major desktop and mobile browsers support the video element but there is no single format of video that all browsers support. To be a standard, the HTML5 Working Group considered that an ideal format should have extremely high compression, the best image quality with a less decode processor use. It should not hold any patents and be a royalty free existence of hardware decoder, also supported by all User Agents. Initially Ogg Theora was recommended and later made a standard that should be supported by User Agents along with Theora video and Vorbis audio. Although there are no known patents regarding an Ogg format, companies like Apple and Nokia are concerned about the number of unknown patents waiting for companies with an extensive resource to use before they can sue. Eventually, Ogg was removed from the specification without any agreement of the standard format. Multiple sources should be provided from which the browser could select the format it support. Browser sniffing is an alternative but it is an error prone. The optional attribute “src” could be replaced with a source element <source>. The “type” attribute specifies the MIME type and optionally some required codecs. (Wikipedia 2012, date of retrieval 10.2.2012)



Video element - Working Draft

Method of playing videos on webpages (without requiring a plug-in)

*Usage stats:	Global
Support:	80.09%
Partial support:	0.58%
Total:	80.67%

Show all versions	IE	iOS Safari	Android Browser
			2.1
		3.2	2.2
		4.0-4.1	2.3
		4.2-4.3	3.0
	8.0	5.0-5.1	4.0
Current	9.0	6.0	4.1
Near future	10.0		

FIGURE 1. The screenshot of support table for HTML5 Video Element (@Fyrd 2012a, date of retrieval 2.11.2012)

5.1.1 Video Container Format and Codecs

To deal with multiple video formats the "type" attribute should contain the MIME type "video/" followed by the container type and the list of codecs. The container must support the codec used because not all video streams are compatible with all containers. There are many video containers available, three major and recommended containers are MP4, Ogg, and WebM.

MP4: MP4 is based on the Quick Time container (.mov). It is usually with mp4/m4v extensions. MP4 accepts H.264 as a video and AAC as an audio codec

Ogg: Ogg is an open source media container managed by Xiph, org. It does not have any known patents and it can be freely used for commercial and non-commercial purposes. It accepts Theora as a video and Vorbis as an Audio codec.

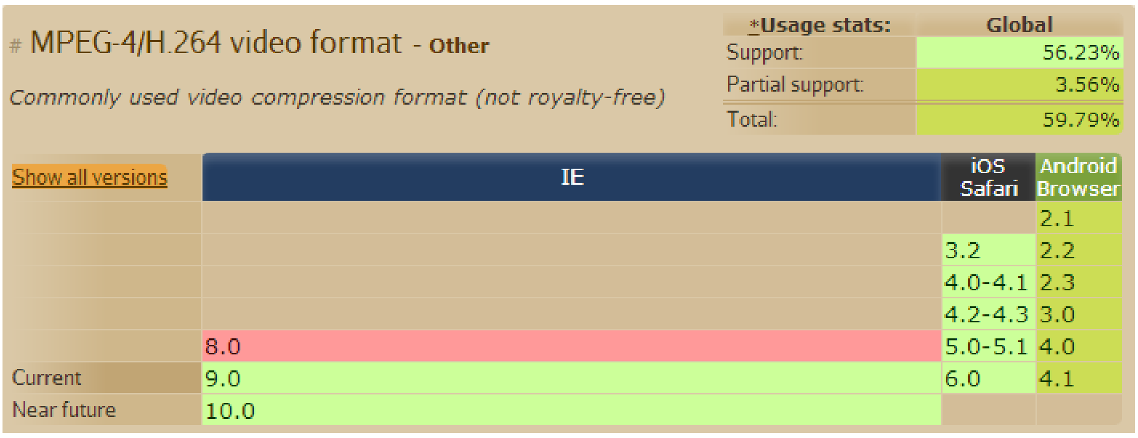
WebM: WebM is an open source container, which accepts a VP8 video codec, developed by Google based on a similar technique by ON2 named Matroska. It accepts Vorbis as Audio codec. The flash container could be used as a fallback option when dealing with desktop browsers. Microsoft also provides its Audio Video Interleave (AVI) proprietary container. It is not compatible with most of the video and audio codec, but it is still used.

Video Codecs:

The above-mentioned containers provide information only about how the audio and video tracks are stored. They do not say what to do next. Video codecs are responsible for decoding the video stream and display images serially. Audio CODECS decodes the audio stream and provide the input to the speakers.

H.264

H.264 was developed by MPEG group to provide a single codec from low processing devices like mobile phones to desktop computers. It provides different profiles so that devices with different processing power can decode accordingly. It is a patented technology, licensed through MPEGLA group. It can be embedded into containers like MP4 and MKV.



MPEG-4/H.264 video format - Other

Commonly used video compression format (not royalty-free)

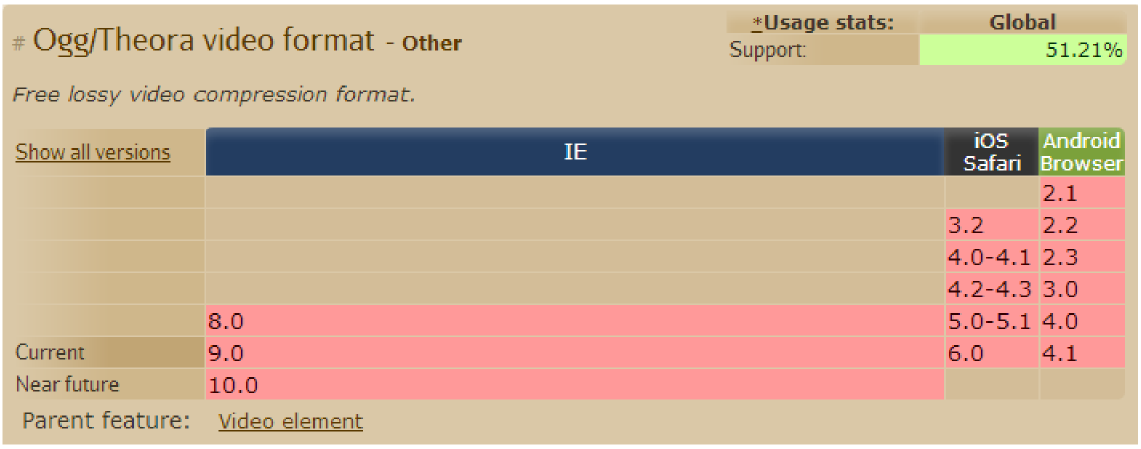
*Usage stats:	Global
Support:	56.23%
Partial support:	3.56%
Total:	59.79%

Show all versions	IE	iOS Safari	Android Browser
			2.1
		3.2	2.2
		4.0-4.1	2.3
		4.2-4.3	3.0
	8.0	5.0-5.1	4.0
Current	9.0	6.0	4.1
Near future	10.0		

FIGURE 2. The screenshot of a support table for H.264 codec (@Fyrd 2012a, date of retrieval 2.11.2012

Theora

Theora evolved from a VP3 codec is an open source and is not subject to any known patents. It is maintained by xiph.org Foundation. It can be embedded on any container but it is mostly used in the Ogg container.


# Ogg/Theora video format - Other

*Usage stats: Global
Support: 51.21%

Free lossy video compression format.

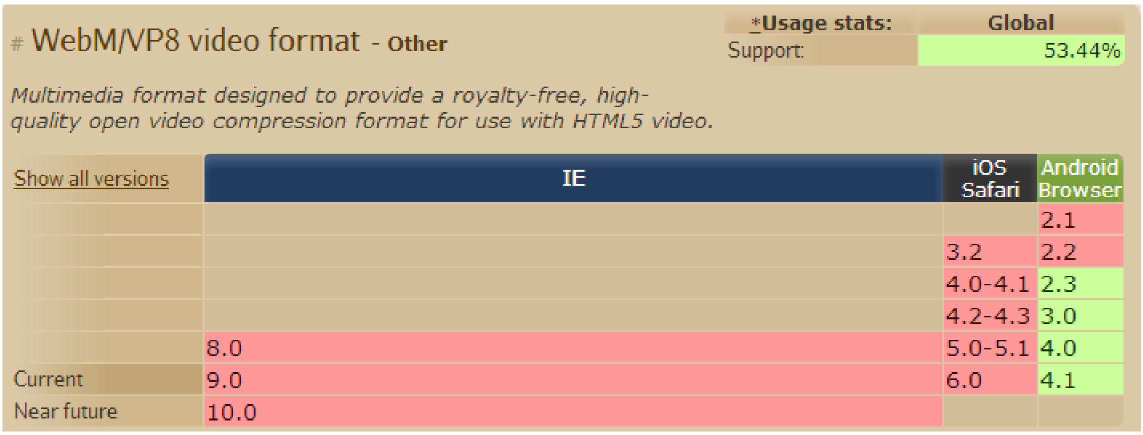
Show all versions	IE	iOS Safari	Android Browser
			🟥 2.1
		🟥 3.2	🟥 2.2
		🟥 4.0-4.1	🟥 2.3
		🟥 4.2-4.3	🟥 3.0
	🟥 8.0	🟥 5.0-5.1	🟥 4.0
Current	🟥 9.0	🟥 6.0	🟥 4.1
Near future	🟥 10.0		

Parent feature: Video element

FIGURE 3. The screenshot of a support table for THEORA (@Fyrd 2012a, date of retrieval 2.11.2012

VP8

VP8 was developed by ON2. VP8 was released as an open source after Google acquired ON2, and it is often compared with H.264 in quality and easy decoding technique. It is embedded in WebM container. Modern videos contain a video track and an audio track, which are interrelated. The marker inside the audio track makes the synchronization with a video possible.


# WebM/VP8 video format - Other

*Usage stats: Global
Support: 53.44%

Multimedia format designed to provide a royalty-free, high-quality open video compression format for use with HTML5 video.

Show all versions	IE	iOS Safari	Android Browser
			🟥 2.1
		🟥 3.2	🟥 2.2
		🟥 4.0-4.1	🟩 2.3
		🟥 4.2-4.3	🟩 3.0
	🟥 8.0	🟥 5.0-5.1	🟩 4.0
Current	🟥 9.0	🟥 6.0	🟩 4.1
Near future	🟥 10.0		

FIGURE 4. The screenshot of a support table for VP8 (@Fyrd 2012a, date of retrieval 2.11.2012)

(Pilgrim 2010, 81-88).

5.1.2 Encoding Video Files

A modern desktop and a mobile browser support at least one of the above mentioned codec. Thus, video files should be encoded in any one of these formats before using in the application. There are plenty of encoders to convert a video from one format to another. Some encoders are

Micro Video Converter:

It runs on both Windows and Mac. It not only converts a video format to Theora, WebM and MP4. It can also convert the audio formats.

Handbrake:

Handbrake is open source and capable of encoding a video to a Theora and MP4 format on Windows, Linux and Mac.

For desktop applications, Media Converter could be a viable option since it is an online converter capable of encoding a video to Theora, MP4 and Flash FLV with or without downloading the converter.

(Devlin 2012, 65-66).

5.1.3 Implementation

HTML5 provides two ways to include the video in applications using <video> element. The Video element has either a single source or multiple sources. In case of a single source, including the video element is fairly easy. It could be done like including an image. To play a WebM file

```
<video src = "video.webm" ></video>
```

It is a good idea to include two optional attributes the height and width of the video element. It could be same as the height and width used in encoding the video. The control of the video could be provided manually using HTML, CSS and JavaScript or the browser could be given the information to provide built-in controls

<video src = “video.webm” width = “320” height = “240” controls></video>

HTML5 provides an optional preload attribute, which starts to download a video implicitly as soon as the application starts. If otherwise stated, the preload could have three values : auto, metadata and none. If the value is “auto”, the video starts to download when the page/application loads. If the value is “metadata”, only metadata is loaded. If the value is “none”, the page/application does not start downloading the video when the application/page loads. In network critical devices, like mobile and tablets where the user might have to pay for every byte they use, it is not a good choice.

The optional attribute “autoplay” provides a way to start downloading or play after the page loads. It can be used with or without providing a value. For video streaming sites like YouTube, it might be great but similar to the preload attribute. It is not a smart move for mobile device with limited network bandwidth and the user might get annoyed with this feature if misused. If “autoplay” is selected, it will override the “preload” attribute since it is necessary to download to play. The “loop” attribute allows the video to play repeatedly and the “poster" attribute provides the image that the user can see when the video is not playing. If the “autoplay” is false, by default it shows the first frame of the video as a poster.

<video src =“video.webm” height = “320” width =“240” controls autoplay loop>

A Video can be included from multiple sources in HTML5 using a source element. The <video> element can include and can contain a multiple <source> element. The browser will then go through each source and play the first possible video. The type attribute in <source> element holds the “MIME” type of the particular container so that the browsers do not have to load each video. It helps to reduce network bandwidth significantly. The optional attributes could be used within the video element in a similar way as a single source

```
<video height = “320” width= “240” controls preload poster = “pic.jpeg” loop>
<source src = “video.mp4” type = “video/mp4””>
<source src = “video.webm” type = “video/webm””>
<source src = “video.ogv” type = “video/ogg””>
</video>
```

Including a codec in the type attribute can be helpful for browsers to decide if they can play the video or not. Using the codec in the type attribute, needs certainty. The format of the string should be accurate, including the quotes used. Otherwise, the browser will not recognize the source.

```
<video height = “320” width= “240” controls preload poster = “pic.jpeg” loop>
<source src=“video.mp4” type=‘video/mp4; codecs= “avc1.42E01E, mp4a.40.2” ’>
<source src = “video.webm” type = ‘video/webm; codecs= “vp8, vorbis” ’>
<source src = “video.ogv” type = ‘video/ogg; codecs= “Theora, Vorbis” ’>
</video>
```

(Pilgrim 2010, 110-112).

Most modern browsers support the video element, to target legacy browsers who do not support the video. It is also a viable option to provide flash as a fallback option and a download option with a caption that your browser could not play the video, so here is the link to download.

5.2 Audio

HTML5 also provides a similar way like video to embed an audio in the application using a native <*audio*> element. Most modern desktops and mobile browsers support audio element. There would be no need of using the third party plug-ins. Major audio codecs supported are:

5.2.1 Audio Codecs

Vorbis

Vorbis is an audio codec mostly used in Ogg container and WebM recently. It is an open source and popular among gaming communities and companies. It can compress the audio file to a small size yet maintaining its quality. It is ideal when streaming across over the web because it saves network bandwidth. It uses “application/ogg” MIME type and “audio/ogg” codec.

MP3

Media Player 3 (MP3) is a patented codec developed by MPEG, which only specifies how to decode the format with no specific encoder. It is one of the most used formats. It has different sound quality due to different encoders used; MP3 uses “audio/mpeg” MIME type and “audio/mp3” codec

AAC

Apple uses an advanced Audio codec for the iTunes store. It was conceived as a successor of the MP3, thus AAC has a better sound quality than MP3. It uses “audio/aac” MIME type and “audio/aac” codec.

MP4

Media Player 4 (MP4) is also used as container (more about the container in the video section). But it can only be used for audio encoding. It uses “audio/mp4” as MIME type and “audio/mp4” as codecs. Audio Element supports all the global attributes and also has its own optional elements.

5.2.2 Encoding Audio Files

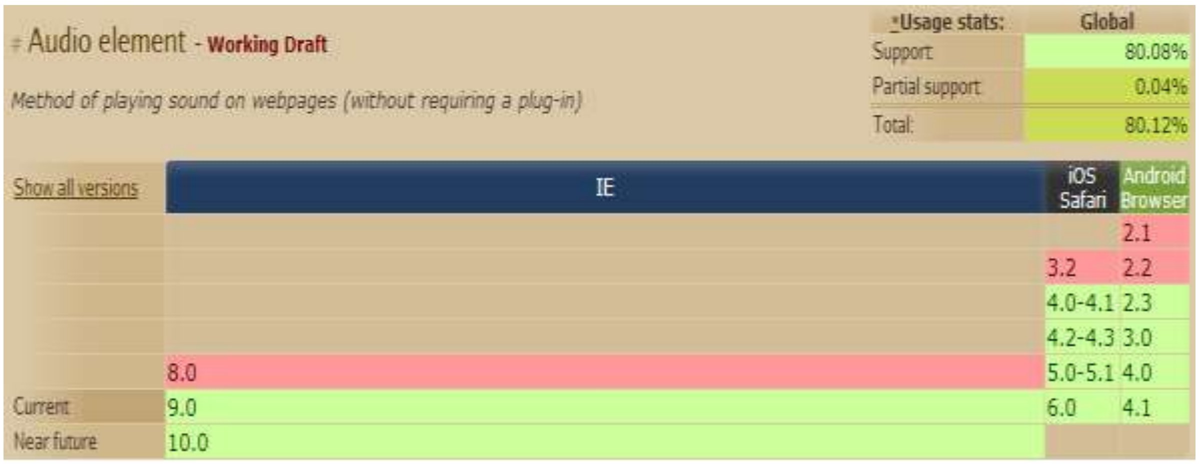
Different browsers support different audio formats. There are audio encoders which convert the audio file from one format to another.

Micro Video Converter

Micro Video Converters with a drag and drop interface allows files to change to a different format like Ogg, WAV, and Mp3.

Media Converter

It is an online conversion application, which provides a way to change the audio file format to MP3, WAV and Ogg. For legacy browsers who do not support a native audio element, the fallback option to flash, QuickTime or other third party could be provided. The download option could be another viable way for the user who does not have both native support and plug-ins.



Audio element - **Working Draft**

Method of playing sound on webpages (without requiring a plug-in)

*Usage stats:	Global
Support	80.08%
Partial support	0.04%
Total:	80.12%

Show all versions	IE	iOS Safari	Android Browser
			2.1
		3.2	2.2
		4.0-4.1	2.3
		4.2-4.3	3.0
	8.0	5.0-5.1	4.0
Current	9.0	6.0	4.1
Near future	10.0		

FIGURE 5. The screenshot of support table for HTML5 Audio Element (@Fyrd 2012b, date of retrieval 2.11.2012)

5.2.3 Implementation

The implementation is fairly easy, for example if the audio file is only in a single format like Ogg Vorbis, then it can be included like

<*audio src*= *“audio.ogg”*> </*audio*>

Then, there are optional attributes like controls, autoplay, loop and preload. Similar to that of the video element, the control attributes provide the control mechanism explicitly. Otherwise the browser uses its built-in control technique, and the auto-play downloads the file and starts to play as soon as the application/webpage starts to load. The loop makes the audio play repeatedly, “preload” only downloads the video and keeps at the ready state to play the format. It might be a great option for audio sharing sites but in other cases, it might be loss of resources or it would be unwanted. Playing an audio file with different source is also similar to that video, It is recommended to use at least mp3 and ogg format, more sources are optional.

<*audio control autoplay loop*>

```
<source src = “audio.ogg” type = “audio/ogg” >
<source src = “audio.mp3” type = “audio/mp3””>
<source src = “audio.wav” type = “audio/wav” >
</audio>
```

(Devlin 2012, 44 – 59).

6 STORAGE

“When web developers think of storing anything about the user, they immediately think of uploading to the server” (HTML5ROCKS 2012b, date of retrieval 14.2.2012)

HTML5 changes that attitude by providing two offline capabilities: Application Caching and Offline Storage. Application caching saves logic whereas Offline storage stores the data. Application Caching uses caching and Offline storage uses client side data storing to serve some common purposes. To make the app work even if there is no network connection, app caching and storage data boosts performance because it reduces the network latency and data could be cached or stored without the need of a dedicated server. (Mahemoff, M. 2011, date of retrieval 14.2.2012)

6.1 Offline Storage

Offline Storage before HTML5 Era

Prior to HTML5 offline storage, some technologies were invented to serve a persistent storage on the client side. Cookies are small software created with the intent to hold a small amount of data on the client side mostly identifying information and the rest of the data on the server. (Mahemoff, M. 2011, date of retrieval 14.2.2012)

Cookies slow down the application as they are transmitted with every HTTP request, they pose threats, and they are mostly sent unencrypted. The size of cookies is about 4KB, which is enough to slow down the speed, but its use does not match its cost. The need was a large storage space on the client’s side, which remains even after the page is refreshed or the connection is lost. The specific and plug-in based hacks of the browsers have been applied to increase the storage space. One of the noticeable browser specific behavior was Microsoft's' “user Data” shipped with IE 5.0. Since it was browser specific, supported with only IE and limited size of 64KB, it could not be widely adopted. The different plug-ins for Google gears, Adobe flash and Java were introduced for storing data. They were somewhat successful, but there was no uniformity. Brad Neuberg and others had to try to hack dojox's storage to provide a unified interface to all the plug-ins. Despite such efforts there was no single storage mechanism on which all agreed. Therefore, different storage limitations and user experience existed. (Pilgrim 2010, 127-128).

Offline Storage in HTML5 Era

HTML5 provides different yet related APIs for the client-side data storage. The APIs are Web Storage API, Web SQL Database, Indexed Database and File Access or File API. These new APIs are more compatible and they are accepted by major browsers compared to plug-ins based or browser specific storage hacks. Most modern desktop and mobile browsers support offline storage. In case of legacy browsers, older technology could be useful. (Mahemoff 2011, date of retrieval 14.2.2012)

All the APIs have their specific features but they still serve some common features. The data passed to the browser's storage is saved locally on the client side. It allows search, retrieval and sometimes batch manipulation. All the APIs provide sandboxed data. The browser allows saving, manipulation and retrieval if, and only if, the protocol domain and port match with each other. Otherwise, the request is disqualified. The storage space is an issue that browsers are enforcing separately. At present, browsers are allowed to store 5MB data in Web Storage and 25MB in Web SQL. The Indexed database is still in the working draft and yet to be implemented by any major browsers. Transactions are supported in both database formats like any relational database and “race conditions” are restricted. It is a phenomenon of operation in the database parallel where one operation could affect the database in such a way that, the state is unknown and results are unpredictable and corrupted. Most storage formats support synchronous and asynchronous modes. The former is a way of doing one operation one at a time, which blocks all other operations, whereas the later allows a parallel operation at the same time. Web Workers is still in its working draft. It could be used alternatively to perform operations in separate threads, which could be expensive or not compatible with all browsers. Thus, it should be checked before use. (Mahemoff, M. 2010, date of retrieval 14.2.2012)

6.1.1 Web Storage

Web Storage was once a part of HTML5 specification and later it split into its own separate API. HTML5 provides two objects, local storage and session storage, for storing data on the client browser. It uses JavaScript to store and access the data. (Refsnes Data 1999-2012d, date of retrieval 16.1.2012)

Local Storage

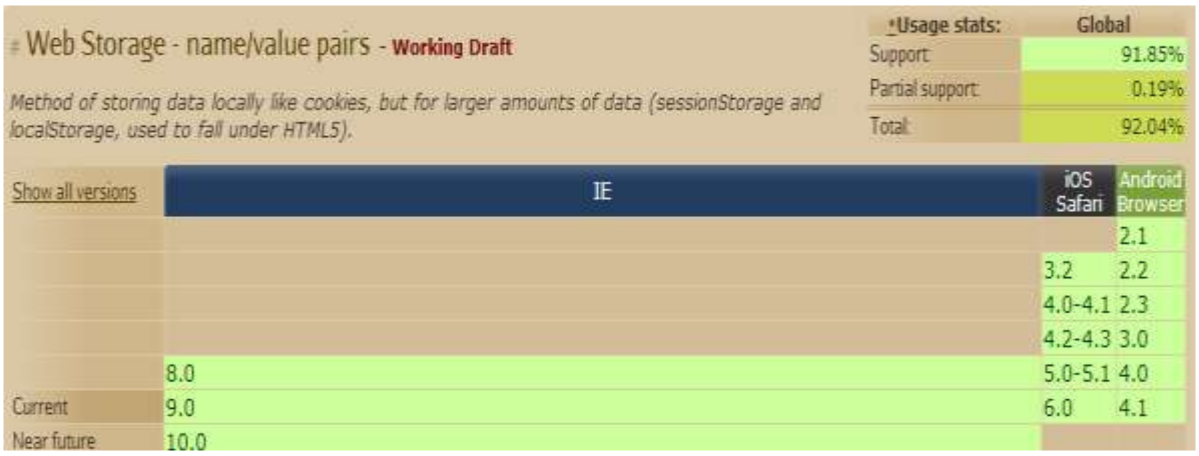
It is a way for applications to store key-value pairs in the client's browser. The specification indicates that the value can be of any type, is a fallacy. The values should be serialized (converted to strings) before storing and parsed before accessing. JSON API can be used to "stringify" and parse to store and access respectively. (Mahemoff, M. 2011, date of retrieval 14.2.2012).

Similar to cookies, this data persists even if the application is closed. Unlike cookies, they do not transmit to a server with HTTP requests (unless a user wants them to do it). The data stored is not available for a cross browser usage. Using "Local Storage" the data can be accessed in an offline mode and stored on the server when there is a network connectivity. It can be a performance booster by minimizing the network requests. (Kappart, L. 2011, date of retrieval 15.2.2012)

Session Storage

It is a variant of Local Storage. In Session Storage, the data stored on the browser persists to the current session only. Once the user exits from the current browsing tab, the data expires. It is suitable for independent browsing, where data from one tab cannot be transmitted to another tab. It is safer than a local storage. Both Local and Session storage are not by any means used to store valuable data. The data stored in a local storage should be manually removed. The location of data depends on the browser or manual configuration. (Sheridan, M. 2011, date of retrieval 15.2.2012).

All the major modern desktop and mobile browsers natively support web Storage.



Web Storage - name/value pairs - **Working Draft**

Method of storing data locally like cookies, but for larger amounts of data (sessionStorage and localStorage, used to fall under HTML5).

*Usage stats:	Global
Support	91.85%
Partial support	0.19%
Total	92.04%

Show all versions	IE	iOS Safari	Android Browser
			2.1
		3.2	2.2
		4.0-4.1	2.3
		4.2-4.3	3.0
	8.0	5.0-5.1	4.0
Current	9.0	6.0	4.1
Near future	10.0		

FIGURE 6. The screenshot of a support table for HTML5 Web Storage (@Fyrd 2012c, date of retrieval 2.11.2012)

Implementation:

Both local storage and session storage are objects of type Storage. Session storage attributes returns the value of type Storage, associated with the documents assigned in the session storage area. Each storage object implements Storage interface

```
Interface Storage {

        readonly attribute unsigned long length;
        DOMString key (unsigned long index);
        Getter DOMString getItem(DOMString key);
        Setter creator void setItem(DOMString key, DOMString value);
        delete void removeItem(DOMString key);
        void clear ();
    };
```

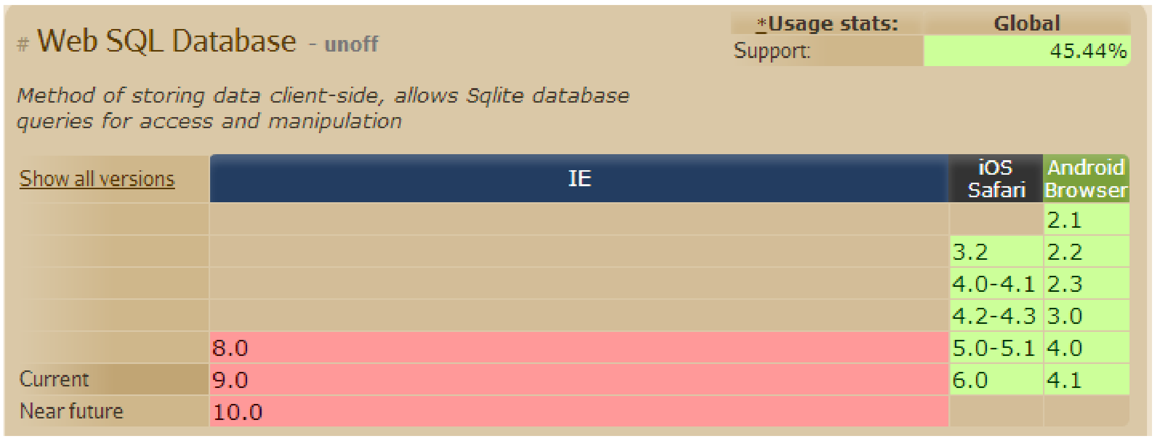
The "length" attribute returns the number of key/value pairs present in the list. The key method returns name of the key in nth index. It provides a getter method to get the values associated with the key and returns null if there is no value associated with the key. The setter method allows setting the key and valuing the pair. The remove method allows the removal of individual key/value pair and the clear method allows removing all key/value pairs in the list. (W3C 2011b, date of retrieval 15.2.2012).

Web Storage fires “storage” event every time the methods in Storage interface are used and some data are changed. The event can be accessed by a global “Window” object using “addEventListener()” method. If the browser does not support the method, new event handlers could be registered. (Pilgrim 2010, 131).

There are some downsides of Web Storage. It is not a database and does not have transactions. The size limit is low, so it is not good for larger data. It also has issues of data integrity and security. (Mahemoff, M. 2011, date of retrieval 12.2.2012).

6.1.2 Web SQL

HTML5’s Web SQL Database provides options to create and access database client’s side using JavaScript. It is based on SQLite with a relational structure, allowing querying and manipulating using joins. It supports an asynchronous and synchronous mode of querying. It does not have a 5MB storage limitation as Web Storage. The Web SQL database is not agile and deprecated from HTML5 specification. Indexed Database possibly replaces Web SQL. (Mahemoff, M. 2011, date of retrieval 12.2.2012)



Web SQL Database - unoff

*Usage stats:	Global
Support:	45.44%

Method of storing data client-side, allows Sqlite database queries for access and manipulation

Show all versions	IE	iOS Safari	Android Browser
			2.1
		3.2	2.2
		4.0-4.1	2.3
		4.2-4.3	3.0
	8.0	5.0-5.1	4.0
Current	9.0	6.0	4.1
Near future	10.0		

FIGURE 7. The screenshot of support table for HTML5 Web SQL Database (@Fyrd 2012d, date of retrieval 2.11.2012).

It is depreciated technology, but currently supported by many browsers, it seems IndexedDB is the future of databases.

Implementation

To open a database or to create one if the database does not exist, the “openDatabase()” method should be used with the name of a database, the version of the database, the display name and an estimated size in bytes as the argument. The optional callback of type Database Callback interface could be invoked if the database has not been yet created.

```
var db = window.openDatabase(“Database Name”, “version”, “Description”, “size”);
```

The default size of database is 5MB. The user can be prompted if an additional amount is needed. The empty parameter in a version means any version is accepted and “INVALID_STATE_ERR” exception is thrown if the database version mismatches. The change “version()” method allows an asynchronous verification of the version number and changes it during schema updating. (W3C 2011c, date of retrieval 5.3.2012).

The transactions are the wrappers of SQL statements, which allow multiple SQL statements within the transaction. It has a unique “rollback" property which enables it to prevent updating the database if one or more SQL statements within the transaction fail. The transaction also contains optional callbacks for success and error.

```
db.transaction (function (tx) {
                // perform the SQL activity
}, errorCallback, successCallback);
```

It is recommended to use read Transaction, if the intention is only to read from the database. If the intention is to read and write, then the read write transaction should be used. The above-mentioned code represents the read-write transaction

```
db.readTransaction (function (tx) {
        // this is read transaction
}, errorCallback, successCallback)
```

The read write transaction blocks the UI. In case of a synchronous processing, it is recommended to use Web Workers.

ExecuteSQL is used to execute the actual SQL query once the transaction object is available. This method is able to perform as queried, for instance creating a table, inserting a row and deleting partial or complete data.

tx.executeSQL (SQLStatement, optional arguments, optional successCallback, optional errorCallback);

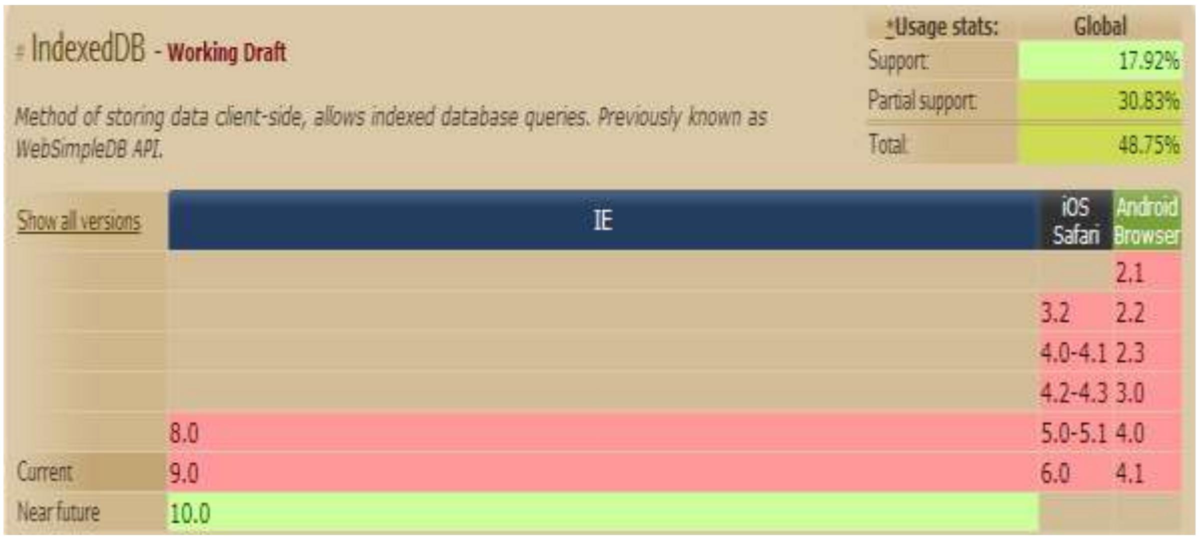
The "tx" is the object of the transaction, used to invoke the method. The SQL statements are standard SQL statements as strings with a possibility of arguments. The "callback" is of type "SQLStatementCallback" and "errorCallback" is of type "SQLStatementErrorCallback"

tx.executeSQL ('CREATE TABLE documented (id, name)');

(Sharp 2010, date of retrieval 5.3.2012).

6.1.3 IndexedDB

Indexed Database also known as IndexdedDB, is a hybrid database with an attempt to combine the strength of Web Storage and Web SQL and leave out their weaknesses. Indexed DB uses JavaScript object store with indexes to perform the queries and to iterate over indexes for an instant searching of data. Unlike relational databases, IndexedDB is a NOSQL database, which is more agile, and no schema should be defined up front. In this way, it is similar to Web Storage with the addition that IndexedDB can have multiple data stores / databases. It provides Asynchronous APIs that improve the performance so that one action does not affect others. Synchronous APIs perform one action at a time.



IndexedDB - Working Draft

Method of storing data client-side, allows indexed database queries. Previously known as WebSimpleDB API.

*Usage stats:	Global
Support:	17.92%
Partial support:	30.83%
Total:	48.75%

Show all versions	IE	iOS Safari	Android Browser
			2.1
		3.2	2.2
		4.0-4.1	2.3
		4.2-4.3	3.0
	8.0	5.0-5.1	4.0
Current	9.0	6.0	4.1
Near future	10.0		

FIGURE 8. The screenshot of a support table for IndexedDB (@Fyrd 2012e, date of retrieval 2.11.2012)

The support for Indexed Database could also be checked programmatically

```
        if (window.indexedDB) {
                // do something
        } else {
                //use third party solution
        }
```

Modernizr could be another option

```
        if (Modernizr.indexedDB) {
                // do something
        } else {
                // use third party solutions
        }
```

Implementation

Creating an Object Store

```
var db = window.indexedDB.open(“database”, “Personal Database”);
if (db.version != “1”) {
            db.createObjectStore (“PersonalDB”,
             // create a new object store
            “id”, //key path
            true); //auto-increment
db.setVersion (“1”);
} else {
            // database is already initialized
}
```

Keypath must be name of an enumerated property of all objects being stored in the Object Store DB versioning is optional.

Storing data in the Object Store

```
var store = db.openObjectStore (“PersonalDB”);
var data = store. put(name: “Ryan”, gender: “Male”, hobby:" football”);
```

The “add()” and “put()” method both could be used for this purpose, using one after another method with the same ID might result into an overriding of data

Finding things in Object Store: There are two options to search for data by a key or by a query.

Retrieving by Key (indexes)
Create Index

```
db.createIndex (“PersonsName”, “Person”, false);
var index = db.openIndex (“PersonsName”);
var id = index.get(“Ryan”);
```

Querying (cursors): To Restrict the name from A to Z

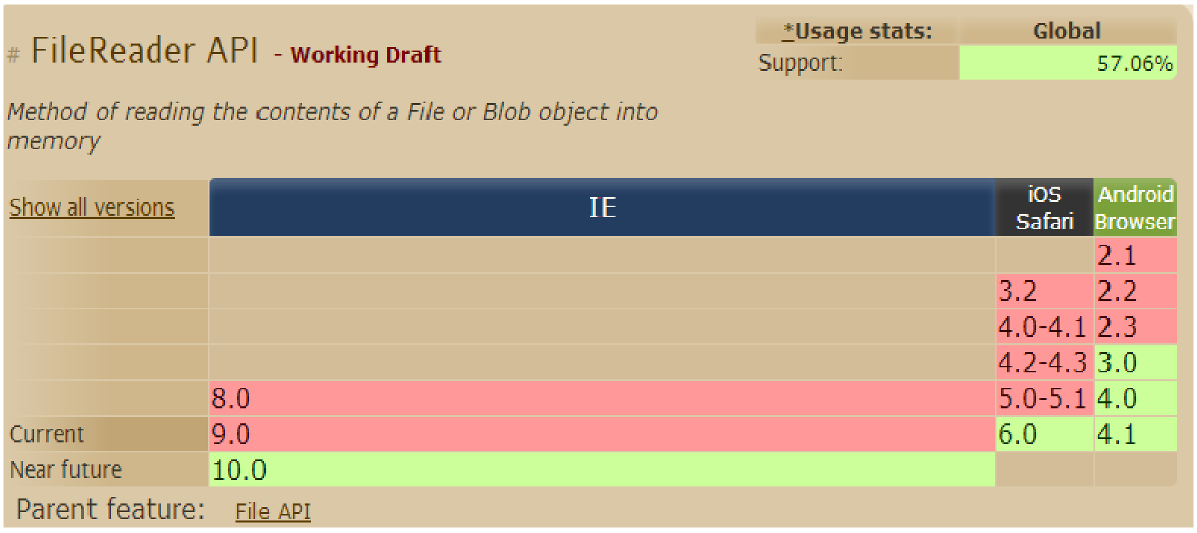
```
var range = new KeyRange.bound('A', 'Z');
var cursor = index.openObjectCursor(range);
While (cursor. continue ()) {
//continue to do something
}
```

Using IndexedDB, JavaScript could be used to perform all the activities without need of SQL statements. (Ranganathan & Wilsher 2010, date of retrieval 16.2.2012).

6.1.4 File System API / File API

HTML5 provides a way to interact with large files and binary data. File System allows to create, read, write and navigate files to the sandboxed section of the user's local file system. HTML5 provides separate APIs to serve different purposes. (Bidelman 2011a, date of retrieval 21.2.2012)

HTML5 has specified three different APIs File API, File API Writer and File API: Directories and system File API for reading and manipulating files include File interface, Blob interface, File List interface, FileReader interface and URIScheme.



FileReader API - Working Draft

*Usage stats:	Global
Support:	57.06%

Method of reading the contents of a File or Blob object into memory

Show all versions	IE	iOS Safari	Android Browser
			2.1
		3.2	2.2
		4.0-4.1	2.3
		4.2-4.3	3.0
	8.0	5.0-5.1	4.0
Current	9.0	6.0	4.1
Near future	10.0		

Parent feature: File API

FIGURE 9. The screenshot of a support table for File Reader API (@Fyrd 2012f, date of retrieval 2.11.2012)

File and Blob

File API introduces separate interfaces representing Blob and File. Blob represents immutable raw data and provides a method to slice the raw data into chunks of raw data. It also provides a size attribute to represent the size of the chunk of raw data to read both synchronously and asynchronously.

The constructor to create a new Blob object could take Array Buffer, Blob or DOMString as a parameter

var blobObject = new Blob ()

The slice method takes three optional parameters start, end, and content. It returns a new blob object ranging from an option start to an end parameter.

Blob slice (optional long start, optional long end, optional DOMString content Type);

The start parameter of type long takes a value for start point treated as a byte-order position, where the first byte represents the zero position. If the start is negative, the relative Start should be the sum of the start and size. The second parameter of type long is a value for the end. The third parameter is optional of type DOMString. It is used to set a content type header on the Blob object returned by the method, which could be empty. It also provides two read only attributes, size and type. They are the size and media type of Blob object. The File inherits from Blob and describes one file in the File List with read only attributes *"name"* and *"lastModifiedDate"* of type DOMString and Date respectively. The error exception should be thrown if the file requested does not exist or is modified in an unusual situation irrespective of a synchronous or asynchronous read. (W3C 2012d, date of retrieval 9.3.2012)

Retrieving File and Reading

FileList interface represents a list of files. The object of this type could be created to know the number of files. The getter method returns the file with the provided index of type long and returns null if no file exists in the list.

File item (unsigned long length) to read file FileReader Interface provides the constructor to create an object and different methods to read the File or Blob into the memory in different ways.

Creating constructor

```
var fileReader = new FileReader();
```

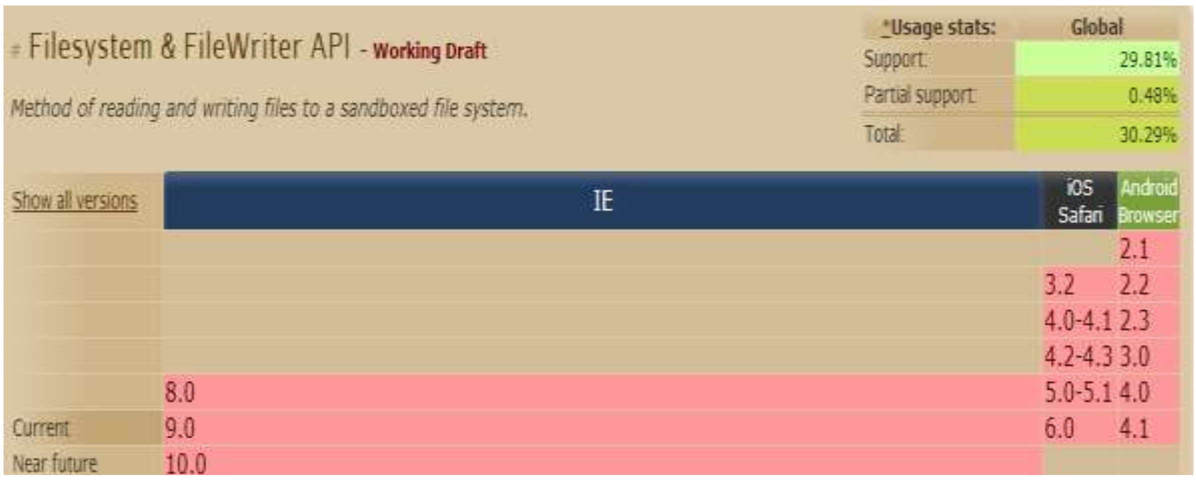
Methods to read Blob or File

```
void readAsArrayBuffer(Blob blob)
void readAsTextBlob(blob, optional DOMString encoding)
void readAsDataURL(Blob blob)
```

The reader object could be in three stated 0, 1, 2 of type short. The API also provides an event handler attribute and a type. They are responsible for firing the events asynchronously, which updates the result. It also provides a way to create an object of type "FileReaderSync" for the synchronous reading of Blob and File. (W3C 2012d, date of retrieval 9.3.2012)

File API: Writer

The API to write into files from web application files, includes BlobBuilder interface, FileSaver interface, FileWriter interface and FileWriterSync interface



Filesystem & FileWriter API - Working Draft

Method of reading and writing files to a sandboxed file system.

*Usage stats:	Global
Support:	29.81%
Partial support:	0.48%
Total:	30.29%

Show all versions	IE	iOS Safari	Android Browser
			2.1
		3.2	2.2
		4.0-4.1	2.3
		4.2-4.3	3.0
	8.0	5.0-5.1	4.0
Current	9.0	6.0	4.1
Near future	10.0		

FIGURE 10. The screenshot of a support table for File System and File Writer API (@Fyrd 2012g, date of retrieval 2.11.2012).

Appending data to Blob / File

BlobBuilder interface provides “append()” method with three different parameters of type “Blob”, “ArrayBuffer” and “DOMString” with optional “DOMString” type ending. (W3C 2012e, date of retrieval 10.3.2012).

Writing to a File

The API introduces FileWriter Interface with methods to write, seek and truncate and read-only attributes position and length of type long. The “fileWriter” is created by calling “createWriter()” method of the “FileEntry” interface from the object returned after successfully creating a file.

```
fileEntry.createWriter (Function (fileWriter) {
                }, errorHandler);
```

To write the provided data at the certain position
void write(Blob data)

Seek the position where the next write action should occur
void seek(long offset)

Change the length of the file
void truncate(unsigned long size)

To write into a file synchronously, the API provides FileWriterSync Interface that inherits from FileSaverSync. It is recommended to use a synchronous way with web workers so that it does not lock other activities and User Interface. “FileError” Interface is used to report error codes asynchronously and “FileException” to report exceptions synchronously.

Note: When an application is granted a write access, it does not necessarily mean that a read access is also provided. User agents could control the quota and interrupt or warn the user if the size exceeds the limit, disk space is a concern or bandwidth is reached. Creating risky names and opening unsafe files should be prohibited and MIME type of file should match. (W3C 2012e, date of retrieval 10.3.2012).

File API: Directories and System

It provides access to files and directories inside a sandboxed section in user's local system. This API targets to mitigate the use cases not served by a database to interact with large files, blobs or directories in the client-side file system. (W3C 2011f, date of retrieval 11.3.2012)

Opening FileSystem

The LocalFileSystem Interface in the API provides a method to access the File System. If the method is called for the first time, a new sandboxed space for a storage purpose is created. The application cannot then access another application's data as well as data from other storage areas such as a hard drive.

> *requestFileSystem (unsigned short type, unsigned long size, FileSystemCallback successCallback, optional ErrorCallback errorCallback);*

The type could be temporary and persistent represented by 0 and 1 respectively. User agents can remove the temporary data whereas users must be prompted to delete the persistent data. The size represents the storage size in bytes and "sucessCallback" is called if the call is a success and "errorCallback" is called if an error is encountered. "LocalFileSystemSync" Interface describes the method and properties in the synchronous opening of File System. (Bidelman 2011b, 11-12)

The method "requestFileSystemSync()" could be used to obtain an access synchronously which could then be used with web workers without locking the UI. (Bidelman 2011b, 53)

Working with Files and Directories

The FileEntry Interface represents files in the sandboxed area of FileSytem, which inherits Entry Interface. The entry interface exposes methods to get metadata about the entry. The methods are move, copy and remove. Various Boolean attributes like “isFile” and “isDirectory” and DOMString attributes “name”, “fullpath” and “filesystem” are provided for reading. (Bidelman 2011b, 15)

Creating a File

To create a file or look up if the file exists, “DirectoryEntryInterface” provides a method to be used with “FileSystem” object that is returned as successCallback.

fs.root.getFile (“filename.txt”, {create: true, exclusive: true})

It creates a file with a name, if there is no file with the same name and causes failure if the target already exists. (Bidelman 2011b, 16-17)

Importing Files

Files can be imported using input field, HTML5 drag and drop API, using XMLHTTPRQUEST to fetch remote binary data or to use a copy pasting technique. These ways to import files into a file system are savior for the security problems and unable to access beyond the sandboxed file system. (Bidelman 2011b, 20-21)

Removing Files

To remove the files from the File Entry, a FileEntry object should invoke the remove method with the first argument for successCallback and an optional errorCallback as the second argument.

fileEntry.remove (function ( ) { }, error);

(Bidelman 2011b, 28-29)

Reading Directory

To read entries from a directory, a directory reader should be created to invoke “readEntries()” method until all the entries have completed reading.

```
var dirReader = fs.root.createReader();
dirReader.readEntries (function (resultOfReading) { }, errorHandler);
```

(Bidelman 2011b, 34-35)

Removing Directory

To remove empty directories, the object of “DirectoryEntry” Interface could be used to invoke “remove()” method and to remove a directory with contents “removeRecursively()” could be used.

```
dirEntry.remove (function() { }, error);
dirEntry.removeRecusrively(function () { } , error);
```

(Bidelman 2011b, 36)

Copying a file and a directory could be done using the same “copyTo()” method. The copy of directory copies all its content, too. The object of FileEntry interface is required to invoke the method.

```
entry.copyTo (DirectoryEntry parent, optional DOMString newName, optional
EntryCallback successCallback, optional ErrorCallback errorCallback);
```

In the similar way, the entry could be moved to a different location in the file system. Moving the file or directory on top of an existing file or directory replaces the file or directory, if the move is successful.

```
entry.moveTo(DirectoryEntry parent, optional DOMString newName, optional
EntryCallback successCallback, optional ErrorCallback errorCallback).
```

The renaming could be done with the same method as moving providing a new optional name for the entry and making current working directory as the destination directory. (Bidelman 2011b, 37-41).

Similar to that of File API, a method is provided to create URL for the file system. The FileEntry object should invoke “toURL()” method to get the URL and “resolveLocalFileSystemURL()” with a URL, successCallback and errorCallback gives back the URL. (Bidelman 2011b, 43-45).

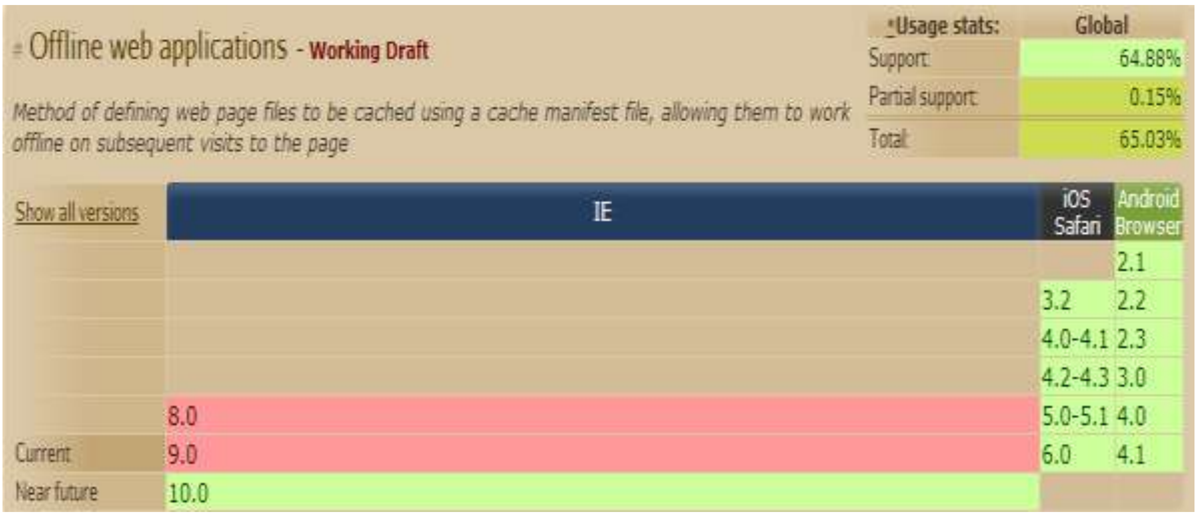
6.2 Offline Web Applications

It is a challenge to remain connected to a network every time in order to use web-based applications. This problem is slightly mitigated by browser caching, but counting on browser cache will not always work. The browser itself may replace or remove the cache for a storage reason or the user might accidentally clear the cache. (Rouget 2010, date of retrieval 24.2.2012).

To overcome this situation, HTML5 provides a mechanism known as Application Cache (App Cache). It provides the browser a manifest file when the user first starts the application in an online mode. The manifest file includes all the resources like HTML, CSS, JavaScript, images, style sheet, video, etc. cached and stored locally to be used later in an offline mode. (Flirtman 2010, 308)

In subsequent visits, the browser tries to download to see if the manifest file has changed. If it has not changed or if the network is not available then it will load the application and resource from Cache memory. Otherwise, the browser will download and store the manifest file again. To check the network state, DOM provides a flag and events are fired once the state is changed. Other actions such as creating or storing the data locally, depends on the developer. HTML5 App Cache can take the application offline by providing the Cache. (Pilgrim 2010, 137)

Implementing cache gives three major advantages. It gives the user a full access to the application even if there is no connectivity. Caching makes loading faster since all the resources are cached locally. It reduces the server-client interaction thus reducing the bandwidth. The decision to specify which resource or files should be cached depends on the developer (Bidelman 2010k, date of retrieval 25.2.2012)



Offline web applications - Working Draft

Method of defining web page files to be cached using a cache manifest file, allowing them to work offline on subsequent visits to the page

*Usage stats:	Global
Support:	64.88%
Partial support:	0.15%
Total:	65.03%

Show all versions	IE	iOS Safari	Android Browser
			2.1
		3.2	2.2
		4.0-4.1	2.3
		4.2-4.3	3.0
	8.0	5.0-5.1	4.0
Current	9.0	6.0	4.1
Near future	10.0		

Figure 11. The screenshot of support a table of Offline Web Applications (@Fyrd 2012h, date of retrieval 2.11.2012)

Manifest File

A manifest file must start with CACHE MANIFEST possibly followed by an absolute or a relative URL of the file or resource to be cached. A manifest file could have optional NETWORK, CACHE and FALLBACK sections. The "#" sign is used for commenting.

NETWORK

Files and resources listed under this section are accessible when there is network connectivity and should not be cached. So, they are not available offline.

FALLBACK

This section includes a fallback page when the resource is not accessible. The reason of inaccessibility could be the lack of network connectivity or failure in caching. It contains one or more pairs of fallback resources. The first URL in the pair is the preferred URL referred to match and the second URL should be served when there is no connectivity. To use the same fallback for every resource, the first URL can be left "/" which means a root path.

CACHE

By default, the manifest file starts with "CACHE" section. URLs could be listed by declaring the CACHE: section explicitly or by leaving the manifest to handle. It is a good practice to include a CACHE section explicitly, at least when other sections are present. URLs of every resource that an application needs to operate, regardless of the type of path, relative or absolute, must be listed on separate lines. If the page visited points to the manifest file, it is cached automatically hence there is no need to list the URI into the manifest. (Apple Inc 2011, date of retrieval 25.2.2012).

Implementation

```
// Start of manifest file by default starts with CACHE section
CACHE MANIFEST
#v1 26-02-2012 //last update description in comment
index.html //caching the html file
styleOfIndex.css //caching the css file
imageOfIndex.png // caching the image

//Start of FALLBACK section with one URL fallback.html
FALLBACK:

//Start of NETWORK section with resource network. Html to be loaded when there
is a connection
NETWORK:
```

When the browser visits a page with "manifest" attribute for the first time or does not find cache, it loads the page and fetches all the resources from manifest. This will be the start of application cache. In subsequent visits, it uses JavaScript "window.applicationCache" objects to fire various events to keep the script updated. Using "window.applicationCache", the status of cache could be accessed. The status properties provided are UNCACHED, IDLE, CHECKING, DOWNLOADING, UPDATEREADY, OBSELETE of type “short” and have constant positive value of 0, 1, 2, 3, 4 and 5 respectively.(WHATWG 2011, date of retrieval 26.2.2012)

To access the cache state, various events and their handler are exposed. The events are “checking”, “noUpdate”, “downloading”, “progress”, “cached”, “updateReady”, “obsolete” and “error”. The “checking” event is always fired. If the browser does not have the manifest cached, it will fire “downloading” event to download resources listed in the manifest file. The “progress” event is fired to inform about the progress and “cached” event to notify the successful caching of the manifest file. If something goes wrong, the browser fires an “error” event and stops the downloading. The reasons for error mostly encountered are “file does not exist”, “failed to download”, “manifest file being changed”, “updated during download period”, “failing to download one or more resources listed in manifest file”. However, there could be many reasons beyond these, it is always better to debug to know the reason and source of error. If the previously visited page has cached the resource and manifest file is unchanged then “noUpdate” event will be fired. The browser fires a progress event periodically and a final “updateReady” event after the successful download. (Pligrim 2010, 137-144).

7 PROOF OF CONCEPT

This proof of concept is a demonstration of implementing the PhoneGap framework to develop a hybrid application using web technologies and deploy in multiple mobile operating systems. The code base remains the same for all the operating systems, which support PhoneGap. User Interface frameworks were selected based on compatibility.

7.1 Work Environment

The following Framework, IDEs and Tools were chosen for the application development.

PhoneGap

PhoneGap is an open source framework, initially started by Nitobi, to build cross-platform hybrid applications. It was acquired by Adobe in 2011. Currently, PhoneGap is licensed by Apache and renamed to Apache Cordova. The standard web-technologies such as HTML, CSS and JavaScript are used to create the PhoneGap applications. PhoneGap is particularly useful to apply the existing web development skills to create a mobile application, and accessing the device functionality, which a mobile browser normally cannot offer. After being wrapped with PhoneGap, the code base built for one environment could be deployed to multiple platforms. However, some tweaking in the code might be required due to the specific features the targeted device might have. PhoneGap largely reduces the resource and effort since the organization does not have to hire developers for every platform. The developer does not have to learn different languages. (Adobe Systems Inc. 2012, date of retrieval 21.10.2012)

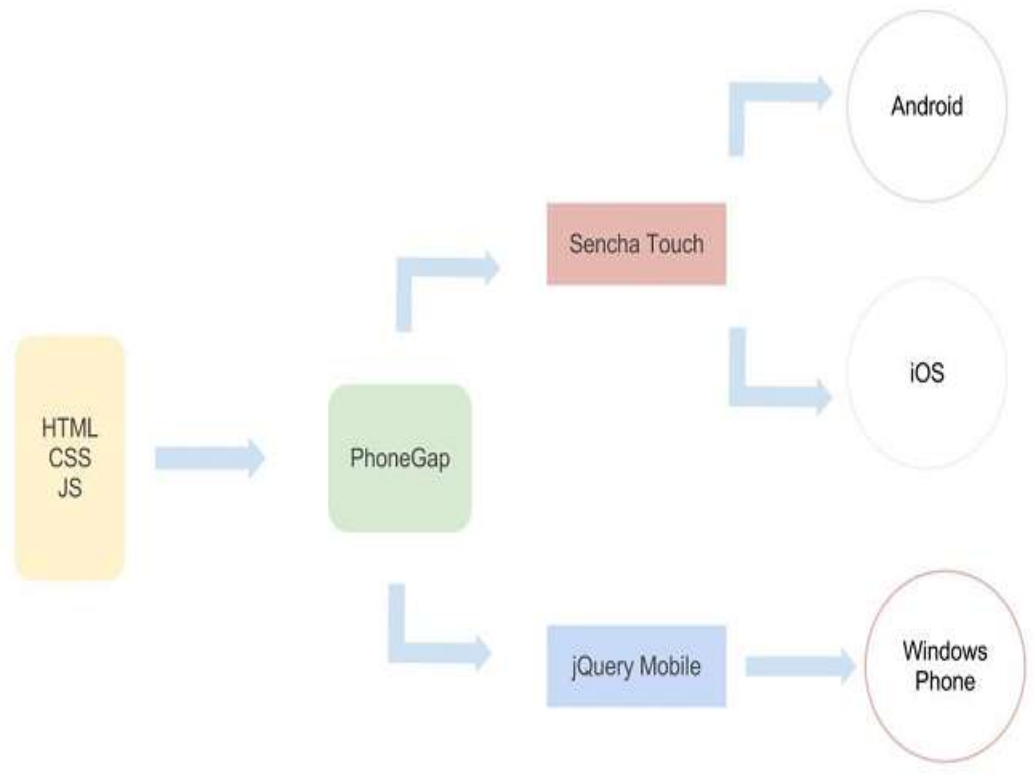


FIGURE 12. The architecture of PhoneGap Application

Figure 12 illustrates the architecture of a PhoneGap application built as a proof of the concept. The application is developed with the web – technologies, but using PhoneGap the result is a binary achieve like native ones. Two different User Interface frameworks Sencha Touch 2 and jQuery Mobile were used to test the application. As the above figure describes, the application using Sencha Touch was built for Android and iOS devices. Since Windows Phone does not support Sencha Touch 2, jQuery Mobile was used.

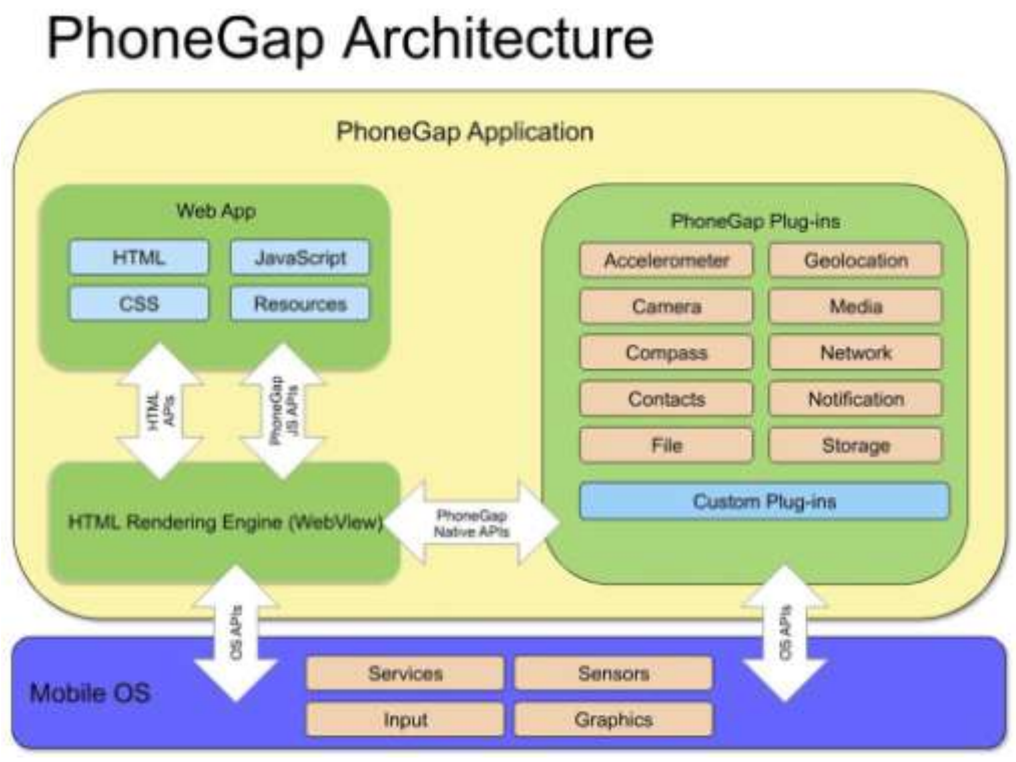


Figure 13. The screenshot of PhoneGap Architecture (IBM 2011, date of retrieval 22.10.2012)

PhoneGap accesses the web view of the native browser when the URL with optional additional properties is provided. The engine that renders the web view could be different, for instance Web Kit for an Android browser. On top of the web view, the UI could be created using HTML, CSS and JavaScript or other UI frameworks such as Sencha Touch, jQuery Mobile etc. The complete height and width of the device is available for creating a PhoneGap application. To access native functionalities, PhoneGap provides a set of APIs. The APIs act as a bridge and handle the communication between PhoneGap and Operating Systems. PhoneGap has provided a mechanism to write custom plug-in. Despite using web technologies, the application is distributed through the existing ecosystem (Google Play for Android, AppStore for iOS, and Windows phone store for Windows Phone). The PhoneGap application can communicate with Users and Servers. The servers are responsible for the business logic and communicating with the databases or other repositories. (Trice 2012a, date of retrieval 22.10.2012).

Sencha Touch

Sencha Touch is one of the most powerful application frameworks to build complex hybrid apps. It has a steep learning curve and it does not completely support all the native functionalities like the PhoneGap but it perfectly replicates the native UI and provides a great support for touch events and animations. (Zwick 2010, date of retrieval 22.10.2012)

jQuery Mobile

jQuery Mobile is a cross-platform User Interface framework under MIT license. It supports major operating systems and could be used on top of other native functionality accessing frameworks such as PhoneGap. Easy learning curve is its strength. (The jQuery Foundation 2012, date of retrieval 23.10.2012).

Eclipse

Eclipse is an open source Integrated Development Environment maintained by Eclipse.org. Along with Android SDK and ADT plugin, Eclipse is ready to create Android applications. The application could be run in emulator and device. Android Virtual Device must be set up, to run in an emulator. For deploying in a device, it could be done by connecting the device via a USB cable or generating the .apk (Android application packaged) file. (Trice 2012b, date of retrieval 23.10.2012).

Microsoft Visual Studio Express

It is a Microsoft proprietary IDE for creating Windows Phone Application. The created application could be deployed to an emulator and a device. Visual Studio comes with the Windows Phone SDK or it could be integrated later if a previous version of Visual C# Express or Visual Studio 2008 or later version is installed. (Microsoft 2012, date of retrieval 23.10.2012).

Xcode

It is Apple's proprietary IDE to create applications for Mac OS and iOS. The iOS application could be deployed to an emulator as well as an iOS device such as iPhone, iPad. (Apple Inc.2012b, date of retrieval 25.10.2012)

Aptana

Aptana is an open source IDE used for web development, maintained by Appcelerator Inc. It is mostly used with an eclipse for writing and debugging web technologies namely HTML, CSS and JavaScript. (Appcelerator, Inc (2005-2011), date of retrieval 23.10.2012)

7.2 Application, Testing and Findings

The application was built to testify:

- If web technologies could be used to create a hybrid application, that runs over different mobile operating systems.
- If PhoneGap can access the native functionalities of different platforms, using the same code base.
- If mobile UI frameworks could be used together with PhoneGap to provide the hybrid app with a better look and feel.
- How the app behaves when ported to different mobile operating systems.

Setting up Environment

Eclipse, Visual Studio Express for Windows Phone, and Xcode should be installed and properly setup. The PhoneGap plug-in for the respective mobile operating system must be downloaded.

Set-up PhoneGap for Android

For Android, the .js file must be placed in a WWW folder, which is a sub-folder of the assets folder. The .jar file must be inside manually created “libs” folder and a build path must be configured properly. The activity class must extend DroidGap. “super.loadUrl (“file:///android_asset/www/index.html”)” is called within “onCreate()” method . The method loads a web view in the index.html file through which the application can navigate. The PhoneGap fires the “deviceready” event when PhoneGap.js is loaded. All other PhoneGap functions can be called after the device is ready.

PhoneGap for Windows Phone

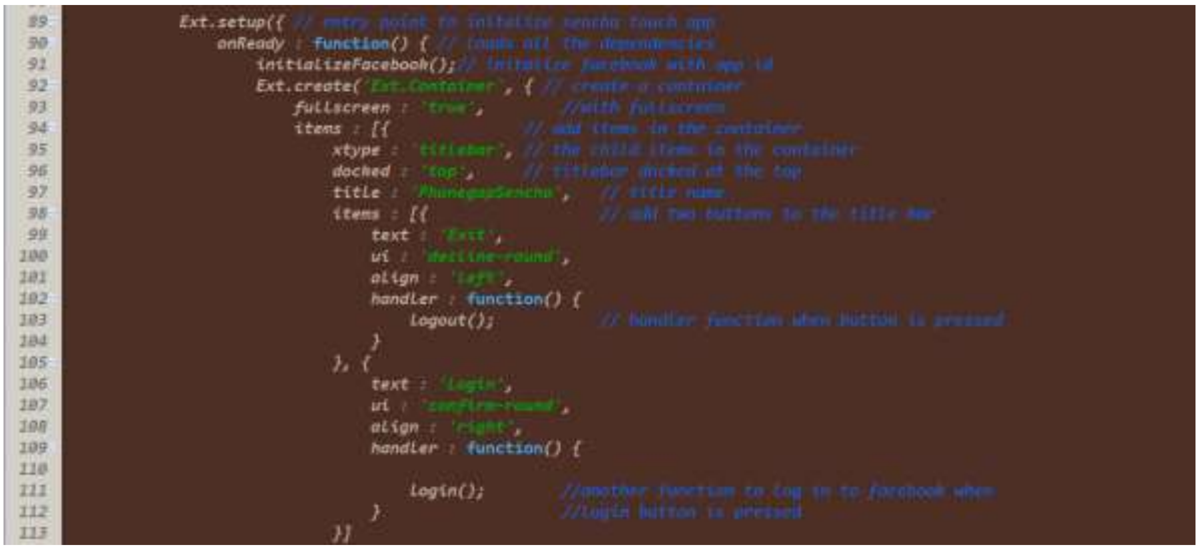
In a windows phone, PhoneGapStarter.zip must be copied to the Silverlight for Windows Phone folder inside C:\Users\[username]\Documents\ VisualStudio2010\Templates\Project Templates. If the folder does not exist, it should be manually created. Select a PhoneGap starter application while creating a new project (CordovaStarter for newer versions with the version number). The “deviceReady” callback is not fired on Windows Phone. Instead of that, it is “binded” to an initialize function. The “app. initialize()” function is called to check if the PhoneGap has finished loading.

PhoneGap for iOS

For iOS, the PhoneGap installation wizard guides the developer to install PhoneGap. Before that the downloaded PhoneGap content should be extracted in under “libs/iOS” folder and the package installer should be run. If the PhoneGap installation is complete, it adds a PhoneGap project template to Xcode. Using that template a new PhoneGap project can be created.

User Interface

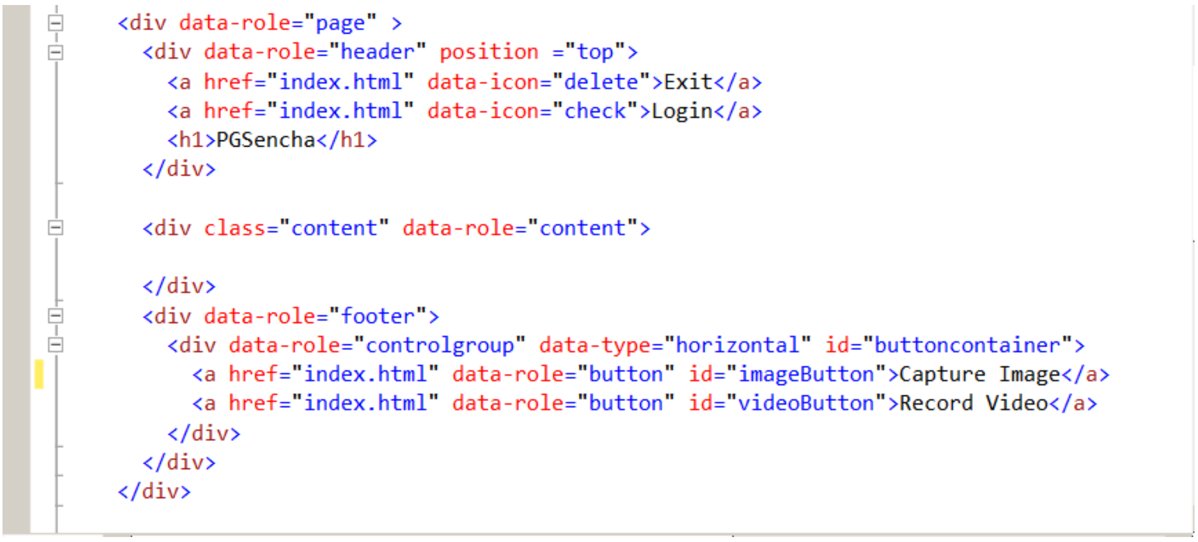
Two different UI frameworks were implemented to build the application. Sencha Touch 2 was used to create a user interface for Android and iOS operating systems and jQuery mobile for the Windows Phone. Sencha Touch only works with a web-kit based browser but not with Windows Phone's browser.



```
Ext.setup({ // entry point to initialize sencha touch app
    onReady : function() { // loads all the dependencies
        initializeFacebook();// initialize facebook with app id
        Ext.create('Ext.Container', { // create a container
            fullscreen : 'true',        //with fullscreen
            items : [{               // add items in the container
                xtype : 'titlebar', // the child items in the container
                docked : 'top',      // titlebar docked at the top
                title : 'PhonegapSencha',   // title name
                items : [{                  // add two buttons to the title bar
                    text : 'Exit',
                    ui : 'decline-round',
                    align : 'left',
                    handler : function() {
                        Logout();               // handler function when button is pressed
                    }
                }, {
                    text : 'Login',
                    ui : 'confirm-round',
                    align : 'right',
                    handler : function() {

                        Login();          //another function to log in to facebook when
                                          //login button is pressed
                    }
                }]
```

Figure 14. The screenshot of UI Code for Android and iOS using Sencha Touch 2



```
<div data-role="page" >
  <div data-role="header" position ="top">
    <a href="index.html" data-icon="delete">Exit</a>
    <a href="index.html" data-icon="check">Login</a>
    <h1>PGSencha</h1>
  </div>

  <div class="content" data-role="content">

  </div>
  <div data-role="footer">
    <div data-role="controlgroup" data-type="horizontal" id="buttoncontainer">
      <a href="index.html" data-role="button" id="imageButton">Capture Image</a>
      <a href="index.html" data-role="button" id="videoButton">Record Video</a>
    </div>
  </div>
</div>
```

Figure 15. The screenshot of UI Code for Windows Phone using jQuery Mobile

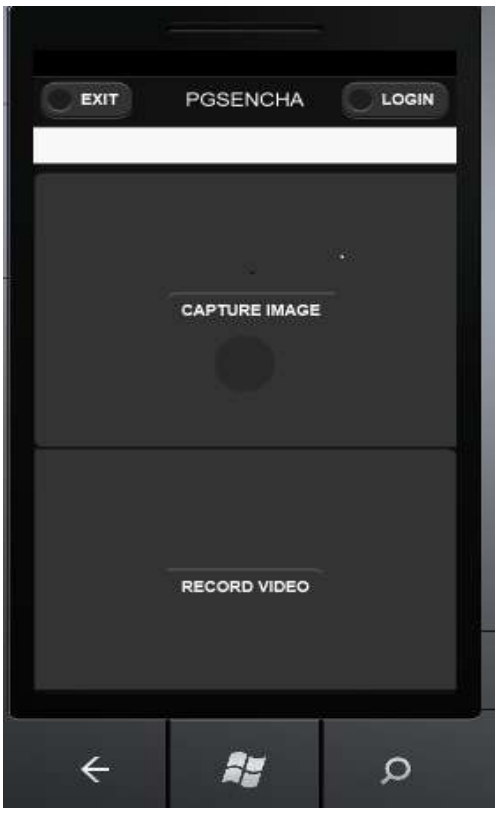


Figure 16. The screenshot of UI in Windows Phone Emulator while capturing image.

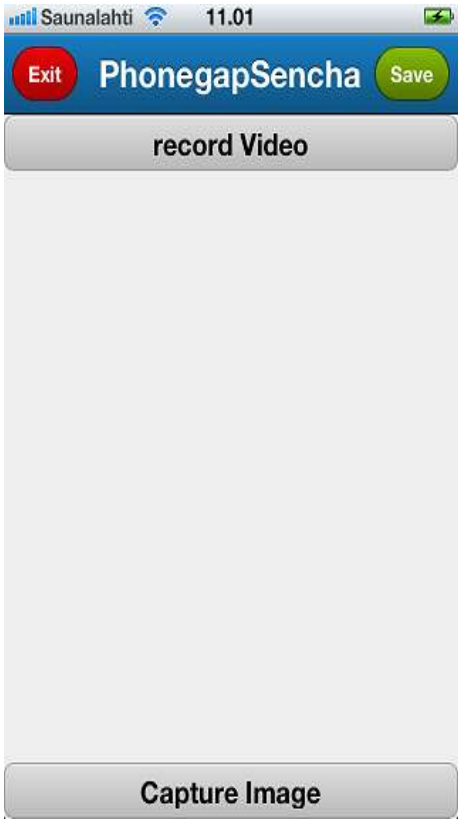


FIGURE 17. The screenshot of UI in iPhone.

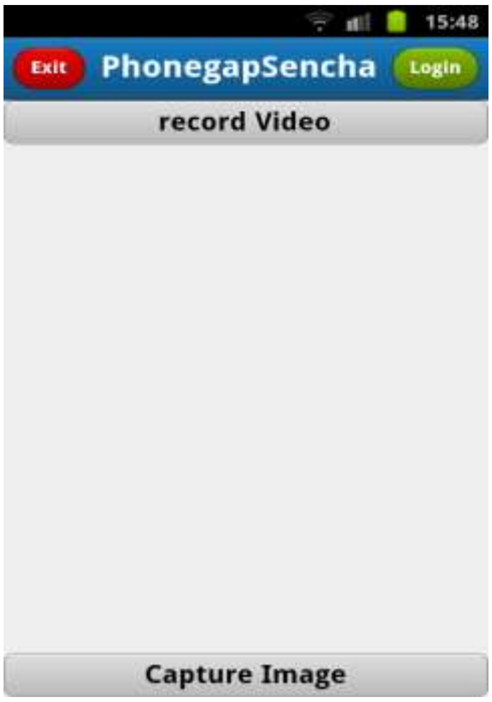


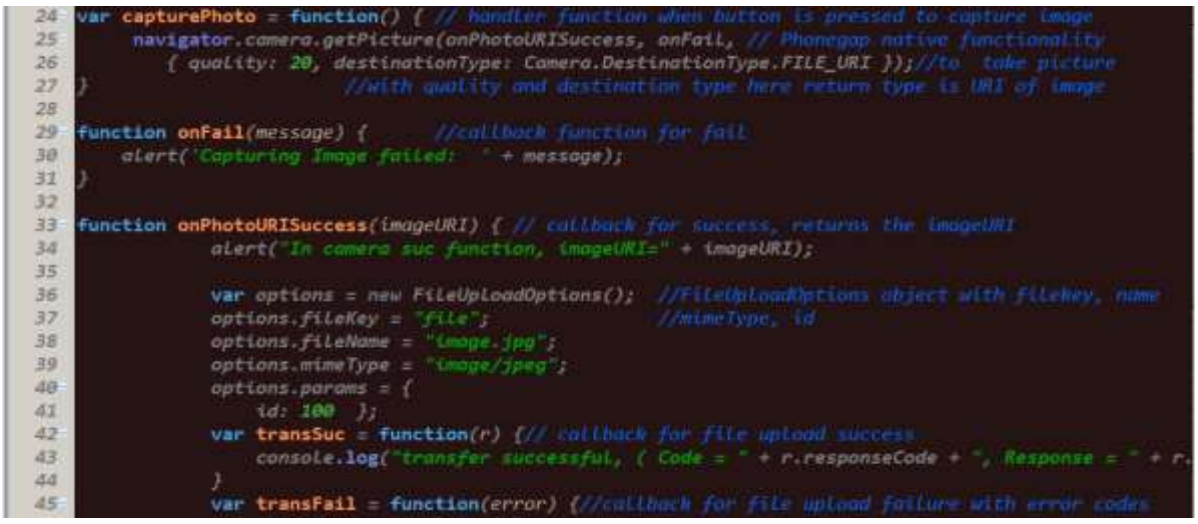
FIGURE 18. The screenshot of UI in SG-S

Single Code Base for Multiple Platforms

The application is intended to capture images and record a video, upload an image to a PHP server and provide the image location in the server. The application uses the URL to upload the image to the Facebook. The image and video recorded are stored in the SD card of the Phone, too. Currently the native PhoneGap-plug-in-facebook-connect are only available for iOS and Android platforms.

Capture Image

There are two buttons available to capture an image and to record a video. To capture an image, "getPicture()" method is invoked by the camera object. The method contains callback methods for success and failure and optional parameters of quality and type of expected image. If the image capture is successful, URI of the image is retrieved. The other option is to retrieve a base64-encoded string of the image. To upload an image, PhoneGap provides File Transfer API, whose instance is created to call the “upload()” method with the parameters including imageURI returned from the success callback function above. Other parameters include the complete URL of the server, success callback, error callback functions and an instance of FileUploadOptions with the key, name, mime type and id. Upload success calls “transuc()” method which returns a response code, a response and a number of bytes sent, the failure returns error code as an alert message.



```
var capturePhoto = function() { // handler function when button is pressed to capture image
    navigator.camera.getPicture(onPhotoURISuccess, onFail, // Phonegap native functionality
        { quality: 20, destinationType: Camera.DestinationType.FILE_URI });//to  take picture
}                       //with quality and destination type here return type is URI of image

function onFail(message) {      //callback function for fail
    alert('Capturing Image failed:  ' + message);
}

function onPhotoURISuccess(imageURI) { // callback for success, returns the imageURI
            alert("In camera suc function, imageURI=" + imageURI);

            var options = new FileUploadOptions();  //FileUploadOptions object with fileKey, name
            options.fileKey = "file";               //mimeType, id
            options.fileName = "image.jpg";
            options.mimeType = "image/jpeg";
            options.params = {
                id: 100  };
            var transSuc = function(r) {// callback for file upload success
                console.log("transfer successful, ( Code = " + r.responseCode + ", Response = " + r.
            }
            var transFail = function(error) {//callback for file upload failure with error codes
```

Figure 19. The screenshot of code to capture image used in multiple platform

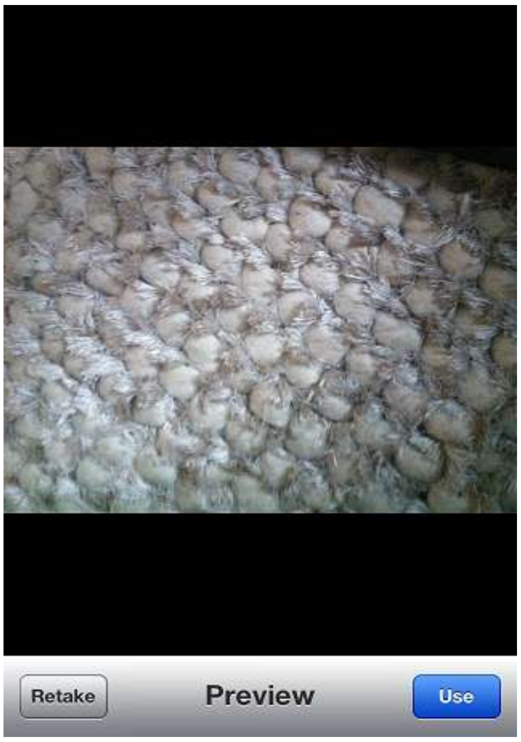


Figure 20. The screenshot of iPhone while capturing image.

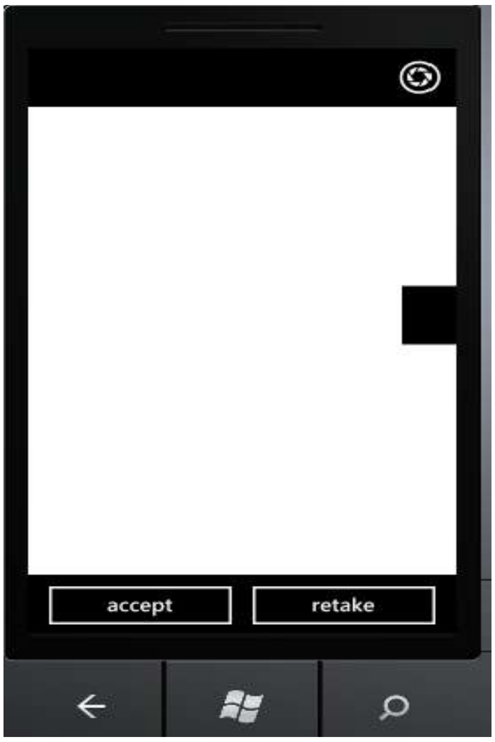


FIGURE 21. The screenshot of Windows Phone emulator while capturing image

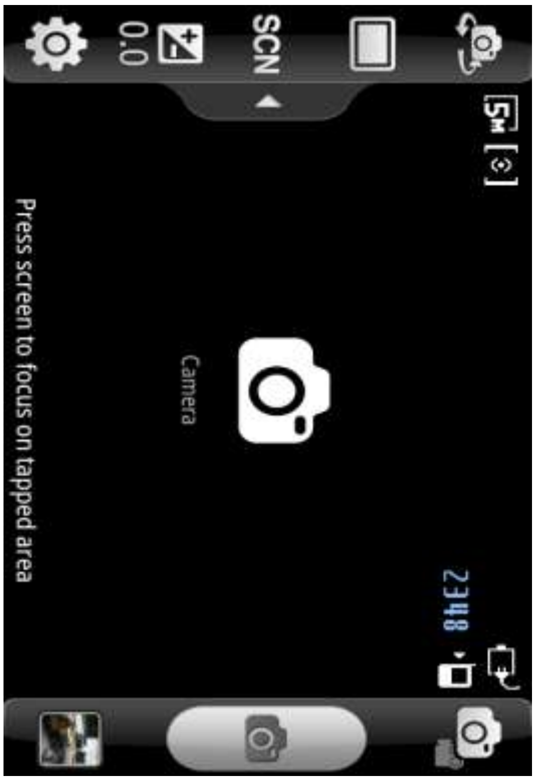
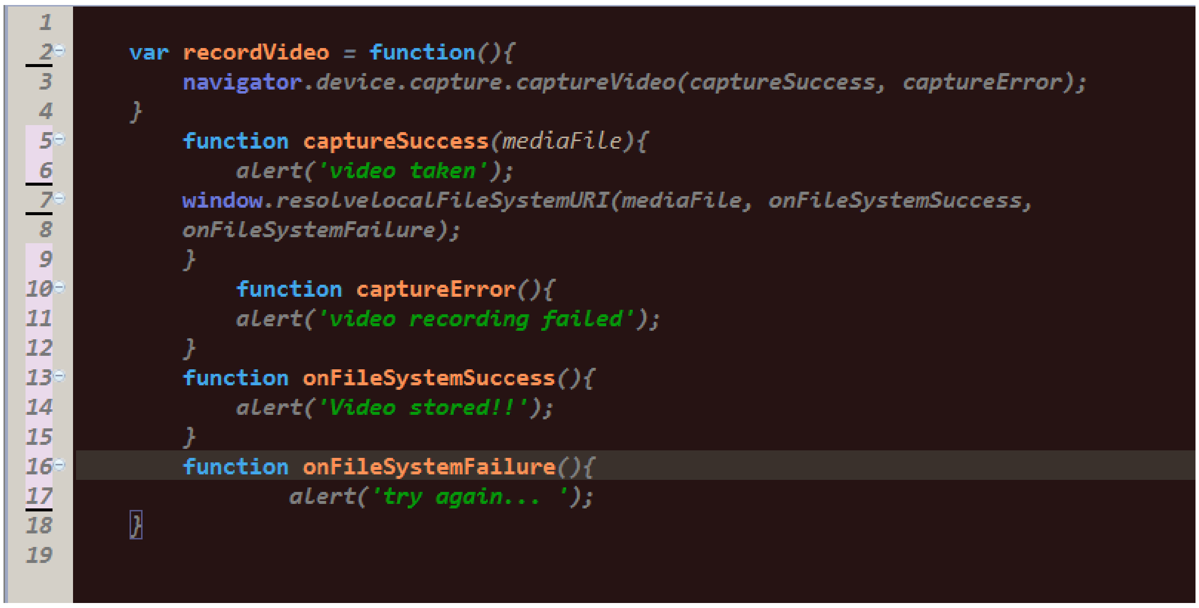


FIGURE 22. The screenshot of SG-S while capturing image.

Record Video

When pressing the Record Video button, the application calls for "captureVideo()" method and starts recording. This method takes success and failure callback functions and an optional, limit option, for the number of times the video recording application can be used. If the application cannot record video, the capture error is called. If the video recording is successful, the success callback which has mediaFile is called. The "resolveLocalFileSystemURI()" function provides the information about the Directory Entry or File Entry depending upon the type.



```

var recordVideo = function(){
    navigator.device.capture.captureVideo(captureSuccess, captureError);
}
    function captureSuccess(mediaFile){
        alert('video taken');
    window.resolveLocalFileSystemURI(mediaFile, onFileSystemSuccess,
    onFileSystemFailure);
    }
        function captureError(){
        alert('video recording failed');
    }
    function onFileSystemSuccess(){
        alert('Video stored!!');
    }
    function onFileSystemFailure(){
            alert('try again... ');
}

```

Figure 23. The screenshot of a code to record a video in multiple platforms

Upload Captured Image to Facebook

To host the image captured and uploaded from the application, a PHP server was set up. The server receives the image from FileTransfer object of PhoneGap and it moves to a folder. The URL of image is used to provide image to "post" method of Facebook's Graph API. Every time a picture is taken and uploaded to the server overriding the previous image.

Setup for Android

Place facebook_js_sdk.js and cdv-plugin-fb-connect.js in the WWW folder. The Facebook JavaScript SDK allows Facebook Login use APIs through JavaScript. Cordova plugin is a JavaScript version of the plug-in. Connectplugin.java, the native plug-in for Android should be placed under “org.apache.cordova.facebook” folder. The “com.facebook.android” package should be placed inside the source folder to compile the ConnectPlugin.

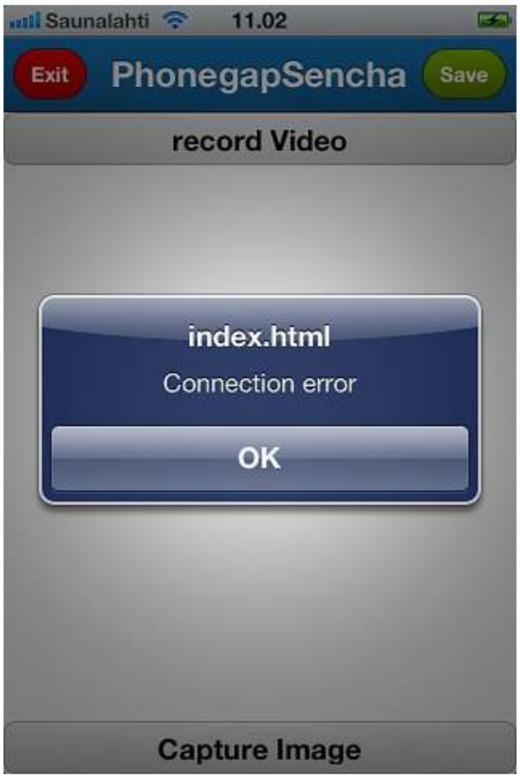
SetUp For iOS

Place the facebook_js_sdk.js into WWW directory in XCode. After selecting the Target, link the necessary frameworks. Copy the contents from “native/ios” in Cordova folder to XCode and WWW folder to WWW directory in Xcode. The necessary domains should be whitelisted and scheme should be added.

Post to Facebook

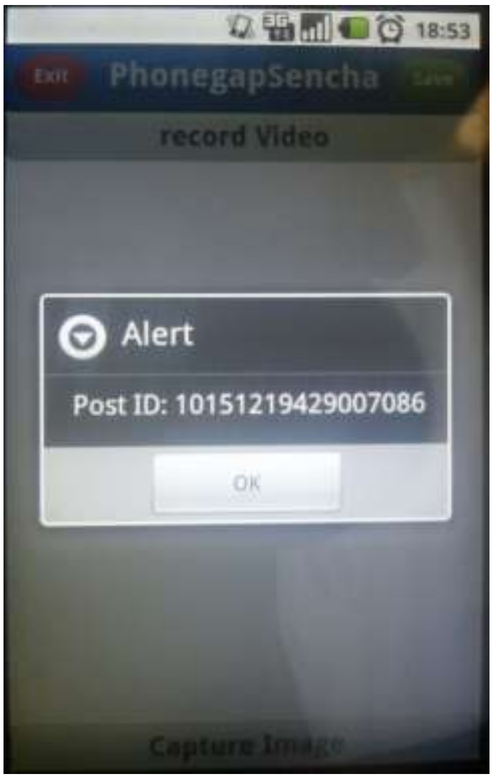
First, the app needs to be initialized with “app ID” and other parameters like “nativeInterface”, “useCachedDialogue”. Init function is called when the PhoneGap is ready to be used. If the app is correctly initialized, then it will display a log message saying “Cordova Facebook Connect plugin initialized successfully.” Otherwise, there might be error in configuration, which needs to be rectified.

To upload an image, the Graph API is called with necessary parameters. In this application the destination to be uploaded is the “me/photos” and the method is “post” and the image URL is above mentioned URL from the PHP server. The image is passed from the URL of the server, where the image is first uploaded using FileTransfer API. The other parameters are message and oAuth. Facebook responds with post ID if successful and response if there is any error.



index.html

Connection error

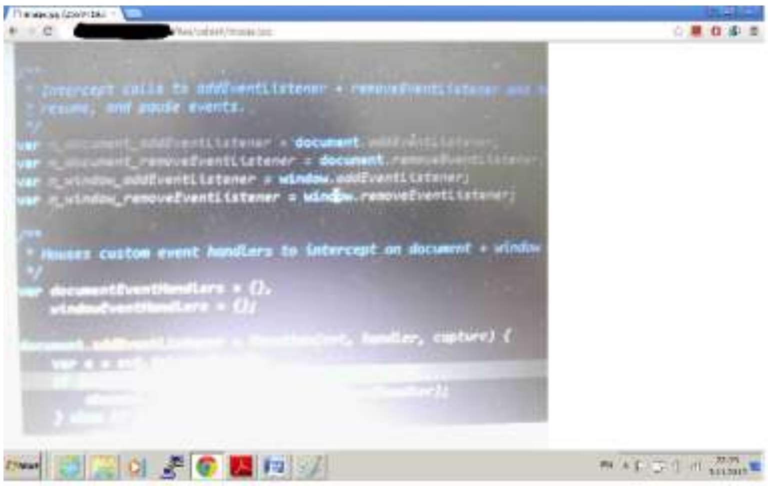


Alert

Post ID: 10151219429007086

FIGURE 24: The screenshot of an Image uploaded to Facebook from iOS

FIGURE 25. The screenshot of an image uploaded successfully to Facebook



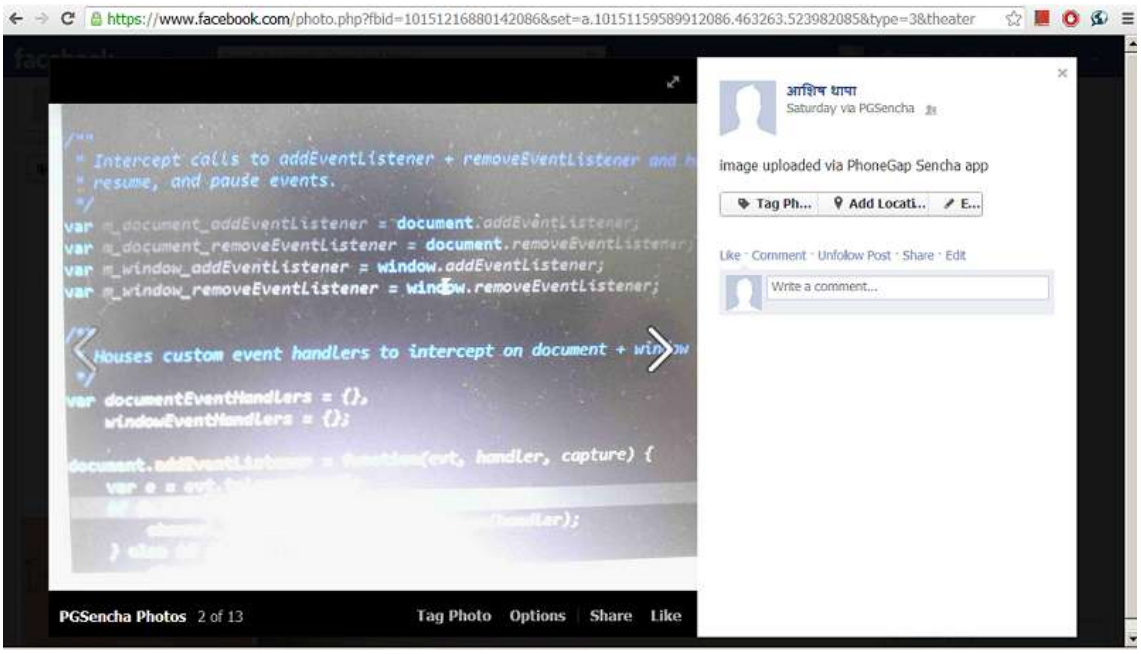
[redacted]

```
/**
 * Intercept calls to addEventListener + removeEventListener and h
 * resume, and pause events.
 */
var m_document_addEventListener = document.addEventListener;
var m_document_removeEventListener = document.removeEventListener;
var m_window_addEventListener = window.addEventListener;
var m_window_removeEventListener = window.removeEventListener;

/**
 * Houses custom event handlers to intercept on document + window
 */
var documentEventHandlers = {},
    windowEventHandlers = {};

document.addEventListener = function(evt, handler, capture) {
    var e = [illegible]
    [illegible]
        [illegible] handler);
    } else [illegible]
```

FIGURE 26. The screenshot of an Image in Server


https://www.facebook.com/photo.php?fbid=10151216880142086&set=a.10151159589912086.463263.523982085&type=3&theater
PGSencha Photos 2 of 13
Tag Photo
Options
Share
Like
Tag Ph...
Add Locati..
E...
Like · Comment · Unfollow Post · Share · Edit
Write a comment...

आशिष थापा
Saturday via PGSencha

image uploaded via PhoneGap Sencha app

```
/**
 * Intercept calls to addEventListener + removeEventListener and h
 * resume, and pause events.
 */
var m_document_addEventListener = document.addEventListener;
var m_document_removeEventListener = document.removeEventListener;
var m_window_addEventListener = window.addEventListener;
var m_window_removeEventListener = window.removeEventListener;

/**
 * Houses custom event handlers to intercept on document + window
 */
var documentEventHandlers = {},
    windowEventHandlers = {};

document.addEventListener = function(evt, handler, capture) {
    var e = evt.[illegible]
    [illegible]
        [illegible](handler);
    } else [illegible]
```

FIGURE 27. The screenshot of an image uploaded by the application in Facebook

Application Results and Findings

The Phonegap application was built and tested in Windows Phone, Android and iOS operating systems. The application runs over all the platforms. Due to unavailability of Windows Phone device, the emulator was used to test the application. All three operating systems supported Image capture event. Android and iOS devices supported video recording and file uploading. But Windows Phone emulator did not support video recording. Image and Video were also stored locally in the SD card of the devices. There was a configuration issue between iOS and Facebook plug-in in the application, so the image uploading functionality did not work on iOS. In Android, the image upload to Facebook was also successful. It was an extended functionality test, if PhoneGap application could be taken to another level.

Possibility of further development

This application has a possibility of further development. I plan to run and test the application in a Windows Phone device. The configuration issue in the application between iOS and Facebook plug-in could be solved and it should be able to upload the image. The application in Android currently does what it was intended to do. With some minor changes in User interface and functionality, it should be possible to launch in the Android Market Place. But, I plan to add a video uploading functionality before launching.

8 CONCLUSION

The topic of this bachelor’s thesis was very challenging, and yet interesting. An immense research and effort was required for the completion of this thesis. The results achieved were very satisfying.

The research was carried out by studying multimedia and storage of HTML5 and their cross-platform nature. The research showed that HTML5 is still on the way to be a complete cross-platform. Some of the desired functionalities are currently available for all the operating systems but other functionalities are on the process of development. The first complete draft is expected by 2014. To test the currently supported functionalities, the application was developed as a proof of the concept. The application worked across all the platforms as expected, using the same code base. However, some obvious minor glitches were discovered. Therefore HTML5 retains the cross-platform nature on functionalities that are currently supported.

Personally, this thesis helped me to gain a good knowledge of different browsers, operating systems, multimedia and storage of HTML5. I now have better understanding about the hybrid application architecture. I believe the knowledge that I have acquired during the study of this new technology will be useful for the future.

LIST OF REFERENCES

2Expert, 2012. What Is It Important for HTML5 To Defeat Flash?. Date of retrieval 22.4.2012
http://www.2expertsdesign.com/web-designs/what-is-it-important-for-html5-to-defeat-flash

@Fyrd, 2010. Can I use. Date of retrieval 16.1.2012
http://caniuse.com/

@Fyrd, 2012a. Can I use. Date of retrieval 16.1.2012
http://caniuse.com/#feat=video

@Fyrd, 2012b.Can I use. Date of retrieval 2.11.2012
http://caniuse.com/#feat=audio

@Fyrd, 2012c. Can I use. Date of retrieval 2.11.2012
http://caniuse.com/#feat=namevalue-storage

@Fyrd, 2012d. Can I use. Date of retrieval 2.11.2012
http://caniuse.com/#feat=sql-storage

@Fyrd. 2010e. Can I use. Date of retrieval 2.11.2012
http://caniuse.com/#feat=indexeddb

@Fyrd. 2010f. Can I use. Date of retrieval 2.11.2012
http://caniuse.com/#feat=filereader

@Fyrd. 2010g. Can I use. Date of retrieval 2.11.2012
http://caniuse.com/#feat=filesystem

@Fyrd. 2010h. Can I use. Date of retrieval 2.11.2012
http://caniuse.com/#feat=offline-apps

Adhikari, R. 2011. Adobe Sends Mobile Flash Packing. Date of retrieval 9.2.2012
http://www.technewsworld.com/story/73720.html

Adobe Systems Inc, 2012. About the Project. Date of retrieval 21.10.2012
http://www.phonegap.com/about

Appcelerator Inc, 2005-2011. Date of retrieval 23.10.2012
http://www.aptana.com/

Apple Inc, 2011. HTML5 Offline Application Cache. Date of retrieval 25.2.2012
http://developer.apple.com/library/safari/#documentation/iPhone/Conceptual/SafariJSDatabaseGuide/OfflineApplicationCache/OfflineApplicationCache.html

Apple Inc, 2012a. Develop Apps for iOS. Date of retrieval 18.1.2012
https://developer.apple.com/technologies/ios/

Apple Inc, 2012b. Developer Tools. Date of retrieval 25.10.2012
https://developer.apple.com/technologies/tools/

Bidelman, E. 2011a, EXPLORING THE FILE SYSTEM APIS, Date of retrieval 21.2.2012
http://www.html5rocks.com/en/tutorials/file/filesystem/

Bidelman, E. 2011b. Using the HTML5 Filesystem API. First Edition. CA: O'Reilly Media

Bidelman, E. 2011k, A BIGINNER'S GUIDE TO USING THE APPLICATION CACHE. Date of retrieval 21.2.2012
http://www.html5rocks.com/en/tutorials/appcache/beginner/

Buzzle, 2012. The Need for a Smartphone. Date of retrieval 30.1.2012
http://www.buzzle.com/articles/the-need-for-a-smartphone.html

Closs, T. 2011. HTML5 vs. Native apps - what’s between the devil and the deep blue sea?. Date of retrieval 1.2.2012
http://www.madewithmarmalade.com/blog/html5-vs-native-apps-%E2%80%93
what%E2%80%99s-between-devil-and-deep-blue-sea

Devlin, I. HTML5 Multimedia: DESIGN and DEVELOP. Berkeley, CA: Peachpit Press

Flirtman, M. 2010. Programming the Mobile Web. CA: O'Reilly Media

Hill, S. 2010. Overview of the BlackBerry OS. Date of retrieval 30.1.2012
http://www.brighthub.com/mobile/blackberry-platform/articles/87707.aspx

HTML5ROCKS 2012b, HTML5 FEATURES STORAGE. Date of retrieval 14.2.2012
http://www.html5rocks.com/en/features/storage

IBM 2011, PhoneGap Day – IBM, PhoneGap and the Enterprise. Date of retrieval 22.10.2012
http://www.slideshare.net/drbac/phonegap-day-ibm-phonegap-and-the-enterprise

Kappart, L. 2011. Introduction to HTML5 Web Storage. Date of retrieval 15.2.2012
http://sixrevisions.com/html/introduction-web-storage/

Keith, J. 2010. HTML5 FOR WEB DESIGNERS: A BRIEF HISTORY OF MARKUP. New York: Jeffrey Zeldman

Kravchick, O. 2011. Native vs. HTML5 Application Development. Date of retrieval 31.1.2012
http://myok12.wordpress.com/2011/06/29/native-vs-html5-application-development/

Lee, Wei-Meng. 2011. Beginning Android Application Development. Hoboken, NJ, USA: Wrox

Mahemoff, M. 2010. CLIENT-SIDE STORAGE. Date of retrieval 14.2.2012
http://www.html5rocks.com/en/tutorials/offline/storage/

Mahemoff, M. 2011. “OFFLINE”: WHAT DOES IT MEAN AND WHY SHOULD I CARE?. http://www.html5rocks.com/en/tutorials/offline/whats-offline/

Microsoft, 2012. Visual Studio 2010 Express For Windows Phone. Date of retrieval 23.10.2012
http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff630878(v=vs.92).aspx

Modernizr, 2009-2012. Documentation. Date of retrieval 31.1.2012
http://modernizr.com/docs/

Pilgrim, M. 2010. HTML5: Up and Running. First Edition. CA: O’Reilly Media, Inc

Ranganathan, A. & Wilsher, S. Firefox 4: An early walk-through of IndexedDB. Date of retrieval 16.2.2012
https://hacks.mozilla.org/2010/06/comparing-indexeddb-and-webdatabase/

Refsnes Data, 1999-2012a. HTML5 New Elements. Date of retrieval 16.1.2012
http://www.w3schools.com/html/html5_new_elements.asp

Resnes Data. 1999-2012b. HTML <video> tag. Date of retrieval 10.2.2012
http://www.w3schools.com/tags/tag_video.asp

Resnes Data. 1999-2012c. HTML Multimedia. Date of retrieval 8.2.2012
http://www.w3schools.com/html/html_media.asp

Refsnes Data. 1999-2012c. HTML5 Web Storage. Date of retrieval 16.1.2012
http://www.w3schools.com/html/html5_webstorage.asp

Rouget, A. 2010. Offline web applications. Date of retrieval 24.2.2012
https://hacks.mozilla.org/2010/01/offline-web-applications/

Raggett, D. , lam, J. , Alexander, I. & Kmiec, M. 1998. Ragget on HTML4. Second Edition. Harlow, England: Adinson Wesley Longman

Siegler, M. 2011. HTML5 Is An Oncoming Train, But Native App Development Is an Oncoming Rocket ship. 1.2.2012
http://techcrunch.com/2011/02/09/html5-versus-native-apps/

Sharp, R. 2010. Introducing Web SQL Databases. Date of retrieval 5.3.2012
http://html5doctor.com/introducing-web-sql-databases/

Sheridan, M. 2011, Building Web Pages with Local Storage. Date of retrieval 15.2.2012
http://www.sitepoint.com/building-web-pages-with-local-storage/

thebeebs, 2011. Mobile Apps in HTML5: do it but realize, it's not a panacea. Date of retrieval 1.2.2012
http://blogs.msdn.com/b/thebeebs/archive/2011/12/19/mobile-apps-in-html5-do-it-but-realise-it-s-not-a-panacea.aspx

The jQuery Foundation. 2012. jQuery Mobile. Date of retrieval 23.10.2012
http://www.jquerymobile.com

Trice, A. 2012a. PhonegGap Explained Visually. Date of retrieval 22.10.2012
http://phonegap.com/2012/05/02/phonegap-explained-visually/

Trice, A. 2012b. Getting started with PhoneGap in Eclipse for Android. Date of retrieval 23.10.2012
http://www.adobe.com/devnet/html5/articles/getting-started-with-phonegap-in-eclipse-for-android.html

W3C. 2012a. HTML5 differences from HTML4. Date of retrieval 16.1.2012
http://www.w3.org/TR/2012/WD-html5-diff-20121025/

W3C. 2011b. Web Storage. Date of retrieval 15.2.2012
http://www.w3.org/TR/webstorage/

W3C. 2011c. WEB SQL. Date of retrieval 15.2.2012
http://www.w3.org/TR/webdatabase/

W3C, 2011d. FILE API. Date of retrieval 22.2.2012
http://www.w3.org/TR/FileAPI/

W3C, 2011d. FILE API. Date of retrieval 9.3.2012
http://www.w3.org/TR/FileAPI/

W3C, 2011e. File API: Writer. Date of retrieval 10.3.2012
http://www.w3.org/TR/file-writer-api/

W3C, 2011f. File API: Directories and System. Date of retrieval 11.3.2012
http://www.w3.org/TR/file-system-api/

WHATWG, 2011. Offline Web Application. Date of retrieval 26.2.2012
http://www.whatwg.org/specs/web-apps/current-work/multipage/offline.html#appcacheevents

Wikipedia 2011, date of retrieval 16.1.2012
http://en.wikipedia.org/wiki/HTML5

Wikipedia 2012, date of retrieval 10.2.2012
http://en.wikipedia.org/wiki/HTML5_video

Zhou, Z., Zhu, R. Zheng, P. & Yang, B. 2011. Windows Phone 7 Programming for Android and iOS Developers. Hoboken, NJ, USA: Wrox

Zwick, C. 2010. Sencha Touch: The HTML5 Mobile App Framework. Date of retrieval 22.10.2012
http://mobile.tutsplus.com/articles/news/sencha-touch-html5-mobile-framework/

LIST OF FIGURES