Multi-platform mobile application development using HTML5, JavaScript and CSS3.

Simon Facciol
June 2011

Submitted to the Institute of Information & Communication Technology in partial fulfilment of the requirements for the Bachelor of Science (Honours) in Software Development
AUTHORSHIP

This dissertation is based on the results of research carried out by myself, is my own composition, and has not been previously presented for any other certified or uncertified qualification.

The research was carried out under the supervision of Mr James Decelis.

Signed ______________________  Date ______________________
COPYRIGHT

In submitting this dissertation to the MCAST Institute of Information & Communication Technology I understand that I am giving permission for it to be made available for use in accordance with the regulations of MCAST and the College Library.

Signed ______________________  Date ______________________
ACKNOWLEDGEMENTS

I gratefully acknowledge the contributions of my tutor, Mr James Decelis, whose patience, dedication, and knowledge guided me throughout the whole research process. I also thank my work colleague Mr Andre Masini for providing me with his iPad for testing, and also my employer Mr Terje Myklebust for allowing me to work flexible hours to work more on my Thesis. Most importantly, I also thank my family for giving me the opportunity to pursue this degree and for supporting me during the process. Finally, I also thank my girlfriend Christianne for her love, support and encouragement to complete my Thesis and the Degree course, but especially for being patient throughout.
CONTENTS

Authorship .................................................................................................................. 1
Copyright ...................................................................................................................... 2
Acknowledgements ...................................................................................................... 3
Contents ......................................................................................................................... 4
Figures and Screenshots ............................................................................................. 5
Abstract ......................................................................................................................... 6
Introduction ..................................................................................................................... 8
Literature Review .......................................................................................................... 12
Methodology ................................................................................................................... 25
Results and Analysis ..................................................................................................... 35
Conclusion ....................................................................................................................... 44
Bibliography .................................................................................................................... 47
Appendix 1 ...................................................................................................................... 51
FIGURES

Figure 1 - Facebook’s native application and HTML5 application (Schranz, 2010) .......... 22

Figure 2 - Google’s gmail native application and HTML5 Application (Schranz, 2010) ....... 22

Figure 3 - Nextstop’s native application and HTML5 application (Schranz, 2010) .......... 23

Figure 4 - Showtime html5 web application (Showtime) ................................................. 23

Figure 5 - Portrait view of the Web Application on the iPad ........................................... 51

Figure 6 - Landscape view of the Web application on the iPad ........................................ 52

Figure 7 - The error shown on the iPad when trying to run the application without an internet connection .................................................................................................................. 52

Figure 8 - Portrait view of the Web application on the HTC Desire ................................. 53

Figure 9 - Landscape view of the Web application on the HTC Desire ............................. 53
ABSTRACT

Although the HTML5 standard was under development since 2004, Web browsers started to implement and support the new standard only in the last 2 years. The emerged popularity of smartphones, tablets and other mobile devices that came pre-installed with an HTML5 ready Web Browser, multi-platform mobile HTML5 Web applications were now a feasible option. One of the new features of HTML5 offered the possibility to cache a Web application offline, and when there was no internet connection available the application could still be used. Therefore, there was a potential to create native looking offline Web applications that would provide similar functionality like native mobile applications. However since the HTML5 standard was pretty new, there was no research whatsoever about the feasibility and success to build an offline HTML5 Web application, nor if it consumed less time to develop one Web application instead of multiple native applications.

As a result, an offline HTML5 mobile Web application was developed, using the AppML framework, in order to examine the current implementation and support of the HTML5 standard on mobile devices. An existing Windows 7 only application was used as a model in order to develop the HTML5 application. A checklist was created to determine the criteria that the Web application was going to be assessed on, to finally determine the status of HTML5 application’s functionality and design on various mobile devices. The application was also tested against the W3C mobileOK checker, to test the application’s structure against a third-party application for a better evaluation of the code used to develop the application.

The outcomes of the results showed how the majority of the HTML5 features were supported, after the Web application was tested on several mobile devices. Nevertheless, the Web browser implementation of the HTML5 standard lacked some vital implementation features, which would otherwise make offline mobile Web applications competitive against Multi-platform mobile application development using HTML5, JavaScript and CSS3.
mobile native applications. When the application that was developed to test HTML5 on mobile devices was analysed attentively, it resulted that sometimes it was not the application’s code which made the application unusable or inconsistent, but merely the incorrect implementation of the standard by the Web browser. To build an HTML5 Web application for multiple mobile devices consumed less time to develop compared to native application development where two or more applications were developed for each device Operation System. Nevertheless, the fact that there were still some bugs that had to be singularly examined and fixed for each targeted mobile device, signified the inconsistency of the implementation of the HTML5 standards in Web browsers.
INTRODUCTION

The Beginning

From the time when the Internet was created, its main scope was to connect various networks at academic institutions, military installations, government agencies, commercial enterprises and Internet support companies together to share any type of information, data and services (Long & Long, 2005). The creation of the World Wide Web which is the biggest and most known application of the Internet today (Long & Long, 2005), accelerated the expansion of the Internet and subsequently with the increase in popularity of the World Wide Web initiated the creation of the HTML standard in the late 1990 (National Academy of Engineering, 2011). Ever since the HTML standard has been under development, it evolved from a simple Markup Language to display text, to including images, and adding CSS to format text. However in 1997, the introduction of HTML4 marked the slowdown of the evolution of HTML as a language because of some major factors, specifically the lack of computer processing power and the limited speed of the web (David, 2010).

The Present

A few years have passed, and things have changed, because today computers are amazingly fast and powerful. Mobile devices, such as smartphones, have become very powerful and very fast while keeping the size of such devices small and portable for people to easily carry such electronic devices around with them all the time. Apart from smartphone devices there are other mobile devices such as tablets and netbooks, some of which are becoming very popular nowadays, especially tablets. The introduction of the iPhone in 2007 induced the increase in the production and sales of smartphones, resulting in the increase of users browsing the web on their mobile device which is gradually gaining popularity over
desktop web browsing. In fact, mobile web browsing has increased from 0.86% in May 2009 to 5.21% in last April 2011 (StatCounter Global Stats) which is a significant increase in the last 2 years. Furthermore internet download speed has increased drastically and now we can reach a maximum average speed of 37.9Mbps (Belson, 2010) which provides a faster and smoother experience when using the Internet, and especially when using the World Wide Web. This means that the limitations and challenges that the HTML4 standard faced years ago have been surpassed by the advancements of today’s technology. Nowadays HTML4 is not up to the task for the modern Web anymore, and the fact is that a new Web standard is needed (David, 2010).

Hence HTML5 was conceived by the Web Hypertext Application Technology Working Group in 2004 (Keith, 2010) where the HTML standard was being modified to be capable of running full applications in a Web browser. With HTML5, new standards and new revolutionary technologies such as Geo-Location, Offline Storage and Offline Web applications brought new possibilities for the Web. Furthermore, the constant implementation and improvement of Web standards in modern Web browsers provide the tools and means to build and run a full application in a Web browser.

**Top Market Players**

With the increase in popularity of the mobile device market, enterprises and businesses are likely to invest and develop a mobile application sooner or later. At the present time there are currently two popular operating systems which come installed on most smartphone and tablet mobile devices, which are iOS developed by Apple and Android that is developed by Google with a US market share of 31% and 49% respectively (Canalys, 2011). Both companies offer their unique development tools to create applications for their respective mobile operating systems; Apple offers the Xcode IDE which enables developers
to code Mac and iOS applications using the Objective-C coding language (Developer Tools, 2011), whilst Google offers the Android SDK which include a variety of tools to help develop and debug an application for the Android platform (Tools, 2011). As a fact an application created for an Android device cannot be installed on an iOS device because of the different development language and also a different API library, which means that two applications must to be created for two different operating systems.

**What is HTML5 up to?**

After nearly 7 years of development of the HTML5 standard along with the subsequent improvements in JavaScript and CSS, can HTML5, JavaScript and CSS3 be considered feasible and sufficient to develop a multi-platform mobile web application in contrast to developing multiple native versions of the same application? Is it less time consuming to develop one multi-platform web application compared to creating multiple versions of the same application for various platforms? These are the key questions that this thesis will examine, and in order to answer these questions, an existing Windows 7 desktop invoicing application will be used as a model to create an HTML5 application using HTML5, CSS3 and JavaScript.

Only one HTML5 application will be created, which should then be able to run on multiple devices, each with different operating systems and different screen resolutions. The HTML5 application will be created using one of the mobile Web application frameworks available after evaluating its feasibility and practicality according to the planned application. The Web application will implement some of the main functionalities of the invoicing application used as a model, such as creating an invoice and adding new items. After the completion of the application, it will be tested on a set of mobile devices to verify the functionality and the usability of the application. While running the tests on the application,
the results will be recorded for analysis purposes. The outcome of the analysis of the results will show if the current state of these Web technologies is at an advanced stage, or if more development needs to be done before such technologies can be considered as a valid alternative to native mobile applications.
LITERATURE REVIEW

Definition of a Native and a Web Application

The Global Intelligence Alliance (2010) defines a Native application within the mobile context as “An application specifically designed to run on a device’s operating system and machine firmware. It typically needs to be adapted/adjusted for different devices”, whilst the definition Global Intelligence Alliance presents for a Web application within a mobile context is “An application in which all or some parts of the software are downloaded from the Web each time it is run. It can usually be accessed from all web-capable mobile devices.” Therefore a mobile Native application is an application that can only run on the mobile platform it was developed for and to deploy or distribute the same application on another mobile device with a different platform it has to be re-written. On the other hand a mobile Web application is accessible from all the mobile devices capable of connecting to the internet and can access the web.

The HTML5 Specification

HTML5 as defined by the World Wide Web Consortium specification is “the 5th major revision of the core language of the World Wide Web: the Hypertext Markup Language (HTML).” The HTML5 specification will offer new and better tools for developers because “In this version, new features are introduced to help Web application authors, new elements are introduced based on research into prevailing authoring practices, and special attention has been given to defining clear conformance criteria for user agents in an effort to improve interoperability.” (Hickson, 2011)
In other words HTML5 will help to keep the HTML syntax clean, simple and easily understandable for web developers, rendering engines and not only. The new version of HTML will help create a multitude of Web applications and will make it easier to target mobile devices with the new tags that were introduced to cater for the mobile market and provide a better experience to the mobile users. HTML was primarily designed as a language for semantically describing scientific documents although its general design and adaptations over the years have enabled it to be used to describe a number of other types of documents (Hickson, 2011). The main area that has not been addressed by HTML is the creation of Web Applications, and the scope of HTML5 according to the W3C specification is to improve the development of these applications while also fixing some of the issues raised within the past few years.

**HTML5 Features**

The new HTML specification introduces several new features such as canvas, video and geo-location, underneath are some details about the several new features introduced by the HTML5 specification.

**Canvas**, is “a resolution-dependent bitmap canvas, which can be used for rendering graphs, game graphics, or other visual images on the fly” (Hickson, 2011). “A canvas is a rectangle in your page within which you can use JavaScript to draw anything you want” (Pilgrim, 2010). By using the new canvas API in HTML5 one can draw shapes, define paths, create gradients, and apply transformations.

The **Video** element is used for playing videos or movies in web pages (Hickson, 2011). “Embedding video used to be impossible without third-party plug-ins such as Apple QuickTime or Adobe Flash” (Pilgrim, 2010). “Plugins can also be a significant cause of
browser instability and can create worry in less technical users when they are prompted to download and install newer versions.” (Lawson & Sharp, 2011)

Storage – There are two storage options provided by the HTML5 specification which are currently implemented in some of the major browsers. The first option is Web Storage which usually has an initial limit of 5Mb of storage space available on the client side and is normally used to store a key with a value. A Web Storage is like the data stored in cookies, but unlike with cookies, this data is never transmitted to the remote web server (Pilgrim, 2010).

The Web Storage API is also subdivided into two other types, the sessionStorage which is used to store data for the current open window only, and then there is the localStorage which is based around a domain storage option and will work across all the windows which are browsing the same domain from the same browser (Lawson & Sharp, 2011). localStorage is persistent, therefore if one reboots his PC the data of a particular domain which has been stored locally in the localStorage will still be available. Since both sessionStorage and localStorage derive from the same API they do not differ in the code used making it simpler for developers to code.

The second option to store data locally using HTML5 is to use the Web SQL Storage which is a wrapper of an SQLite (3.1.19) database (Sharp, 2010). Similarly to the Web Storage, it has a default limit of 5Mb, but can be set to a higher size during the creation of the database itself via JavaScript. Web SQL as it is more known (Oehlman & Blanc, 2011) has three main methods which are:-

- **openDatabase**, used to open/create a database;
- **transaction/readTransaction**, used to open a transaction for the execution of one or more SQL statements;
• **executeSql**, used to run an SQL statement within a transaction.

Although Web SQL has been implemented in most WebKit browsers the work on the specification itself has stopped, in fact on the W3C specification page there is highlighted the following: “Beware. This specification is no longer in active maintenance and the Web Applications Working Group does not intend to maintain it further” (Hickson, 2011). This happened because according to Mozilla “there isn’t a credible, widely accepted standard that subsets SQL in a useful way” (Ranganathan, 2010), since Web SQL is based on the SQLite specification, and continues on to write that “IndexedDB does not have this problem; even though our underlying implementation of IndexedDB may be based on SQLite, we keep developers insulated from changes to SQLite by exposing an API that isn’t based on SQLite’s supported syntax” (Ranganathan, 2010). Currently the implementation of the IndexedDB specification is only found in the latest Firefox 4.0 release and in the latest Developer versions of Chrome.

**Offline Web Applications** — “At its simplest, an offline web application is just a list of URLs pointing to HTML, CSS, or JavaScript files, images, or any other kinds of resources that may be present” (Pilgrim, 2010). Offline web applications make use of a manifest file stored on the web server of the website (normally in the root directory) which points to the web application resources. “A web browser that implements HTML5 offline applications will read the list of URLs from the manifest file, download the resources, cache them locally, and automatically keep the local copies up to date as they change” (Pilgrim, 2010). Then according to Oehlman and Blanc (2011) “The next time the user navigates to your application, the browser will use its cache instead of retrieving the files from the server, regardless of the connection status.”
The W3C Geolocation specification provides a mechanism for the browser to query the device’s location irrespective of the specific location technology being used (Melamed & Clayton, 2009). As this technology is pervasive to protect the privacy of the user and the user itself, the specification says that “User agents must not send location information to websites without the express permission of the user” (Lawson & Sharp, 2011).

**Input Types** - The HTML5 specification introduced new form types and also built-in validation. New form fields such as the input of type email which accepts only a valid email address, input of type date which could show a calendar according to how the browser handles the control, and input of type tel which accepts a telephone number are only some of the new input types available in HTML5. The new controls facilitate the developers’ work and integrate nicely with mobile devices. For example on an Android or an iPhone the input field of type tel will show the telephone number input keyboard instead of the usual QWERTY keyboard; the result of this action enhances the user interaction with web applications (Lawson & Sharp, 2011).

**JavaScript scripting language**

JavaScript is “… a lightweight, object-oriented language, most known as the scripting language for web pages, but [has been] used in many non-browser environments as well” (JavaScript, 2011), this is the definition for JavaScript written on the website of the Mozilla Developer Network. It states that JavaScript is not only used for web pages but also in many “non-browser environments as well” (JavaScript, 2011); a clear example is Palm’s mobile webOS development platform that lets “… developers use common skills such as JavaScript, CSS and HTML to quickly create, convert and distribute sophisticated mobile device applications” (Gittlen, 2010). According to Palm these applications can then “… be deployed across a host of popular devices such as the Palm Pre and Palm Pixi smartphones” (Gittlen,
2010), apart from Palm’s development platform JavaScript is also used “… in Adobe Acrobat, Photoshop, Yahoo!’s Widget engine, and even server side environments”, which clearly shows the power and flexibility of the JavaScript scripting language (A re-introduction to JavaScript, 2011).

**Designing an Application with CSS3**

The importance of CSS3 for mobile Web applications resides in the fact that it helps to create the “fastest and lightest experience possible”, and is a “… wonderful tool for creating complex designs using the minimum [number] of images, making it ideal for mobile design” (Fling, Mobile Design and Development, 2009). CSS3 helps in laying out the interface of the application and to provide the best user experience possible while keeping the application layout user friendly and simple. Since the CSS3 specification is not yet finalised CSS3 definitions are implemented using proprietary prefixes such as –webkit which would be dropped in favour of the official standard after the CSS3 specification is finalised (Oehlman & Blanc, 2011), but quoting The New York Times (2011) “Engineers say the technology will make it possible to write Web applications, accessed with a browser, that are as visually rich and lively as the so-called native applications that are now designed to run on a specific device, like an iPad or an Android-based tablet.” Barbara Ballard (2007) explains that “Media, including graphics, animation, video, and sound, is perceived and consumed differently on mobile devices”, in fact Oehlman and Blanc (2011) write that “technologies like HTML5, CSS3, and JavaScript can now offer solutions that compete with native technologies”, they also add that “CSS3, adds a lot of improvements, including shadows, rotations, and gradients.”
Device Fragmentation

The provision of well documented APIs, the application store of each platform from where one can buy applications for the mobile device, and the fact “… that you can actually charge people for the app”, gives an advantage to Native mobile applications. On the other hand Fling (2009) continues on writing that “Native applications, however, do not solve the greatest problem in mobile, from its inception to today: device fragmentation. Creating applications for multiple platforms is extremely difficult and costly to do.” As a result of the fact that device fragmentation exists, and is one of the biggest problems in mobile devices, the costs of developing a multi-platform mobile application is the main problem that companies need to face.

The problem of device fragmentation is furthermore sustained by researchers at Hewlett-Packard, were they argue that “Device fragmentation continues to be a problem for developers with many different phone operating systems, and programming APIs” (Melamed & Clayton, 2009). Melamed and Clayton (2009) also state that, “The web browser is rapidly evolving from a renderer of simple html into a runtime environment capable of delivering rich interactive applications across many application domains”, especially due to the introduction of the new HTML5 specification by the World Wide Web Consortium (W3C). Milikich (2010) says that, “We aren’t yet at the point where Web apps can be write once, run anywhere”, but he is positive that by “Coupling HTML5 with the current momentum, we can expect to see more standards based features and convergence in the future.”

Browser Support

According to Milikich (2010) “Browsers are undergoing a sea change to better support Web apps and mobile”, nevertheless former situations are still repeating themselves for instance when the “… past quantum changes in the Web [occurred,] such as the migration
to AJAX and Web 2.0” (Milikich, 2010). In fact Milikich (2010) argues that “… there are some platforms that are ahead of others, resulting in fragmentation and inconsistencies”, but this time the fragmentation is referring to the browsers’ rendering engine that differs from each browser. However this time “there are some important differences” (Milikich, 2010), because WebKit is perceived as the pioneer that is improving the web browsing experience by improving the conformity to web standards. Besides the potential monetisation from developers who develop web applications, and likewise the growth in the number of end users using mobile devices with web browsers which “…wasn’t present during previous quantum changes in the Web” (Milikich, 2010) is contributing to the success and faster implementation of this new web technology.

There are those who debate the reliability of mobile Web applications, and state that it’s “Unpredictable, Slow and Disappointing” (Kony Solutions, 2009). The reason for this statement is because, during high traffic, or when using a poor internet connection, the performance and usability of the application is affected. Kony Solutions (2009) also write that, Web applications make “limited use of the device’s native functionality” such as notifications and alerts, and in many cases users would have to minimise the application to make use of certain features. Alternatively Native applications are recommend as they are “Always Available, Fast and Feature-Rich”, and their reason is “Because a native on-device application is stored entirely on the actual device, its UI performance is unaffected by network congestion or server traffic issues” (Kony Solutions, 2009). Hence the fact that a Web application depends on internet connectivity and speed does not guarantee the best user experience and may also result in the refutation to use the application. This is not the case for Native applications as the application is deployed locally on the mobile device and is not affected by network connectivity unless it is designed to connect to the internet to provide its service to the user.
Some try to answer the question “When should you make a mobile web application versus a native mobile application, or a downloadable application designed for a specific platform, like a Java application or an iPhone?” and the decision normally comes down to the money or budget available (Fling, Mobile Design and Development, 2009). “Native applications can produce a lot of short-term revenue for developers and operators alike, especially with the proliferation of app stores” (Fling, Mobile Design and Development, 2009). App stores make a very important part of Native Applications, as they are the main drive of income for companies and developers since they provide a wide access to the platform’s market. Nevertheless “… the mobile web is the only long-term commercially viable content platform for mobile devices” and four significant keys which contribute to this belief are fragmentation, the Web, control and consumer expectations (Fling, Mobile Design and Development, 2009).

**Mobile Computing**

The mobile world is very different from the desktop computing, and there is no economically feasible means to create a native application that can support the majority of the market. Indeed the difference lies in the fact that “The mobile JavaScript standard, EcmaScript Mobile Profile, allows proprietary API extensions, so expect to encounter fragmentation and inconsistent event models in mobile browser implementations” (Frederick & Lal, 2009), which clearly indicates that each part of a Web Application is susceptible to the device and the compatibility with the platform’s operating system or software running on the device itself. While the concept of API fragmentation endures, this will require “browser detection and conditional code to include only the JavaScript supported on the mobile device” (Frederick & Lal, 2009), but JavaScript is well implemented in smartphones and newer mobiles with Internet capability which decreases inconsistencies between mobile devices.
Mobile application vendors always have to rely on middle-men to get their products to the market, which affects the control of the distribution of a mobile application after its completion. This was previously under the control of the operator, but nowadays it has shifted to the device and the platform makers, which make it easier to market applications (Fling, Mobile Design and Development, 2009). In fact the Apple’s App Store, the official application store for the iPod Touch, iPhone and iPad, is a clear example of innovation where it is helping to reduce the barriers to application development by providing easy access to distribution. Google’s App Market, Blackberry App World and Windows Marketplace for mobile, are likely to drive the success of existing applications while also driving new developers to their platform (Allen, Graupera, & Lundrigan, 2010).

**Existing HTML5 Mobile Web Applications**

Thomas Schranz discusses the potential of developing HTML5 web applications in contrast to learning Objective-C to develop an application for the iOS platform, or Java to create applications for the Android platform (Schranz, 2010). Schranz (2010) compares and reviews some of the HTML5 Web applications that are also implemented as a native application, and presents screenshots of the applications for a visual comparison. Thomas Schranz (2010) also he discusses that “*With more performant devices, faster JavaScript engines and hardware accelerated graphics it is just a matter of time until virtually all of them could be created as cross platform html5 applications.*” Below are some screenshots provided by Thomas Schranz, which show some of the HTML5 applications he reviewed, namely the Facebook application, Google’s Gmail application and Nextstop.
Multi-platform mobile application development using HTML5, JavaScript and CSS3.
Showtime (Showtime) “is a web app for the iPhone and iPod Touch which keeps track of your favourite TV shows and lets you know when new episodes are available to watch”; the developer used jQTouch plugin to develop the application.
The Future of mobile applications

Some agree that the market is going to the web as it offers many advantages and only a few disadvantages. According to Oehlman and Blanc (2011) “The era of mobile computing is upon us. The adoption of web-connected mobile devices is one of the fastest-moving trends worldwide. Consumers have become accustomed to mobile technology, and for most of us that now includes mobile web access.” The perception is that since mobile browsers are adding more services to detect the location, acceleration, or use of hardware, this will reduce the need for native mobile application. The necessity to build a native application will be specific to those applications that will need the full utilisation of the mobile device such as games. Furthermore there are those who discuss that the way we develop for the web is influencing emerging cross-platform techniques for mobile (Fling, Mobile Design and Development, 2009). They argue that before any cross-platform frameworks existed, namely Rhodes, PhoneGap and Titanium Mobile, developers embedded Web UI in a native application, which was a practical way to develop mobile applications quickly and make cross-platform applications easier to maintain. (Allen, Graupera, & Lundrigan, 2010)
METHODOLOGY

Features and Functionality of the Web Application

The application mimicked a small percentage of an existing invoicing application called Faktum Invoicing, which was used as a model to develop and implement the Web application. The features that were implemented are the following:

1. Creation of a new invoice;
2. Deleting an invoice;
3. Adding items (name or description, quantity, price) to an invoice;
4. Deleting items from an invoice;
5. Editing items in an invoice;
6. Edit the discount percentage on the net price;
7. Edit the Vat percentage on the subtotal price;
8. Add new clients;
9. View a report of the total monthly sales.

Functionality was kept as easy as possible, since the application was also targeted for smartphones that did not have a large screen space. In fact, an intuitive design was used in order to make full use of the whole screen, while at the same time the application had a native look and feel. Confirmation messages were shown whenever items were deleted or all the data was going to be cleared from the database, so that the user would not delete any information by mistake. On small screens the user would be more prone to press another button by mistake due to the limited space compared to tablets, and other large display mobile devices. At the top a toolbar was placed with a menu button on the right hand side of the toolbar, which when clicked a submenu would display with more buttons that add more...
functionality to the application. On the left hand side of the toolbar there would be either the back button to navigate back to the previous section, or a “Clear Data” button would be visible to clear all the data of the application and have a fresh new start.

Frameworks

A few months ago there were very few mobile frameworks which could be used to create an HTML5 application, and the relative young and immature state of the frameworks did not guarantee a complete multi-platform support. Though, in a few months, there has been a great advancement in multi-platform mobile frameworks. The number of frameworks has more than doubled, plus different mashups which combined different frameworks and plugins were created. In fact, the framework I chose to develop and implement my mobile multi-platform application is named AppMl (Scarcello & Allori), which combined the power of the jQTouch, iScroll and jQuery. This framework provided a combination of the most powerful and mature plugins, and frameworks, which ultimately should increase the compatibility with various mobile devices.

The choice to use a framework instead of creating a whole new application from scratch, offered some benefits, as the application would look more native and some of the native functionality, which normally one would expect, would be difficult and time consuming to create from scratch. Some of the important features and functionality provided by the AppMl framework are the following:

1. Add to home button, which add a floating balloon (for iPhone, iPod and iPad), and enable the user to “install” the web application as a bookmark on the home screen;
2. Switchable styles, enable the developer to switch between two pre-defined styles, and also separates the design from the code so that in a larger team designers and developers can work separately without any conflicts;

3. Automatic generation of toolbars, navigation bar and sidebar;

4. Page transitions, to provide a more intense user experience.

5. History Management, which prevent pages being added to the browser’s history which would affect the functionality of the device’s back button.

Other prominent frameworks that were very popular for the development of multi-platform HTML5 mobile applications, were jQuery Mobile (The jQuery Project, 2010) which combined the jQuery (The jQuery Project, 2010) and the jQuery UI (The jQuery Project and the jQuery UI Team, 2010) frameworks, Sencha Touch (Sencha Touch: Mobile JavaScript Framework, 2011) mobile JavaScript framework, Unify (Unify Project, 2010) and the Dojo Toolkit (The Dojo Foundation). All these frameworks made use of HTML5, CSS3 and JavaScript code in order to provide the tools and means to create a native looking mobile web application that worked on the most popular mobile devices such as smartphones. There were also other tools to create a multi-platform mobile application using the Web technologies, such as PhoneGap, and Titanium Appcelerator, but the different characteristic which made these tools unique from other development frameworks was, that they provided access to the device’s native functionality that was also available for native application developers, via their own API. Therefore, developers wrote one application using HTML5, CSS3 and/or Javascript, with some additional code that targeted different systems. The tools provided by PhoneGap, and Titanium Appcelerator, wrapped the Web application which produced a native application that could then be re-distributed as a normal native application on multiple devices.
Structure

The structure of the application focused on the implementation requirements that were required by the AppML framework that was used to develop the application. The framework consisted of the following file structure, which was modified and customised to cater for the invoicing Web application:

- **css** – a folder used to store the framework CSS and images
- **js** – a folder used to store the framework’s JavaScript files including the JQuery framework and other plugins used by the application
- **themes** – a folder which is used to store the default and custom themes that can be used to style the application
- **loading.gif** – used as a loading screen until the application has completed loading the content and appropriate files
- **favicon.ico** – this file is used by web browsers
- **index_Tablet.html** – if a tablet such as the iPad is detected as the device the application redirects to use this file
- **index_Phone.html** – if a smartphone such as the iPhone or an Android device is detected the web application will use this file
- **Large_icon.png** – a higher resolution icon that is used by tablets
- **Small_icon.png** – a smaller resolution icon that is used by smartphones
- **cache.manifest** – this is the manifest file that will indicate which files need to be cached to load and run the application when there is no internet connection

The AppML framework provided the necessary files to build a mobile Web application and since the whole application was implemented using HTML5, CSS3 and JavaScript, and due to the way the framework created the various application views dynamically, one only...
needed to make use of a single .html file and multiple JavaScript files to organise the application to be more object oriented.

**Multi-Platform Support**

HTML5 was a relatively new technology, and implementation for such technology in mobile and desktop browsers started just over 2 years ago (Lawson & Sharp, 2011). There was still way to go before the complete HTML5 specification was finalised (Is HTML5 Ready Yet) and fully supported (When can I use..., 2011). The same could be said for the CSS3 specification, which was still yet not fully supported by all the browsers, but the current status, of both specifications was enough to create a multi-platform mobile Web application.

Due to the fact that the CSS3 and HTML5 specifications were still not a standard, different browsers made use of different prefixes and different tags to increase the support of the specifications. An example was that prefixes such as "-webkit", "-moz-" and "-o-" (4 Syntax and basic data types, 2011) were used to implement the specification for WebKit, Mozilla and Opera browsers respectively, but were not supported by other browsers since they were proprietary tags. This forced developers to use all prefixes in order to support all browsers at their own disadvantage, because there was more code to write and sometimes the code was different for the different browsers, which complicated the development of the application.

HTML5 had the same issues with the mobile Meta tags, where proprietary tags had been created to support mobile web applications. For instance, there was a Meta tag to specify the application icon, which was displayed once the application was added as a bookmark and a shortcut had been created on the home screen of the device. There was also a Meta tag to specify the viewport properties of the application, such as the size and fit of the application within the browser. Likewise, there existed other specific tags that had been specifically
created for different mobile browsers. Below is an example of an Apple’s proprietary tag for
the mobile Safari browser (Apple, 2011):

    <meta name="apple-mobile-web-app-capable" content="yes"/>

Data Storage

WebKit mobile browsers totalled to 55% of the market share (2010 mobile browser
stats), as presented on the StatCounter chart (Top 9 Mobile Browsers on May 11, 2011), and
supported most of the new HTML5 specification. Most importantly mobile browsers
supported the HTML5 WebSQL Storage, which was used to store the application’s data in a
persistent manner (When can I use Web SQL Database?, 2011). Apart from the WebSQL
database, mobile WebKit browsers also supported the LocalStorage, and SessionStorage
features of HTML5, which were used to “permanently” or temporarily store application’s
data.

Device Hardware Access

At the time of writing, access to the hardware of mobile devices from Web
applications was very limited, namely the API gave access to the device’s orientation and
location, but it was not possible to access the button events or other hardware features.
Therefore, such functionality needed to be re-thought specifically for the Web application, to
keep the application similar to a native application as much as possible. The invoicing Web
application, that was created to demonstrate the potential of mobile web applications
compared to native mobile applications, did not need or made use of any other hardware
components, such as the device’s camera, the accelerometer, or Bluetooth. Besides, if one
would have needed to make use of the device’s hardware features, there were other
frameworks that could have been used in combination or as an alternative, such as PhoneGap
(PhoneGap) or the more JavaScript oriented Titanium Appcelerator (Appcelerator).
Development Tools and Testing

No special tools were needed to develop the mobile Web application. In fact the only tools that were used to develop the application were a WebKit browser, namely Chrome and a text editor like Notepad++. The benefits of using the Chrome browser were that its default debugging tools were enough to test and analyse the application’s code. Also the Chrome browser supported WebSQL that was used as a persistent storage for the Web application created. The Notepad++ tool, due to its syntax highlighting feature, helped to highlight various codes such as HTML, JavaScript or CSS which facilitated the development of the Web application.

Since the multi-platform mobile application was targeted to the majority of the mobile devices on the market that supported HTML5, CSS3 and JavaScript, the application had to be tested on the actual hardware to determine if it was working as it should. For this purpose an Android smartphone, a netbook with Windows 7 installed and an iPad were used to test the functionality, design and feel of the Web application.

A series of pre-defined tests were conducted on each device to evaluate the status of the Web application with respect to the design of the application, the functionality and also the similarity to native applications. The checklist that was compiled for grading the Web application helped to determine the success or failure of the web application with regards to the support of HTML5 features, CSS3 tags and also JavaScript code.
<table>
<thead>
<tr>
<th>Multi-platform Web Application Checklist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Web Application loads without errors?</td>
</tr>
<tr>
<td>2. Web Application fits within the device’s screen resolution?</td>
</tr>
<tr>
<td>3. Design displays correctly in the main device browser?</td>
</tr>
<tr>
<td>4. Application displays correctly in both orientations?</td>
</tr>
<tr>
<td>5. Buttons are clickable and redirect correctly?</td>
</tr>
<tr>
<td>6. Application is always loaded in one page? (i.e. the same URL is loaded in the address bar)</td>
</tr>
<tr>
<td>7. Menu is shown when clicked?</td>
</tr>
<tr>
<td>8. New invoice item added successfully?</td>
</tr>
<tr>
<td>9. Edit invoice item saved successfully?</td>
</tr>
<tr>
<td>10. Clients can be added?</td>
</tr>
<tr>
<td>11. Clicking on client phone button loads up the device’s caller (when available)?</td>
</tr>
<tr>
<td>12. Report loads fine?</td>
</tr>
<tr>
<td>13. Web application looks and feels like other native applications?</td>
</tr>
<tr>
<td>14. Web application works in offline mode?</td>
</tr>
<tr>
<td>15. Passes user usability test?</td>
</tr>
</tbody>
</table>
Apart from the pre-defined checklist the application was also tested using the official W3C mobileOK Checker (W3C mobileOK Checker, 2010). The W3C mobileOK Cecker performed a series of tests on the pages of the Web application, and determined its level of mobile friendliness by checking if a set of criteria defined by the Mobile Web Best Practices: Basic Guidelines (Mobile Web Best Practices 1.0, 2008), that was issued by the World Wide Web Consortium 3 years ago, were met. Since the W3C checker had not been updated since “2010-09-03” (W3C, 2010), it meant that some of the latest changes made to the draft in the last 8 months before the final call for changes in May 2011 might not be implemented in the online checker. Therefore the results of this test were considered accordingly, and the application was not expected to pass at 100%.

Testing JavaScript on the actual mobile device was critical for the effective development of the Web application. If the tests were to be performed in desktop browsers and in device emulators only, any syntax problems and performance issues that would occur on the target mobile device would not be uncovered (Frederick & Lal, 2009). Although the desktop and mobile version of JavaScript had nearly the same syntax, there were some limitations in the support of DOM events, reduced character sets and the exclusion of computationally intensive language elements.

**Determining if Multi-platform Web Applications are possible**

This research answered the following questions, “Are HTML5, JavaScript and CSS3 sufficient to develop a multi-platform mobile application compared to a native mobile application?”, and “Is it less time consuming to develop a web application for multiple mobile devices than developing several native applications?” To answer these questions, the checklist criteria needed to be met in order for the application to be further evaluated and compared to a native mobile application. All the tests conducted on the different mobile
devices needed to be equally passed, for instance, if the design test on the iPad failed but on the Android smartphone the test was passed successfully, it meant that work still had to be done on the iPad version to fix the design issues. This showed that more time would still be consumed to fix bugs, and therefore the development stage would take longer. The analysis of the final results would also be important in order to answer the questions posed, as the outcomes sowed the level of support for the HTML5 features, especially for offline Web application caching, native design look, and device functionality integration throughout all the devices.
RESULTS AND ANALYSIS

Tests conducted

Tests were conducted to verify the functionality, the usability and design fidelity of the mobile web application on various mobile devices. The checklist was used to determine important design and functionality features that the application had to perform correctly on each mobile device where the application was tested. This was done in order to be able to compare and point out certain weaknesses or possible bugs in the software which would thereafter need to be fixed.

Apart from the pre-defined checklist the application was also tested using the official (W3C mobileOK Checker, 2010) W3C mobileOK Checker, and what this checker does is performs a series of tests on a given web page and determines its level of mobile friendliness by checking if certain criteria has been met according to the (Mobile Web Best Practices 1.0, 2008) Mobile Web Best Practices: Basic Guidelines issued by the World Wide Web Consortium 3 years ago.

Test Results and Analysis

Tests were conducted on the Android smartphone device, which was an HTC Desire that ran Android version 2.3.3 and had a browser rendering engine installed, versioned WebKit/533.1. Although the mobile Web application passed most of the checklist requirements, it did not mean that the application was near perfection because as one might notice, the most important checklist items failed during the tests conducted for several reasons.

First of all the Android OS version 2.3.3 installed on the HTC Desire did not support full screen Web application view, which meant that the application had to run only in browser.
view. The address bar at the top was always visible in addition to the top Android bar, which showed the battery level and other notification alerts. There was also another problem with the input fields that were generated by the AppML framework, and how they were rendered by the default Android browser, which consequently rendered the usability of the application pretty difficult. The problem was that, since the AppML framework made use of the iScroll plugin to make the page content scrollable similar to the native applications, the developers of the framework needed to generate an invisible “<div>” tag to wrap the input component. This workaround was done in order to avoid conflicts with the iScroll plugin and kept the scrolling of the page content smooth. However, the different rendering of the HTML and CSS code in this particular mobile browser version, the input fields did not work all the time. For this reason the application did not pass the usability test on this device.

Another issue that was encountered during the tests conducted on the HTML5 Web application while using the Android device was that, when the device’s orientation changed the application design did not scale correctly. This bug made the whole Web application nearly unusable, and the user would need to refresh the whole application in order to be able to use the application.
### Multi-platform Web Application Checklist

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>16. Web Application loads without errors?</td>
<td>✔</td>
</tr>
<tr>
<td>17. Web Application fits within the device’s screen resolution?</td>
<td>✔</td>
</tr>
<tr>
<td>18. Design displays correctly in the main device browser?</td>
<td>✔</td>
</tr>
<tr>
<td>19. Application displays correctly in both orientations?</td>
<td>✗</td>
</tr>
<tr>
<td>20. Buttons are clickable and redirect correctly?</td>
<td>✔</td>
</tr>
<tr>
<td>21. Application is always loaded in one page? (i.e. the same URL is loaded in the address bar)</td>
<td>✔</td>
</tr>
<tr>
<td>22. Menu is shown when clicked?</td>
<td>✔</td>
</tr>
<tr>
<td>23. New invoice item added successfully?</td>
<td>✔</td>
</tr>
<tr>
<td>24. Edit invoice item saved successfully?</td>
<td>✔</td>
</tr>
<tr>
<td>25. Clients can be added?</td>
<td>✔</td>
</tr>
<tr>
<td>26. Clicking on client phone button loads up the device’s caller (when available)?</td>
<td>✔</td>
</tr>
<tr>
<td>27. Report loads fine?</td>
<td>✔</td>
</tr>
<tr>
<td>28. Web application looks and feels like other native applications?</td>
<td>✗</td>
</tr>
<tr>
<td>29. Web application works in offline mode?</td>
<td>✔</td>
</tr>
<tr>
<td>30. Passes user usability test?</td>
<td>✗</td>
</tr>
</tbody>
</table>
Tests were also conducted on an Apple iPad, with iOS version 4.3.3 using the default WebKit browser on the device. The results on the Apple iPad device were different compared to the HTC Desire. Whilst on the Android smartphone the application had usability, orientation, and native looking issues, Apple’s tablet had issues to draw up the report chart, which on the other hand displayed correctly on the smartphone device.

Another serious issue that occurred while the application was being tested on the particular iPad device was that, support for offline web applications did not work whilst the same code was working on the Android smartphone. This was a major flaw for the HTML5 Web application as one of the objectives of the application was to work also when there was no internet available. In fact there seemed to be some mixed results, where some developers succeeded to make their HTML5 Web application work in offline mode on the iPad, whilst others still struggled to make the “cache.manifest” file work with their Web application.
## TABLE 3 - TEST RESULTS ON THE APPLE IPAD

<table>
<thead>
<tr>
<th>Multi-platform Web Application Checklist</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Web Application loads without errors?</td>
<td>✔</td>
</tr>
<tr>
<td>2. Web Application fits within the device’s screen resolution?</td>
<td>✔</td>
</tr>
<tr>
<td>3. Design displays correctly in the main device browser?</td>
<td>✔</td>
</tr>
<tr>
<td>4. Application displays correctly in both orientations?</td>
<td>✔</td>
</tr>
<tr>
<td>5. Buttons are clickable and redirect correctly?</td>
<td>✔</td>
</tr>
<tr>
<td>6. Application is always loaded in one page? (i.e. the same URL is loaded in the address bar)</td>
<td>✔</td>
</tr>
<tr>
<td>7. Menu is shown when clicked?</td>
<td>✔</td>
</tr>
<tr>
<td>8. New invoice item added successfully?</td>
<td>✔</td>
</tr>
<tr>
<td>9. Edit invoice item saved successfully?</td>
<td>✔</td>
</tr>
<tr>
<td>10. Clients can be added?</td>
<td>✔</td>
</tr>
<tr>
<td>11. Clicking on client phone button loads up the device’s caller (when available)?</td>
<td>✔</td>
</tr>
<tr>
<td>12. Report loads fine?</td>
<td>☒</td>
</tr>
<tr>
<td>13. Web application looks and feels like other native applications?</td>
<td>✔</td>
</tr>
<tr>
<td>14. Web application works in offline mode?</td>
<td>☒</td>
</tr>
<tr>
<td>15. Passes user usability test?</td>
<td>✔</td>
</tr>
</tbody>
</table>
Tests were conducted on an Acer netbook too, which had a Windows 7 operating system installed and made use of the default Internet Explorer 9 Web browser. Unfortunately, Internet Explorer 9 did not support most of the HTML5 features, such as WebSQL databases, and offline caching amongst others, so the application was not able to run using the default browser supplied with the Windows 7 operating system. Additional options were available for Windows users, as the Google Chrome and Apple’s Safari browsers were both available for download on the operating system. Since both browsers were built on the WebKit engine, they equally supported WebSQL that was used to store data on the user’s personal computer when there was no internet connection. Apart from that, other HTML5 features were also supported, but this implied that the user had to download another Web browser to use the application, and this could have presented a difficulty or an impediment for those users that were not highly computer literate.

Another issue that was noticed on the netbook device with Windows 7 was that, the HTML5 Web application could only be run within a Web browser. This meant that the Web browser’s frame was still visible, unless the browser was set to full-screen mode by the user. This also meant that the Web application lacked the native look and feel when compared to other native Windows 7 applications.

**Testing the application using the W3C mobileOk Checker**

The HTML5 web application was not only tested against the pre-defined checklist but was also tested using the online W3C’s mobile validity checker. The aim of the W3C mobileOk checker was to help check the level of mobile friendliness of web documents, whilst it also asserted that the Web document was mobileOk. The results obtained from the online checker showed if any severe issues had been detected in the HTML or CSS code implemented, and how they could be improved according to the current web standards.
When the results were examined to check for any potential issues, one needed to keep in mind that the Web standards were not yet been fully implemented in all Web browsers. In fact some proprietary prefixes had to be used in the CSS code implemented in order to cater for the majority of the mobile devices. Furthermore, the whole Web application was contained into one single HTML Web document, which resulted in an expected large download, since all the images, CSS and JavaScript needed to be downloaded the first time the application was loaded. However this was mitigated with the offline cache manifest file, which downloaded all the specified resources in the file on the client’s device. In this way the load time of the Web application was decreased, because the offline cached files were used to load the application. Only in the exception that the manifest file was updated, that is a new version of the application was uploaded, all the files needed to be re-downloaded.

A total of 9 issues were raised by the checker, 1 was marked as severe, and another 3 were marked as medium level issues, whilst 5 issues were marked as low level issues. Some of the important issues that were pointed out by the W3C checker are listed below to show the nature of these errors.
1. Syntactically invalid CSS code. As explained before specific CSS code was used to target different Web browsers, and if more mobile devices were available to conduct tests, design bugs would have been listed and tackled accordingly.

2. No Caching and total page size exceeds 20 kilobytes limit.

Other small issues were also pointed out by the W3C mobileOk checker, but were not serious and did not affect the usability of the HTML5 Web application on mobile devices. The last update done to the W3C mobileOk checker was dated “2010-09-03” (W3C, 2010), which meant that some of the latest changes made to the draft in the last 8 months before the final call for changes, might not be implemented in the online checker. Therefore the validity of the test was considered as a third party verification of the HTML5 Web application.

**Summary**

The results that were achieved showed that the current state of HTML5, JavaScript and CSS3 on mobile devices was still not very stable between different devices. The limiting factors were not the actual standards though, but how and if they were implemented in the respective default browsers that came pre-installed on mobile devices. From the checklists that were compiled for each device that was tested, it was noticed the majority of the tests were passed and 12 out of 15, 13 out of 15 tests were successful for the Android HTC Desire smartphone and the Apple iPad respectively. But the tests that were not successful, even if in the minority group, were very important and critical for the Web application to be comparable with native applications, although some of the issues could be fixed. Therefore if the default or pre-installed Web browsers improved their standards compliance especially with regards to HTML5 and CSS3 when it came to mobile devices, there would be a higher percentage that the results would improve.
The implementations of HTML5 and the related standards lacked some critical usability features that were needed for a full-fledged mobile Web application to work on multiple mobile devices. It all depended on the browser the application was running in, because the correct implementation of Web standards as well as support for mobile application tags could not be controlled from within the application itself, and therefore the success of the application depended on third party software. This could in fact be a setback for those companies and enterprises looking to develop their next applications for mobile devices using the Web technologies, as they could think twice before choosing in favour of a mobile Web application or a native application. Of course one could have also used PhoneGap to wrap the Web application in order to produce a native application, which could then be re-distributed as a normal native application, with the possibility of re-distributing to multiple device platforms too.

During the research and analysis of possible ways to develop an HTML5 mobile Web application an interesting mix with ASP.Net MVC was found. This type of mashing of Web technologies with server-side coding, provided a complimentary scenario. Various Web applications could be created and would provide additional features that were provided only by using HTML5 standards and enhanced user-friendliness. HTML5 could also be combined with other server-side technologies such as PHP, JSP and ASP.Net which confirmed the wide compatibility, and usability of HTML5 as a Web technology in relation to other technologies, which showed the great potential of this technology.
CONCLUSION

During the analysis of the results, there was an indication that although HTML5 coupled with JavaScript and CSS3 had great future potential for multi-platform online Web applications, it did not have the same potential for offline Web applications. After the multi-platform mobile Web application was created with HTML5, JavaScript and CSS3 based on the model of an already existing Windows 7 only application, tests were carefully planned and the results were recorded for each test device available. The outcome of the results were examined and showed that the majority of the features listed in the HTML5 standard document did work on mobile devices, and provided an enhanced feel and greater user interactivity which benefitted the end-user. Though, the lack of refinement on some of the most crucial HTML5 features, incised on the success of online mobile Web applications, and especially for offline mobile Web applications. The features offered by the HTML5 standard along with other joint standards, gave way for developers to plan and develop a near native mobile application. This would reduce the costs and excessive waste of time to learn and develop two or more native mobile applications to target the desired devices. Unless the supporting technologies were implemented and supported as it was proposed in the standards documents correctly, the developers would be reluctant to switch from native mobile applications development in order to develop mobile Web applications.

After the results were recorded and analysed attentively, HTML5, JavaScript and CSS3 were examined to see if they were feasible and sufficient to develop a multi-platform mobile Web application, in contrast to develop multiple native versions of the same application. From the results obtained during the tests that were conducted, it was observed that not everything worked as expected on all the devices that the application was tested. In fact some issues were encountered on different devices albeit the issues vary from device to device.
device. When the application was tested on the Apple iPad, the offline mode of the application did not work as expected; likewise the results graph did not show the bar chart correctly. On the other hand, the HTC Android device had different issues, such as lack of a native application look and feel, deficiency of correct design resizing after the orientation was changed, and also other usability issues. These results show that a multi-platform mobile Web application could be feasible for small scale applications, but the issues mentioned above should be solved before the release of the application.

It was also examined if, to develop one multi-platform web application, it was less time consuming compared to creating multiple versions of the same application for various platforms. The outcome of the results pointed out that, in fact, it took less time to develop a multi-platform mobile Web application, due to the AppML framework that was used as the base structure of the Web application. The AppML framework was already tested on multiple devices, so the hard work normally incurred to design the application, and to test it on multiple devices was already done by the team who developed the mobile framework. When a mobile framework is used, less time is spent on the design of the application and more time is spent to plan the application and implement the functionality. This means that in the final release version of the application, there would be fewer bugs as more focus was concentrated on the functionality of the mobile Web application. On another note the fact that one application was developed rather than two separate applications, already halved the time taken to develop the application. The fact that the languages used to develop the application, were popular and known by the majority of developers, there was no need for any training. Apart from the popularity of the markup languages used, they were simple and produced a clean outcome which made the application relatively small and fast, which for a mobile device were vital requirements in order to have a responsive and functional mobile Web application.
There was still enough time left to implement the HTML5 standard along with the other standards proposed. Due to the latest proceedings after the distribution of the latest version of Internet Explorer 9 from Microsoft, it seemed that Google’s model of continuously updating the Chrome Web browser support for bugs and standards was gaining popularity amongst Web browser developers. In fact, both Mozilla which developed the Firefox browser, and Microsoft which developed the Internet Explorer browser, were starting to grasp this concept and model. If this trend of updating Web browsers frequently could get momentum, we could maybe see a wide implementation of the HTML5 specification by the end of the year on most desktop and mobile devices. In my opinion the only way to get developers and enterprises grasp over the concept of HTML5 Web applications, would be to mix both server-side Web applications developed using ASP.Net, JSP or PHP, in order to enhance the user interface, the usability of the applications developed and also enrich user interaction.

Ultimately HTML5 revolutionised the way we interacted with the internet through Web browsers, and the features found in most mobile devices such as Geo-location, the Accelerometer and other functionalities, provided new ways to develop websites and Web applications. The success of HTML5 mainly depended on the support for standards, and on the overall implementation throughout all the mobile and non-mobile devices which were able to connect to the internet and were capable to render the new HTML5 Web standard.
BIBLIOGRAPHY


W3C mobileOK Checker. (2010, September 3). Retrieved May 18, 2011, from W3C: http://validator.w3.org/mobile/


4 Syntax and basic data types. (2011, April 12). Retrieved May 14, 2011, from W3C: http://www.w3.org/TR/CSS21/syndata.html#vendor-keywords


http://www.jqtouch.com/


Kony Solutions. (2009). Buyer Beware: Not All Mobile Apps are Created Equal.


http://www.qualcomm.com/blog/2010/10/13/html5-back-future-fragmentation


Ranganathan, A. (2010, June 1). Beyond HTML5: Database APIs and the Road to IndexedDB. Retrieved March 18, 2011, from hacks.mozilla.org:

Multi-platform mobile application development using HTML5, JavaScript and CSS3.
Multi-platform mobile application development using HTML5, JavaScript and CSS3.


http://www.appml.org/


http://cubiq.org/iscroll

http://gs.statcounter.com/#mobile_vs_desktop-ww-monthly-200905-201104

http://dojotoolkit.org/features/mobile.php


Multi-platform mobile application development using HTML5, JavaScript and CSS3.  
Page 50
APPENDIX 1

Appendix 1 represents a set of photos of the Web application, which can be accessed at http://www.simonfacciol.info/html5, that were taken during the tests performed on the various mobile devices.
Multi-platform mobile application development using HTML5, JavaScript and CSS3.

FIGURE 6 - LANDSCAPE VIEW OF THE WEB APPLICATION ON THE IPAD

FIGURE 7 - THE ERROR SHOWN ON THE IPAD WHEN TRYING TO RUN THE APPLICATION WITHOUT AN INTERNET CONNECTION
Multi-platform mobile application development using HTML5, JavaScript and CSS3.