Mobile web apps as an alternative to native mobile apps

The future of mobile web apps on the competitive marketplace

Kim Hellbom

Degree Thesis
Film och TV / Online Media
2013

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Film och TV, Online Media
Identifikationsnummer:	4315
Författare:	Kim Hellbom
Arbetets namn:	Mobile web apps as an alternative to native mobile apps
Handledare (Arcada):	Owen Kelly
Uppdragsgivare:	

Mobila webbapplikationer har funnits på marknaden i flera år, men de har inte lyckats tävla med nativa applikationer i form av kundnåbarhet och användarupplevelse. Nativa mobilapplikationer har varit konsumenternas primära val, tack vare deras väl utvecklade distribueringssystem och för att människor helt enkelt inte har hört om mobila webapplikationer.

I mitt arbete kommer jag att undersöka ifall mobila webbapplikationer kommer att kunna tävla med nativa mobilapplikationer inom den förutsebara framtiden. Jag kommer att jämföra nativa, webb och hybrida mobilapplikationslösningar i form av kundnåbarhet, utvecklingskostnader, utvecklingshastighet och produktionsmiljöer.
En hybridlösning som heter PhoneGap möjliggör uppladdningen av en mobil webbapplikation till en digital distribueringskanal och denna lösning kommer att jämföras med webb och nativa utvecklingslösningar. Eftersom arbetet inte är en teoretisk studie, baserar sig studierna på erfarenhet jag fått från att utveckla en mobil webbapplikation med PhoneGap ramverket. För att få en bild över konsumenternas åsikt om mobila webbapplikationer, kommer en intervju att hållas i slutet av denna studie.

Resultat från denna studie visar att mobila webbapplikationer sannerligen har en möjlighet att tävla med nativa mobilapplikationer i när framtiden. Konsumenterna är beredda på att börja använda mobila webbapplikationer istället för nativa mobilapplikationer, så länge användarupplevelsen är lika bra. För mobila webbapplikationer att nå samma mängd konsumenter som nativa mobilapplikationer gör, kommer en digital distribueringskanal att vara oundviklig.

Slutligen kommer jag att sammanfatta resultaten och presentera min slutsats samt idéer för vidare undersökning inom området.

Nyckelord:	Mobile web applications, native apps, PhoneGap, BUILD
Sidantal:	50
Språk:	Engelska
Datum för godkännande:	

DEGREE THESIS	
Arcada	
Degree Programme:	Film and TV
Identification number:	4315
Author:	Kim Hellbom
Title:	Mobile web apps as an alternative to native mobile apps
Supervisor (Arcada):	Owen Kelly
Commissioned by:	

Mobile applications developed with web technologies have been around for some time, but they have not yet been able to compete with native applications in form of consumer reach and user experience. Native applications have been the number one choice for consumers, mainly because of their powerful distribution channels and also because very few people actually know what a mobile web app is.

In this thesis I will find out if mobile web apps will be able to compete with native applications in the foreseeable future. I will compare native, web and hybrid mobile application solutions in form of customer reach, development cost, development speed and production environments. A hybrid solution called PhoneGap that enables mobile web app developers to wrap their app in a native container and upload it to an app store will be compared to web and native solutions. This will not be a theoretical study. Research made for this thesis will be based on the experience I got while developing a mobile web app with the PhoneGap framework. To gain an understanding on what consumers think of mobile web applications, an interview will be conducted in the end of this thesis.

Results from this study show that mobile web applications will have a chance of competing with native mobile applications in the near future. Consumers are ready to start using web apps over native apps, as long as the user experience is good. A digital distribution platform is inevitable if mobile web applications are to have the same reach of customers as native applications.

Finally I will summarize the results and present my conclusion and ideas for further study on the subject.

Keywords:	Mobile web applications, native apps, PhoneGap, BUILD
Number of pages:	50
Language:	English
Date of acceptance:	

CONTENTS

Figures

Tables

1 INTRODUCTION

This chapter will introduce the main research topic and present essential background information on the central theme. The purpose, structure and method of the thesis will be defined and because of the wideness of the topic, a chapter on delimitation will be introduced. Finally, terminology used in the thesis will be presented.

1.1 Background

Smartphone usage vastly increased during the last couple of years in hand with mobile applications. According to Mary Meeker (2012) there are now over 1 billion global smartphone subscribers, a 42% growth from last year. Meeker points out that even though the tremendous ramp of the smartphones that has occurred, the smartphone user adoption rate is still far behind from mobile phones, that stand on a steady 5 billion global users.

Mobile Applications have taken a very central role in our daily life. Nowadays, they are the key to smart phone user experience. We use them to read news, listen to the radio, watch television and play games. The bigger the usage base grows, the more applications are developed. In February 2013, Apple had 800 000 applications in their App Store. It's an impressive number if compared to last year when they only had 475,000 applications available (148 Apps, 2013).

Mobile applications have largely been based on native development platforms such as C++, JAVA and Objective-C. When developing for a native platform, there are limitations that are important to take into account. These include for example: cross platform incompatibility, code management and the need of special expertise, just to name a few.

What if there would be a possibility to write an application that could run on all mobile devices, not being dependent on what operating system the device is using or what screen resolution it has. That is possible, thanks to HTML5, the catch-all term used to describe the latest web standards, HTML, CSS and JavaScript. Therefore, as long as the target device has a web browser, it will be able to run the application.

HTML5 has been around for quite a while now but it was not until 2009 when people really started talking about developing web apps using HTML5, thanks to Steve Jobs praise on how awesome web apps are (Killian Bell, 2011) and Google's successful implementation of HTML5-based apps (Alex Chitu, 2010).

There are several reasons why web apps have not yet boomed in the wide world of consumers. The lack of notifications, HTML5 does not provide an API for accessing the devices native functionality, such as the battery status or file storage and there is no central payment system for the platform. These functionalities are currently under development of the Device APIs working group, a group under the World Wide Web Consortium (W3C) international organization and might end up as a feature in our browsers in the near future (Frederick Hirsch, 2013). But before that day is upon us, we have to get creative. Luckily there is a hybrid solution, a framework called PhoneGap.
An API is a set of instructions that makes it possible for two pieces of software or web applications to communicate with each other.

1.2 Statement of the problem

Will web apps be able to compete with native apps in the marketplace in the foreseeable future?
The inescapable truth is that most people think of web apps as just another bookmark in the browser, and not as an experience. It is a trivialization that is occurring and it is not fair but that is just how peoples mind think of web apps these days (Matt Gemmel, 2011).

> "As mobile devices are taking the center of stage as computing devices, there will be less investment in improving Web technologies if it is not seen as relevant on mobile devices.(W3C, 2013)"

It is important that the web keeps evolving and expanding at the same phase as mobile operating systems and native development code bases. Native application distribution channels are controlled by single commercial entities, pushing the mobile experience on risk of censorship and monopoly. The web was made to be open, and it should stay that way. New APIs for accessing native functionalities are being developed, but the process is slow because of the multiple phases. Fingers are being pointed at W3C that administrate and make sure that HTML constantly gets updated with new APIs and other functionalities. The problem is not only the administrative organization but also the browser vendors.

They choose when and what HTML APIs they include in their web browser software. Some vendors are faster on deploying and some take more time. The evolution of HTML is also affected by the mindset of consumers and developers.

1.3 Purpose of the thesis

The purpose of this thesis is to find out if web apps will be able to compete with native applications in the marketplace in the foreseeable future. The debate between the power of web and native application development is fiercer than ever. While native apps provide a more fluent and delicate experience than web apps, the development process is longer and requires special expertise. And even then when the native application is at a stage of deployment, it can only be deployed on one operating system, for example iOS or Android. Web applications are developed using standard-based web technologies HTML, CSS and JavaScript. They only require a web browser to function, which means that every device on planet earth that has a web browser can run the application. For example smart fridges, smart televisions, smartphones and desktop computers. Developing web apps can save time and money and the app will be deployable on several operating systems.

Hypothesis 1: Web apps can be able to compete with native apps in the coming years if they are visually attracting.

Will the experience and customer reach be greater if the web application is wrapped in a native container and uploaded to an app store? Will consumers ever be convinced on the quality and the positive aspects of web applications? Can PhoneGap provide the same experience as native applications or will it only clash with the standards of the open web?

1.4 Structure of thesis and method

The thesis will start by presenting background information on the development of mobile web applications, and how developing web apps differ from native app development. It will then continue to examine why consumers have a different mindset on web apps than native apps. With this background information the aim is to justify that demand on mobile applications is overwhelming and mobile web apps need to find their place in between

native and hybrid applications. Further on the thesis will go into detail on how creating hybrid apps with PhoneGap differentiates in comparison to web apps, and how the two development solutions differ from each other.

To be able to answer the main research question, an inductive approach to collect qualitative empirical knowledge on how consumers view mobile web applications was used. The author of this thesis talked to over 60 people during the whole research process, asking what they know and what they think of mobile web applications. This helped portray a broad picture of consumers understanding on the subject. To support the empirical knowledge and consumers unawareness about mobile web apps, short interviews were conducted at the end of this study.
To gain a deeper understanding on the possibilities web applications and its core technology HTML5 offer and how it matches native applications, a hybrid mobile web application was developed as a part of this study. The development process and results is presented in detail and work as a valuable source of information for the final theoretical conclusion.

After analyzing and presenting data collected from articles and journals and the experience gained when developing a web application, data collected from the interviews will be presented, with the goal to analyze on how consumers value different aspects in mobile web apps compared to native applications. Finally, based on primary and secondary research, a theoretical conclusion will be presented. There will be a brief summary at the end of each chapter to recap the main points discussed.

1.5 Delimitation

While the thesis will address differences between natively developed mobile applications and web-based applications, it will not compare the two physical ways of programming the applications on a deeper level such as benchmarking and other relating techniques to prove the power of raw programming language. Neither will it explain the basics on how HTML, CSS or JavaScript work.

When comparing and talking about the different operating systems, the thesis will mainly be focusing on iOS, Android and Windows Phone.

This thesis will focus on web applications and their possible strengths in comparison to native based mobile applications and how consumer power will shape where web apps will be in the foreseeable future.
The functionalities and properties of the mobile web application developed as a part of this research will not be presented. This is due to it not being relevant to the thesis main research question and its goals.

1.6 Terminology

1.6.1 Abbreviations and acronyms

Android	Operating system made by Google
API	Application Programming Interface
App stores	Digital distribution platform for applications
Apps	Shorter and commonly used word for Applications
CSS	Cascading Style Sheet
Eclipse	IDE for Android Operating system
HTML	Hyper Text Markup Language
IDE	Integrated Development Environment
iOS	Operating system made by Apple
JavaScript	A used to make web pages interactive and dynamic
JIT	just-in-time compilation
jQuery Mobile	Web application user interface framework
OEM	Original equipment manufacturer
OS	Operating System
PhoneGap	An open source Framework
SDK	Development Kit
Sencha	Web application user interface framework
Skia	Open Source graphics library owned by Google

UI	User Interface
UX	User Experience
Web app	Web Application
Xcode	IDE for iOS Operating system

Table 1. Glossary

1.7 Summary

This chapter has introduced the main research topic, presented background facts on the central theme, the purpose of the thesis and delimitation of research data. The following chapter will present more in depth about the different technological development solutions currently on the market.

2 TECHNICAL SPECIFICATION

This chapter will present possible technical advantages and disadvantages between the different solutions that currently are, and will be competing with web apps on the market in the foreseeable future. Information about competing mobile application development solutions are crucial to take into account to fully understand the history and future developments within this field. The thesis will then more closely interpret the core programming languages used for developing web apps. After the interpretation of the programming languages and frameworks available, possible advantages in flexibility and speed between web and native technologies will be discussed.

2.1 Different solutions

Numerous aspects must be taken in to account when choosing the optimal solution for a mobile application. The optimal solution principally depends on what the main purpose of the application is, will it be able to run offline, who is the target audience, with what devices do people access current services and what is the budget for the project. Even after answering the questions above, the final decision may still widely vary from company to company. The final decision depends on the company's strategy such as making

beautiful experiences tailored as native applications exclusively for each and every operating system or concentrating on openness and delivering something for everybody.

Figure 1 shows how each solution has its pros and cons. The difference between a hybrid and a web solution is the cost, and between a web and a middleware solution it is agility.

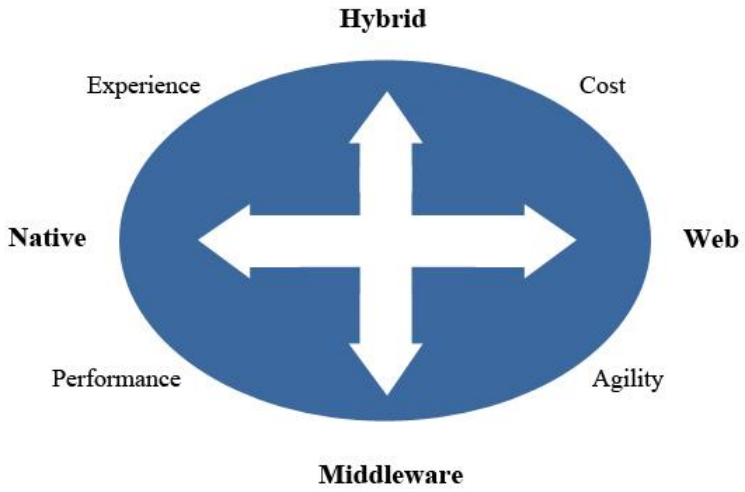

Figure 1. Differentiations between the Web, Native, Hybrid and Middleware development solutions

2.1.1 Native

Native applications are designed to run on the operating system level. This means that the application is developed using native programming languages, for example if developing for the Windows Phone platform, the language used is C#. Native applications are downloaded and installed on the device through an OEM app store like the Apple App Store or the Google Play Store. Native apps usually provide a more fluent and delicate experience than for example web apps, due to their tighter integration with the operating system. The disadvantage for developing native applications is their limited reach of customers. This is due to the fragmentation between the different operating systems code bases, for example a native app developed for iOS devices can only run on a device using iOS operating system. If the developer later decides to port over the app to Android OS, the app has to be rewritten in JAVA from scratch using the SDK provided by the OEM, in this case Google.

2.1.2 Web application

By choosing web as the platform for a mobile application, the mobile app will be fully based on web technologies such as HTML, CSS and JavaScript. Web apps run in the browser and are only accessible through a web address. It is not possible to install a web app physically on the phone, although it is possible to cache a web app for offline access, with for example the Application Cache interface (Mozilla Developers, 2013). Web apps can be bookmarked to the devices home screen, which will give them more of a native application feel.

When Apple launched the first iPhone in 2007, they did not provide a SDK for native development. Instead, they offered a Web SDK that would give developers the chance to make apps that work directly in the browser. At that time Steve Jobs praised web apps for their openness and easy development. This shows that already in 2007 web apps were considered as a possible substitute for native apps.

2.1.3 Hybrid

Hybrid applications are mobile web apps placed in a native container that enable them to access the devices native functionality. This is done with the framework called PhoneGap. Hybrid solutions are most suitable for projects with lower budgets but still require access to the devices native functions. Web applications using the PhoneGap framework can be deployed to several different operating systems with the same code base. Hybrid apps are faster to develop than native applications, they provide more functionality than a Web app and they can be uploaded to an OEM app store.

2.1.4 Middleware

Middleware mobile applications are apps used in the enterprise environment. They are built on a server that collects and organizes data from different sources in an enterprise back-end network. Data delivered through the server is then presented either as a web experience (web app) or a native application. This type of applications work well for larger enterprises or universities that have a complex back-end data network that needs to be able to communicate easily through a mobile optimized user interface. Thanks to their

integrated iOS and Android libraries, middleware apps are able to directly compile into native applications.
Kurogo is an open source mobile middleware application platform that collects raw data from different back-end data sources and delivers them through the requested UI provider, which can either be a native mobile application or a mobile optimized website. The platform has been highly popular within University's and Hospitals. Some web apps made with the Kurogo platform are the Universität St.Gallen: http://app.unisg.ch/home/ and St. Edwards University: http://m.stedwards.edu/.

2.2 HTML5

HTML5 is the fifth revision of the HTML markup language and also works as a term to describe the latest web technologies HTML, CSS3 and JavaScript. With these technologies, HTML5 can provide rich features and APIs for vibrant and desktop app like experience. Together, these web technologies give developers the possibility to build apps and websites with speed, functionality, performance and experience like desktop applications (HTML5Rocks, 2013).

The first draft of HTML5 was made public in 2008, but it was not until 2011 when HTML5 got released by the World Wide Web Consortium (W3C). At launch, support from different browser vendors for the new standard was still very poor, but it attracted a lot of attention and people were really excited about using it. Today all major browsers (Chrome, Firefox, Safari, Opera and IE) support HTML5, except older versions of IE (Christian Vasile, 2013).

When HTML5 was released in 2011, it was perceived as a dream come true for web developers. Rounded corners could now be done with CSS3, video content could easily be displayed on all devices with the new <video> element and updated forms gave the ability for validation straight in the HTML code.
HTML5 is still under development and the HTML Working Group is trying to get a recommendation status for the final HTML5 specification from W3C in 2014 Q4. The recommendation status means that it is a ready standard. The objective of W3C is not only ship the HTML5.0 standard, but also an HTML5.1 specification in 2016 (W3C, 2012).

The reason for splitting the 5.0 specification into two versions is clear. Due to the tight schedule of delivering the 5.0 specification in 2014, W3C wants to defer any new issues that are raised until HTML5.1 and concentrate only on issues that can be taken care of without major changes to the 5.0 specification.

2.2.1 Future of HTML5

Exciting APIs like getUserMedia and Web Intents will give web applications access to the target devices physical camera and the ability to pass rich data back and forth between web apps. These two examples are just a couple of the working drafts that are currently under the development by the W3C Device APIs group.

> "The mission of the Device APIs Working Group is to create client-side APIs that enable the development of Web Applications and Web Widgets that interact with devices services such as Calendar, Contacts, Camera, etc. (W3C)"

Below is a list of current working drafts active at the W3C Device APIs Working Group.

- Battery API
- HTML Media capture
- Network information API
- Proximity events
- Vibration API
- Web Intents
- Pick Media Intent
- Pick Contacts Intent

The development of numerous APIs show that HTML5 is taking huge steps and there is no sign of it slowing down. There is no exact date on when these APIs will be added to the HTML specification or to what version.

2.3 PhoneGap

PhoneGap is an open source framework that offers web developers the chance to develop native mobile apps with a set of standardized web APIs that gives control of the operating systems native functionality such as the camera and the accelerometer using JavaScript.

Apps are wrapped in a native container, so they can then be installed on a device as a natively developed application. The application UI is entirely rendered with HTML, CSS and JavaScript. Developers only need to write the application logic in JavaScript and the PhoneGap API takes care of the communication with the native operating system.
Most applications downloaded from app stores are natively developed, but the stores also contain a large number of web apps (Apple Corporation). This means that some applications have been developed using native code and a SDK provided by the OEM, and some have been developed with standard-based web technologies such as HTML, CSS and JavaScript. In September 2011, 4% of all apps in the Apple Apps store were built with PhoneGap (Atrice,2012). It is believed that this number has radically grown.

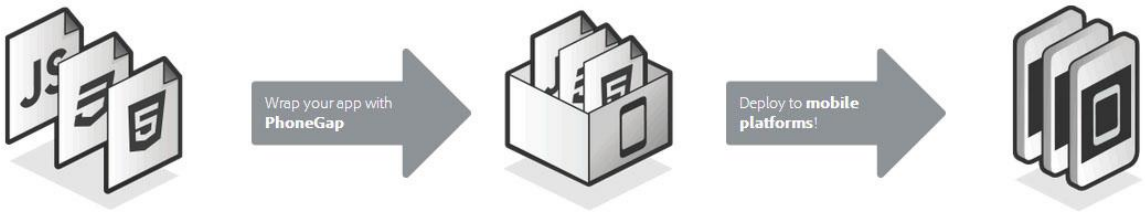

Figure 2. The cycle of the PhoneGap process. Image Phonegap.com

Web apps that are being deployed to an app store need to have a native container wrapped around, otherwise they will not be accepted. This layer is what PhoneGap offers. Even though after wrapping a native container around the web app, it is not guaranteed that app stores will accept applications made with PhoneGap. Apple particularly, keeps high standard applications in the App Store. According to Andrew Trice who is a technical evangelist at Adobe Systems, applications might get rejected from app stores due to the following reasons:

- "2.12: Apps that are not very useful, unique, are simply web sites bundled as Apps, or do not provide any lasting entertainment value may be rejected"
- "10.3: Apps that do not use system provided items, such as buttons and icons, correctly and as described in the Apple iOS Human Interface Guidelines may be rejected"

- "10.6: Apple and our customers place a high value on simple, refined, creative, well thought through interfaces. They take more work but are worth it. Apple sets a high bar. If your user interface is complex or less than very good, it may be rejected"
- "12.3: Apps that are simply web clippings, content aggregators, or a collection of links, may be rejected"

This signifies that the most common reasons for apps being rejected from the app stores are bugs, poor implementation of the UI and not following the human interface design principals provided by the OEM.

2.3.1 History

Development of PhoneGap first started at the iPhoneDevCamp in San Francisco 2008. One year later in 2009 PhoneGap gained attention from several medias and won the People's Choice Award at O'Reilly Media's 2009 Web 2.0 Conference. In 2011 Nitobi, the original software developer of PhoneGap was acquired by Adobe and the code got contributed to the Apache Software Foundation under a new project called Apache Cordova.

In 2012 PhoneGap announced BUILD, a new service that allows developers to upload and compile their web app code directly in the cloud. BUILD then generates packages of the application for all supported operating systems that can be directly downloaded and installed on the device. Currently PhoneGap is able to package applications to the following operating systems: iOS, Android, Windows Phone, BlackBerry, WebOS and Symbian.

2.3.2 Features

The main feature of PhoneGap is to link together native functionalities through its API with JavaScript calls. The process has three stages. Firstly, PhoneGap initiates a native WebView class (UIWebView on iOS devices). The WebView class is used to display web pages in native applications, (Apple Corporation, 2012). From the native code through the WebView class JavaScript is then called. When a connection has been established with the JavaScript in the web app and the WebView class, native functionalities

can then be executed, for example search for contacts or see what network the device is currently using.
The PhoneGap documentation provides a full list of native functionalities that can be used through the PhoneGap API:

- Accelerometer – Provides access to the devices motion sensor
- Camera – Capture photo's using the devices camera
- Capture – Provides access to the audio, image, and video capture capabilities of the device.
- Compass – Obtains the direction the device is pointing
- Connection – Gives access to the devices cellular and WiFi connection information
- Contacts – Provides access to the device contacts database
- Device information – Device specific hardware and software information
- Events – PhoneGap detects native events like "volumebuttondown" and "backbutton"
- File storage – Hook into the native file system
- Geolocation – Access to device location
- Globalization – The globalization object obtains information and performs operations specific to the user's locale and timezone.
- InAppBrowser – A web-browser shown inside the app
- Media – Record and play back audio files
- Notifications – Applications can push notifications to the device
- Splashscreen – Loading screen for the application
- Storage – Create and maintain a local database

2.4 Flexibility

The catch with web apps using the PhoneGap framework is, that even though they are installed as native apps, they are still just web pages. This means that the same techniques used to optimize and speed up websites, will also work on web apps installed on mobile devices. Caching, image optimization, CSS and JavaScript minification scripts all make

an impact on the performance of the web app. These are common ways of optimizing the performance of a website. Optimization techniques were only briefly mentioned to show the different possibilities developers have.

Native applications can also be optimized in many different ways, however require significantly more knowledge from the developer in that specific area to accomplish the same results.

The main advantage web apps have over native apps is their cross-platform availability. But at the same time, it is also their weakness. Every smartphone has a web browser, meaning that every smartphone regardless of what operating system they are running on, can access the same web application. However, different operating systems have different default web browsers and they all render webpages and web apps a bit differently.

Web browser	**Rendering Engine**
Safari	WebKit
Google Android	WebKit
Firefox Browser	Gecko
IE Mobile	Trident
Chrome	Blink
Opera Mini	Presto (moving to Blink)

Table 2. The most used Mobile web browsers and their rendering engine (Netmarketshare, 2013)

The way in which web browsers render webpages differ amongst vendors. The differentiating factor is the rendering engine and how they are used by each vendor. Differentiations can be noticed in how text is being rendered and placed and the support of CSS properties may also vary. As it is visible from the table above, most browsers have implemented the WebKit rendering engine. The first thought would be "then they all probably render the web pages the same way". Unfortunately that is incorrect. Even though the browsers use the same rendering engine, they still have different APIs implemented and differentiating ways of rendering text and graphics.

Paul Irish who works with the Google Chrome development relations has listed up some of the components that modern web browsers use to function:

- Parsing (HTML, XML, CSS and JavaScript)
- Layout
- Text and graphics rendering
- Image decoding
- GPU interaction
- Network access
- Hardware acceleration

Of these components, only the top two are shared between WebKit-based browsers. The other components are handled differently on each WebKit port. A WebKit port is a different software that has been branched from WebKit and been modified. Most known WebKit ports are: Chrome (Chromium), Safari and Android Browser.

> "For example, if you specify a flat colored button with specific border radius, well WebKit knows where and how to draw that button. However, the final actual responsibility of drawing the button (as pixels on the user's monitor) falls into CoreGraphics. As mentioned above, using CG is unique to the Mac port. Chrome on Mac uses Skia." (Paul Irish, 2013)

CoreGraphis is a C-based framework used to display lightweight 2D rendering with unmatched output fidelity (Apple Corporation). The reason for going in to such depths on rendering engines and the WebKit engine is that web developers have to be aware of the issues and be prepared to perform detailed testing in a wide range of browsers. Preferably also the same browsers but with different WebKit versions.

2.5 Power

On iOS devices, HTML5 and JavaScript performs faster in a mobile web browser than inside a natively wrapped web application. This is because the JavaScript engine is faster in a mobile web browser than in the WebView (UIWebView for iOS) class that native applications use to display websites (Ali Saffari, 2012). Tests made by Ali Saffari show that the difference is minimal, but other research results disagree.

As mentioned earlier, when developing web apps with PhoneGap for iOS devices, the HTML and JavaScript is viewed through the UIWebView class. The UIWebView class has the same Safari mobile rendering engine but it does not have a JIT enabled JavaScript engine that the Safari mobile web browser has. JIT stands for just-in-time compilation

and it makes the running of JavaScript based applications much faster. As the name suggests, it compiles the code "on the go" while the application is running. This makes the application quicker and more optimized (Oracle Corporation, 2008). Apple does not allow third-party developers access to the JIT enabled compiler and the Nitro JavaScript engine, according to Apple this is due to security issues. It is reported that the UIWebView class can be up to two times slower than the rendering engine used in the Safari mobile web browser (Ariya Hidayat, 2012).

What is interesting here is that the Google made Chrome browser for iOS devices was released last spring and Google said during the announcement that the application is running through the UIWebView class, which should make it slower than the Safari browser. And it surely is (Frederic Lardinois, 2012). Tests made by Michael Simon shows that the Google Chrome browser for iOS is nearly 50% slower than Safari on all JavaScript tests. It fails to outrun the native Safari browser on all fronts (Michael Simon, 2012).
These tests do not show the overall performance of a web app using the PhoneGap framework. They portray the raw performance of the JavaScript engine, for example how fast the engine can render a 3D canvas. Once a webpage has been loaded and cached, the speed between the UIWebView and a native web browser is nearly identical.
Most PhoneGap apps are quite simple and light and do not include tasks that involve heavy computation. The matter of JavaScript engine speed is something that should not be taking too seriously. Even though web technology will never run as fast as compiled native code, it still provides a decent performance.

2.6 Summary

This chapter has introduced the different technical solutions that are available when developing mobile apps, what HTML is and where it is going, the opportunities PhoneGap offers, compatibility issues in web apps and how JavaScript rendering engines can compete against natively compiled applications.

3 PRODUCTION

This chapter will introduce the different phases during web app development using the PhoneGap framework. Cost efficiency and speed of development are important factors to take into account when developing mobile applications. Possible advantages and disadvantages for web apps in that area will be discussed. As a part of this research a web application with the PhoneGap framework was developed. The observations from the development process are intended to complement the research and analysis in this chapter. This chapter will also introduce historic examples of cross platform mobile app development.
A short part of the sub chapter Coding will require some basic understanding of HTML and JavaScript.

3.1 Historic examples

It has never been easy to port a native application to several different operating systems. Native apps are built using the preferred technology for one specific device and OS, so if one choses to develop a Windows Phone application, that application will only be able to run on a Windows Phone device. This means that iOS, Android and Windows Phone all have own preferred technologies to work with.
Sun Microsystems once created the slogan “Write once, run anywhere” (WORA) for the programming language JAVA to demonstrate the cross-platform benefits from the programming language. In an ideal world, it would mean that a program is written once, after which it is then able to run on all different devices with a JAVA Virtual Machine, the runtime environment that interprets compiled code and performs the requested actions. The purpose is to save on development costs and make it easier for developers to reach out to a larger volume of devices with the same code. Much of the same goals that HTML5 is aiming for.

The mindset of WORA is growing and it is good to see that web apps have this area already covered. In an interview made by Goldman Sachs, Microsoft reveled that they are pushing their operating system forward to make development of apps for all Windows devices the same. Developers will be able to release the same application for PCs, tablets

and smartphones. Microsoft is also supporting the creation of PhoneGap apps for the Windows 8 operating system thanks to the Windows Library for JavaScript.

This is important because it indicates that what HTML5 is now trying to accomplish is what JAVA and many other always have had as their goal.

3.2 Development speed

In the middle of 2012 Facebook announced a native mobile application for iOS devices. The native application was going to be "blazing fast" and it would also be replacing their current web-based application. Mark Zuckerberg reviled in an interview that the biggest mistake Facebook did was betting too much on HTML5. He explains that HTML5 "just wasn't there". After Zuckerberg made this announcement, rumors started circling on the internet that the problem was not actually HTML5, but the lack of expertise inside the Facebook development team. (Matt Asay, 2012)
Developers of the Sencha mobile web app framework made a copy of Facebook's web app called "Fastbook", to prove that the problem was the Facebook developers, not HTML5. The application made by the Sencha team was significantly faster and more efficient than the one Facebook made.
While the process of developing web apps can be relatively faster than native app development, if not done correctly by following documentation, the results can be underperforming and sluggish.

One of the most important factors in a mobile application is speed. Research has shown that consumers expect their application to launch within three seconds or less (Compuware). Speed is something that web apps have always been good at, at least when the web app stays fairly simple. Accounting software startup Xero announced 18.3.2013 that they will be moving away from their web-based application to a native version because of resource intensive maintenance work and difficulties in further development of their web app. They continue on explaining that developing a complicated mobile application in HTML5 is hard and that it takes a lot of time and testing to bring the HTML5 app to the native level of performance.

If the goal with the web app is to give the exact same feel and performance as a native application, it is wiser to choose a native development technology right from the beginning, as Matt Vickers explains.

> "Our view is that HTML5 technologies can deliver as-good-as-native experiences…but the lesson from Fastbook is that it's hard work – you don't get those experiences out-of-the-box. And the lesson we've learnt over the last 12 months has been that the cost in time, effort and testing to bring an HTML5 application to a native level of performance seems to be far greater than if the application was built with native technologies from the get-go. (Matt Vickers, Xero 2013)"

A very important part with mobile app development is to do proper research before starting development. The target audience has to be identified, a market reach plan and a strategy for the coming five to ten years has to be made. This research will help to portray who the target audience is and which technological solution might be the best fit.
At the moment mobile web applications are ideal solutions for simple and lighter projects, even though PhoneGap offers endless opportunities for the creation of high demanding applications. To develop a complex web app that will perform and look as a natively developed, it will take much more time and resources to produce it then choosing native technologies directly from the start. That is at least to which conclusion the startup firm Xero came to. Research made for this thesis shows that it mostly depends on the developers set of skills.

As explained in the chapter about Power, the most substantial difference in speed between web and native applications on iOS devices is the reduced accessibility for third party applications to use the JIT enabled JavaScript rendering engine. Web apps using the PhoneGap framework are generally slower than natively developed applications. This is mostly due to the amount of feature-heavy transitions, effects and user interface frameworks that are being implemented in web apps. For example when a button is pressed with the action to show an overlay dialog, the DOM needs to alter itself in order to show that overlay. These changes use the browsers resources and JavaScript engine, which on mobile devices are less powerful than on desktop browsers. The more complex HTML and CSS3 markup that is being added to the web app, the more it will drain on the performance. By following the tips listed below, the performance of the web application can be increased.

3.2.1 Add hardware acceleration

By adding the CSS translate3d property to a HTML class, the scrolling and transitions will be smoother on iOS devices. The code below enables hardware acceleration on devices using a WebKit powered browser. Translate3d will normally transform elements on the page but if the parameters are left empty, it will only enable hardware acceleration. On Android devices this code can cause some unpredictable behavior. It is not suggested to be used on that OS. Transform3d is supported on WebKit Browsers and Firefox.

```
div.yourClass {
       -webkit-transform: translate3d(0,0,0);
}
```

3.2.2 Pre-compiled templates

The use of logic-less templates like Handlebars.js and Dust.js can make significant boost in the web applications performance. Greg Avola explains how they managed to make their PhoneGap app "Untapped" two times faster thanks to the Handlebars.js semantic templates (Greg Avola, 2013).

> "We settled on using HandlebarsJS because it offered us the ability to add logic into our templates and the ability to pre-compile the HTML templates into JavaScript-optimized versions for better performance. Through compiling, we were able to increase page rendering by two times compared to the time taken to page render pages that were not pre-compiling."

Templating will require more advanced JavaScript skills. The web application developed as a part of this research, did not use templating due to limited resources.

3.2.3 Keeping the markup minimal

When developers write code in a hurry, unnecessary classes and tags can surface and make the code more complex than it really needs to be. Always remember the WORA term. For example when writing a simple list populated by hyperlinks instead of looking like this:

```
<ul>
        <li>
          <a onclick="doSomething();">Stuff</a>
        </li>
</ul>
```

It could look like this:

```
<ul>
        <li onclick="doSomething();">
           Stuff
        </li>
</ul>
```

By leaving out the hyperlink anchor tag, the markup is simplified. Here it might not look like it would make a greater impact, but when the app in question has a lot of data and many pages, everything can make a difference.

3.2.4 The correct way of using user interface frameworks

JQuery mobile and Sencha are powerful user interface frameworks to use with web apps. They provide ready to use components and flexible theming systems. When these frameworks are used together with PhoneGap, they can cause performance issues if not implemented properly. JQM has listed up some tips and recommendations in their documentation that can help when developing web apps with PhoneGap and JQM (jQuery Mobile, 2013).

3.2.5 Summary

By following the tips mentioned, the performance of the web application can be improved.
Currently the issues for mobile web apps in regards of speed is the lack of resources in web browsers and device hardware. Resources meaning cache size limits and slower rendering engines, compared to desktop browsers. With the rise of quad core processor powered mobile phone devices, the problem with limited resourced device hardware can hopefully soon be tackled.

3.3 Developing with PhoneGap

Getting started with the process of developing web apps using the PhoneGap framework requires the developer to firstly set up a proper development environment. There are different ways this can be done, mostly depending on what operating system the application is targeted for and the developers background as a designer or developer. As Andrew

Trice, technical evangelist at Adobe Systems points out “Since PhoneGap applications are really just editing HTML, CSS & JavaScript, you can use whatever editor you want” (Andrew Trice, 2013). People who come from a development background might prefer larger-scale IDEs, because they tend to offer richer languages/features. But any light-weight HTML editors works as well. An IDE is a programming environment that usually consist of a code editor, debugger, compiler and a GUI for building graphical interfaces, like for example a mobile app. These environments can be very powerful thanks to their versatility and broad library of extensions and plugins. IDEs can be very expensive and that is why they are more used within companies rather than private persons. There are a few catches for using IDEs. For example when the web app reaches the stage of deployment (not using the cloud service BUILD), it has to be deployed using the IDE for a particular platform that is being targeted. So for example if the app is developed for iOS devices, the Xcode IDE has to be used and if the app is developed for Android devices, then the IDE is Eclipse.

Dreamweaver is a preferable choice for developers who do not want to be dependent on a specific operating system. Dreamweaver offers a rich WYSIWYG editor and an integrated BUILD plugin from where the app can be directly uploaded to the cloud where it then gets compiled to all different operating systems. Developers already familiar with Dreamweaver should notice how nice and fast it is to use with PhoneGap. There is also a jQuery Mobile plugin for Dreamweaver, which makes developing web apps extra neat if the jQuery mobile framework is preferred.

3.3.1 Set up

In this subchapter I will switch to the first person to report on my personal experience in different ways of setting up a development environment. These personal observations are intended to complement the research and analysis in the rest of the thesis.

When I started developing apps with PhoneGap, instead of using the BUILD cloud compiling service, I decided to try installing respective OS IDEs with SDKs. SDKs are packages that contain the necessary building blocks for developing the application such as headers, libraries and frameworks. SDKs are used within IDEs to make the use of SDKs more user friendly.

So when I started developing a Windows Phone app, I had to install Visual Studio 2012, which was a pretty easy thing to do since the installation of the SDK was very straight forward. After successfully making a Windows Phone app, I then moved on to building an app for the Android OS. Setting up the development environment was much trickier. To develop Android apps Google uses a software called Eclipse, an open development platform developed and maintained by the Eclipse Foundation. When installation of Eclipse was done, the next step was to set up the correct repositories from where you can download the Android Phone SDK. Eclipse is a software that supports development on a vast range of platforms. That is why setting up the correct environment can take a lot of time and effort. To develop HTML and JavaScript files in Eclipse, a specific plugin for Web development had to be installed, which I was not really fond of.

The attraction to develop hybrid apps for Android devices is that the operating system is open source. When your application is at a stage that you need to test and debug it on a real device, Apple and Windows Phone requires you to register the phone you want to debug on, that means you have to pay the annual developer fee of 99$ (for Android it is 25$). If you are just interested in starting to develop apps for mobile devices and want to give the platforms a go without paying the subscription fee, Android is the right choice. If you later on decide that you want to submit the app to the Google Play store, then you can pay the registration fee.

All these steps can be skipped if you choose to use BUILD, a cloud service made by Adobe and PhoneGap. BUILD lets you upload your HTML, CSS and JavaScript files to the cloud service where they are then compiled and packaged into app-store ready applications.

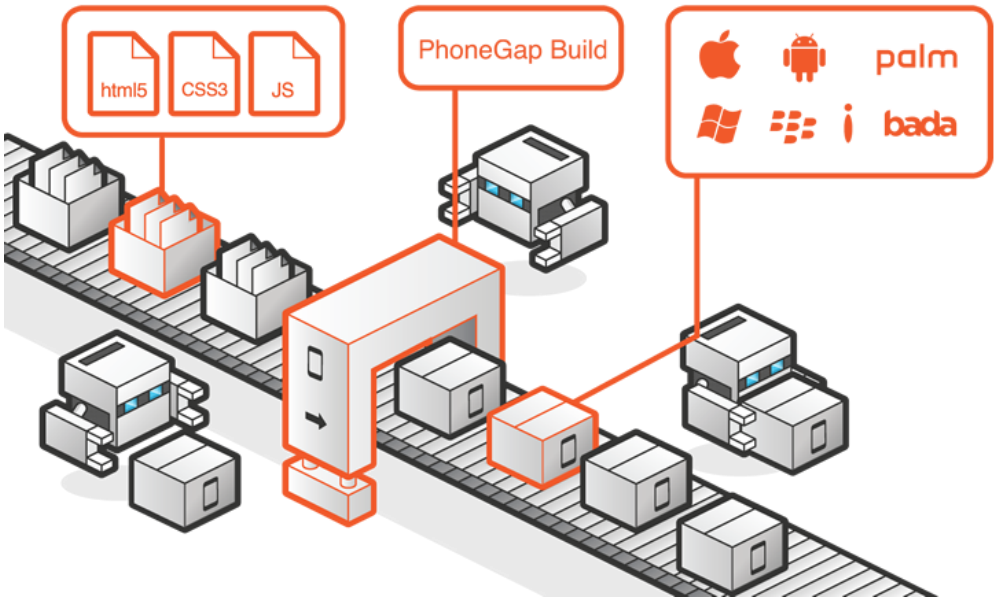

Figure 3. How PhoneGap Build works. Image from build.phonegap.com

This is a very powerful service. It lets developers concentrate entirely on the code itself, not spending ridiculous amount of time trying to set up the development environment and then keeping the SDKs up to date. BUILD is free to use with up to one private application. Private application means that nobody else can see the application you are developing. If the app is open source and hosted at a GitHub repo, there is no limit on the application amount.

As mentioned earlier, Adobe made a powerful plugin for Dreamweaver that lets you directly upload your HTML, CSS and JavaScript files to the BUILD cloud service. As with all development, debugging is an important part of the process and PhoneGap has not forgot about it. They have included a web-based remote debug mode in to the BUILD cloud service. The debugger detects when a device is running the application and can then interactively modify the application. The service is similar to the Chrome developer tools and the Firefox Firebug.

Getting started developing PhoneGap powered web apps with the BUILD service was significantly faster and easier than developing using a physical SDK and a native development environment. The native development environment does have a superior advantage with the integrated Debug emulators. Developers are able to test their web app in

all different emulators, ranging from low resolution phones with an old version of Android to high resolution tablets with the newest version of the OS. So it highly depends on the purpose of the application.

PhoneGap offers a prototype Emulator that is powered by Ripple, a web-based emulator that can be installed as a Chrome extension. The Ripple emulator enables developers to quickly test how their application works and feels on different screen sizes. The emulator can be used for JavaScript debugging, HTML DOM inspection, automated testing and it also supports a wide range of PhoneGap APIs.

When the development environment is all set up and the HTML file is communicating with the PhoneGap API, the next steps are to start developing the app.

3.3.2 Coding

Once the development environment has been set up and a new project has been created, the developer will be able to focus their attention on HTML, CSS and JavaScript. This thesis will not explain how to create a new project since it varies a lot from platform to platform. PhoneGap offers detailed documentation on how to get a project up and running (PhoneGap, 2013). The first thing PhoneGap will do when the app is launched is listen for the event "deviceready". When this event is called, that means that the application has been fully loaded.

> "PhoneGap consists of two code bases: native and JavaScript. While the native code is loading, a custom loading image is displayed. However, JavaScript is only loaded once the DOM loads. This means your web application could, potentially, call a PhoneGap JavaScript function before it is loaded. " (Phonegap)

When this event fires, it means that JavaScript has established a connection with the native UIWebView class and that it is now safe to call PhoneGap JavaScript functions.
A normal setup for an HTML PhoneGap application will look like this:

```
<!DOCTYPE html>
<html>
  <head>
    <title>PhoneGap Device Ready Example</title>

    <script type="text/javascript" charset="utf-8" src="phonegap-1.0.0.js"></script>
    <script type="text/javascript" charset="utf-8">

    // Call onDeviceReady when PhoneGap is loaded.
    //
    // At this point, the document has loaded but phonegap-1.0.0.js has not.
```

```
    // When PhoneGap is loaded and talking with the native device,
    // it will call the event `deviceready`.
    //
    document.addEventListener("deviceready", onDeviceReady, false);

    // PhoneGap is loaded and it is now safe to make calls PhoneGap methods
    //
    function onDeviceReady() {
        // Now safe to use the PhoneGap API
    }

    </script>
  </head>
  <body>
  </body>
</html>
```

The JavaScript found below is a very simple and basic way of getting the device geographical location and then presenting the values through an alert. As you can see in the "onSuccess" function:

```
<script type="text/javascript" charset="utf-8">

    // Wait for Cordova to load
    //
    document.addEventListener("deviceready", onDeviceReady, false);

    // Cordova is ready
    //
    function onDeviceReady() {
        navigator.geolocation.getCurrentPosition(onSuccess, onError);
    }

    // onSuccess Geolocation
    //
    function onSuccess(position) {
        var element = document.getElementById('geolocation');
        element.innerHTML = 'Latitude: '+ position.coords.latitude + '<br />' +
                            'Longitude: '+ position.coords.longitude + '<br />' +
                            'Altitude: '+ position.coords.altitude + '<br />' +
                            'Accuracy: '+ position.coords.accuracy + '<br />' +
                            'Altitude Accuracy: '+ position.coords.altitudeAccuracy +
'<br />' +                  'Heading: '+ position.coords.heading + '<br />' +
                            'Speed: ' + position.coords.speed + '<br />' +
                            'Timestamp; + position.timestamp + '<br />';
    }
    // onError Callback receives a PositionError object
    //
    function onError(error) {
        alert('code: '    + error.code    + '\n' +
              'message: ' + error.message + '\n');
    }
    </script>
```

When the code is at a stage that it needs be tested on a real device, there is a couple of different ways to do it as mentioned earlier in this chapter. The easiest and fastest way is to upload the files to the BUILD cloud service where they will be compiled and packaged ready for download and installation on the device.

3.4 Cost efficiency

Due to the proliferation of different browser versions especially on Android, the WORA term for web apps has changed its real meaning to WOOE, Write Once Optimize Everywhere. The continuously evolving web market introduces new browser vendors and devices with different screen sizes at a tremendous pace, which requires developers to test and optimize their web application for a wide range of different devices. When the target audience is getting more dispersed out with different mobile phone operating systems and browsers, more testing and optimization is required which results in more expenses. When developing native applications for the iOS platform, all target users have the same operating system. With less target devices and more controlled versions of the operating system, there is a possibility that natively developed applications for iOS devices require less optimization than web apps developed for their fragmented market. Android and Windows Phone devices share the same dilemma as web apps. They have a lot more different sized devices with different technical specifications. Windows Phone is a bit more controlled with their OS versions 7.5 and 8 but Android has over six active versions of their OS running on peoples mobile phones around the world.

3.4.1 Time and money

Let us say that company X has decided that they want to provide a sleek and powerful mobile UX for their company's website. Before they can even start developing any kind of application, they have to do proper research on who their target audience is. They also have to know what the consumer does on their website and with what devices they are currently accessing it. When they have the answers to the questions above they can start thinking on what kind of solution would works best for them. Native, web or hybrid?
Will the application be made for a shorter marketing campaign that only last a couple of months with a goal of reaching all the people around the world or will it be made as the main way for consumers to access the company's services with a superior UX?

If for example a bank is planning on developing a mobile application from which their customers can access their personal internet banking services through, the obvious choice for the bank would be to make a native app. Banks like the Bank of America and Barclays

Bank do offer lighter web-based versions of their online banking services. But for the full-fledged service, they provide native applications. W3C is improving the security in HTML5 in upcoming releases, but for services like banking, extra layers of encryption and firewalls is still needed to securely protect the data.

Web apps are more efficient for certain tasks but if the main arguments for the mobile app is to be extra secure, have a proper payment system and provide a delicate experience, then the choice would be native.

3.5 Summary

As HTML5 continues to evolve and introduce new and powerful APIs, so will also the processes of developing web apps. Getting started using PhoneGap with web apps can require a bit of patience, especially if choosing an IDE as development environment. Luckily that is something that is going to get easier and easier, as we have seen with the support Adobe is showing for PhoneGap with their BUILD service. It is important to remember that web apps should look like and act like web apps. If the world is to be persuaded that web apps are here to stay, then developers have to start thinking like that as well.

4 COSUMPTION

In order to get consumers to use more web apps instead of native apps, developers have to start creating web apps with beautiful UI and a good UX. Research and interviews made for this thesis show that most consumers don’t have a clue on what a web application is. People simply refer all kind of activity done in the mobile web browser to as surfing on a mobile optimized website, which web apps in some consent are, but add some PhoneGap functionality and make it downloadable from an app store and nobody will know it is an application based on web technologies. According to the results from the interview in Chapter 5, the consumer doesn’t care and doesn’t need to know that they are using an app based on web technologies, as long as it is fast and they enjoy the experience.

4.1 Acceptance and market reach

Web apps are based on the same web technologies as any other website which makes them accessible through any web browser. A big substantial difference between native and web apps in form of market reach, is that web apps do not have an ecosystem as native apps do. Native applications are uploaded to a digital distribution platform often referred to as an App store, from where consumers can then download the applications. The name App Store is an official trademark registered for the Apple Corporation. OEM's have branded their app stores under different names. Apple's is the App Store, Google's is Google Play and the Windows Phone app store is called the Marketplace.
The question is, how will web apps be able to have the same global reach as native apps?

In the beginning of the year 2013, Apple had around 780 000 applications in their app store, Google's Play store had around the same amount and the Windows Phone Marketplace had 130 000 applications (148 Apps, 2013). That is an impressive number if it is compared to the year 2012 when Apple only had 475,000 applications in their app store. The app stores are there to make it easier for consumers to find new apps, read reviews and also offer developers an easier way of monetizing their apps.

In 2010 Google launched the Chrome Web Store, an attempt to make web apps more accessible and easier to find. The Chrome Web Store is a web app store from where people can download web apps that run in the browser. Currently all web apps found in the Chrome store are for desktop computer web browsers only. Google has not given any indication on if it will start populating the web app store with mobile web apps as well (Erik Kay, 2010). Apple and Firefox are following Google's step with the launch of their own web app stores. Although big corporate giants like the ones recently named, show great support for web apps, the selection of mobile web apps in these stores is still sparse.

Mobile web app stores are a must if web apps are to reach the handsets of consumers. People will have to be able to view, search, review and of course download the apps. With a digital distribution platform, the global reach of consumers would be much greater.

To help consumers find mobile web apps on the Windows Phone OS, a company called Sydience developed the app WebApps. The app has a collection of web apps designed for mobile phones. Users can browse, review and pin apps to the start screen, much of the same functionalities that are already found in OEM app stores. The biggest difference between WebApps and the actual app store is that applications can't be installed on the device, since they are web apps that are running in the web browser. WebApps have taken care of this problem intuitively by letting users keep their own collection of their favorite web apps inside the app. This gives the feeling that the web apps are actual native applications. (Sydience, 2013).

The only feature this app is missing is an integrated payment system. Nevertheless, this is the right direction for HTML5 and this kind of solutions will definitely help attract more people to using web apps.

4.2 Perceived professionalism

BlackBerry (formerly known as RIM) has lost more than 30 percent of its market share to competitors iOS and Android in the United States. Their market share is expected to be around 1.8% in the United States, 30 percent lower than it was three years ago (Forbes, 2013). The biggest problem Blackberry faces today is how they can attract new people to jump over to their ecosystem. One key factor that affects the consumers choice of mobile ecosystem is the amount of apps the platform offers. BlackBerry has tried to tackle the matter with offering at launch of their new BlackBerry 10 operating system 70 000 applications, from which 1000 will be premium apps from top developers. That number seems quit big for a totally new operating system, if it is compared for example with the Windows Phone ecosystem that just managed to cross the 130 000 application mark in their Marketplace. It certainly makes people think "wow they have really put a lot of effort to please us, the consumers". But the thing is, that that number is not totally correct. Every fifth application found in the BlackBerry app store is a ported Android application. BlackBerry 10 has the ability to run Android applications as they were BlackBerry ones. The BlackBerry 10 operating system uses an emulator to simulate an Android OS from which they can then run Android applications. People who have had the chance to test the Black-

Berry 10 operating system and its ported Android applications have reported that the experience offered through this Emulator is not as fluent as with native BlackBerry applications.
Matt burns writes how BlackBerry might have caused a momentum around their OS with this tactic but for the platform to really succeed, proper native and web-based applications are needed for the OS (Matt Burns, 2013).

BlackBerry could have tried to attract web developers to port over their PhoneGap powered web apps instead of Android applications that would have a bad performance. How would the reaction have been if one fifth of the applications would have been a PhoneGap powered web app? It could have been a good opportunity for BlackBerry to show their support for the open web and web apps in general. They chose not to do that because they wanted to port well known Android apps that already had large user bases like for example Skype.

4.3 Summary

This chapter explained how a digital distribution platform would help consumers explore a broader collection of web apps and how BlackBerry chose to port over Android apps to their new operating system.

5 INTERVIEW

In addition to the web application that was developed as a part of this research, qualitative research was obtained through consumer interviews. This method helped portray a picture view on how consumers view and react to web apps and a possible mobile web app store.

5.1 Method

Four semi-structured face-to-face interviews were conducted with people aged between 22 and 26, all of which were active smartphone users. The goal of conducting interviews was to find out if consumers know about web apps and how they value different aspects in web apps compared to native app.

The interviews were semi-structured so that the interviewees could easily elaborate and share their own opinions and knowledge on the matter to gain deep qualitative insight. Questions chosen for the interviews were inspired by a study made by Compuware in 2012 and the findings in chapter three. The chosen questions were constructed to encourage the interviewees to express their own personal experience. This would provide more in depth knowledge from each individual respondent. Themes for the questions included the web app UI, UX in web apps compared to native apps and what value a digital distribution platform brings. In the beginning of each interview, a short introduction about what a web app and native app is was made.

The interviews were not intended to produce a statistically valid quantitative survey. The goal was to gather consumer opinions on web apps and their usage of web apps. The questions can be found in table 3.

An iPod Touch was used as a demo device. The Financial Times web app was bookmarked to the home screen and the native Guardian app was pre-installed on the device. The interviewees were asked to play around with the two applications.

After the interviewees had explored and used the mobile applications for five minutes, they were asked questions listed in table 3. The interviews were recorded with the respondent's acknowledgement and allowance.

5.2 The Apps

The Financial Times and the Guardian were chosen as demo applications because of their opposite strategic paths. In 2011 Financial Times decided to drop their native mobile application in favor of a web-based. One of the reasons for them moving away from a native application to a web-based was Apple's newly announced 30% cut of in in-app subscriptions. This would mean that every time a new person signs up as a Financial Times subscriber, Apple would take 30% of the payment. According to Financial Times, this was not the only reason for switching platforms. A web-based application would also allow them to maintain a direct relationship with the consumer and to be able to quickly scale across different devices and platforms. (Mobithinking, 2012).

Thanks to their new HTML5 mobile web app, smartphone users have increased by over 52%, while tablet users are up 70% (Lauren Indvik, 2013).
The Guardian has chosen a different route. They have developed a native version of their app for every major mobile operating system. A native version of the Guardian is found on iOS, Android, Windows Phone, Kindle and BlackBerry. For none smartphone users they offer a mobile optimized website.

It is noticeable that the Guardian has invested in a more native user experience. The Guardian app lets users customize the front page with specific news categories, choose what content is cached for offline reading, the possibility to mark articles as favorites and set up goal alerts for football live coverage. From the features listed above, offline reading is the only feature that is also available on the web-based Financial Times app. However, as the Financial Times is a web app, it offers cross-platform access and the possibility to deploy app updates seamlessly on-the-go. Technically all the features available on the Guardian app could also be implemented in the web-based Financial Times, except for the Goal alert which uses notifications that are not yet available with web technologies.

After analyzing these two applications, as a designer and developer, The Guardian is a more stable and polished application that offers a more delicate experience with more functionality.
During testing, the Financial Times app did occasionally crash and scrolling through articles did sometimes cause flickering on the screen. As a web developer, these are commonly known issues with web technologies, but not for a consumer who will think of them as bugs. One functionality that made FT more interesting over the Guardian was the possibility to scroll through news categories by swiping to the left or right on the screen (see figure 4).

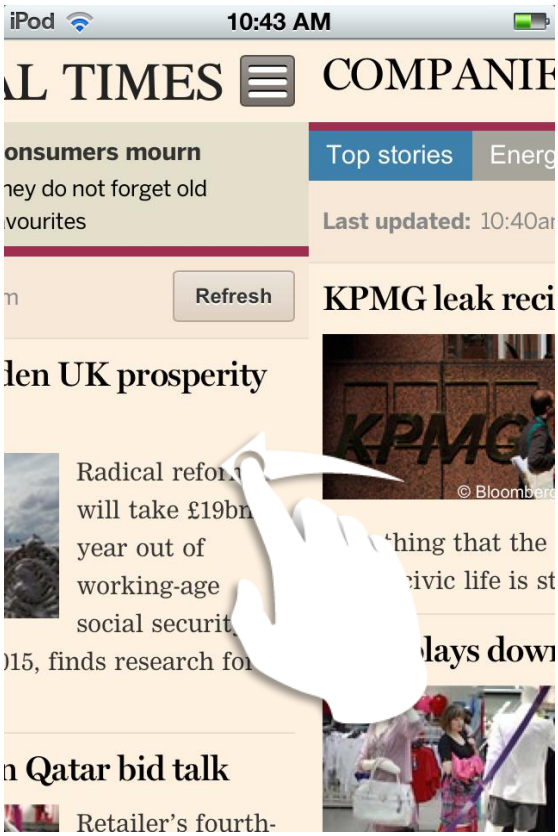

Figure 4. Swiping to the left on the Financial Times web app will move the user to the next news category. Image source: Screenshot taken with an iPod touch.

For the reader the biggest differences in form of content consumption is that the Financial Times require you to be registered as a subscriber in order to read the full articles. When the Guardian offers a free trial of 14 days. The Guardian has on the front page a lot more content directly visible than the Financial Times app.

Figure 5. Frontpage of the Guardian native app to the left and the Financial Times web app to the right. The Guardian only shows the titles and keywords of the articles, when the Financial Times shows the title and the first 100 characters of the article. Image sources: Screenshots taken with an iPod touch.

Figure 6. Article view. The Guardian to the left and the Financial Times to the right. Both applications have similar functionality except for the possibility to favorite and comment on articles in the Guardian. Image sources: Screenshots taken with an iPod touch.

5.3 Results

5.3.1 Questions

Do you know what a web app is?

Respondent 1: “No, not entirely sure. I think it is something that you click on to go into something like Google Play, like a link.”

Respondent 2: “No.”

Respondent 3: “Yes, it’s kind of just a web page that you can bookmark to your screen. The webpage is optimized to the device.”

Respondent 4: “Yes, it pulls in the mobile website from the web and shows it as an application on your phone.”

What do you like and dislike with the Financial Times web app?

Respondent 1: “I like that the articles are on the front page directly and you can read a short glimpse of what the article is about. I don’t really know how to use this, too much information confuses me.”

Respondent 2: “Firstly I don’t like how they have too much information on the front page. It would be better if they only showed the titles from the articles and then there would be the option to read more.”

Respondent 3: “It feels a bit weird, it takes a long time for it to load the news and images. I like how everything is on the front page, it’s quite clear. The sideways scrolling between the news categories is really cool. I would not notice that this is a web app, the only thing is that is a little bit slow. The biggest difference is that the app does not have a “home” button that apps usually have. I would not have realized that you can scroll through the news categories by swiping to the left or right if I would not have done it accidently. There is nothing that shows how the app itself works.”

Respondent 4: “Yeah it’s cool, the updated one looks nice. It’s good that it’s always up to date. It’s better than the Guardian app because it’s faster and does not crash as often. It looks more like a website which is good because people are used to how articles are displayed on their website.”

What do you like and dislike with the Guardian native app?

Respondent 1: “Because it’s less information on the page, it feels easier to navigate then the FT app. I know what to do.”

Respondent 2: “Everything is more clearer. I can choose between news categories easily with the buttons on the lower toolbar. Commenting and favoriting is unnecessary, I just want to read the news.”

Respondent 3: “I think it’s a good idea to have a 14 day trial. That will probably attract more people to buy the subscription.”

Respondent 4: “I like how the videos are also on the front page. I like to see news as a video. It is cool that you can favorite news articles and maybe later on continue reading them or showing some friends.”

Could you see yourself using any of these apps?

Respondent 1: “Yes I could, but I don’t really read the Financial Times or the Guardian.”

Respondent 2: “I don’t read the Financial Times, but yes it works all right so I guess I could use it on a daily basis if it would be a news source that I actually read.”

Respondent 3: “I would not read it daily, but if it would be something interesting like Mashable I probably would.”

Respondent 4: “Yes.”

What does an app store mean for you?

Respondent 1: “To find apps. See what is being mostly downloaded.”

Respondent 2: “Find apps.”

Respondent 3: “Find apps mostly. You can see what other people are downloading, what’s trending.”

Respondent 4: “I guess it means that you know the apps found there can be trusted. You get recommended stuff. It’s more like a browsing experience. With web apps you have to know what exist.”

Does an app store bring more value to the app?

Respondent 1: “Easier to update the apps”

Respondent 2: “Yes it feels safer.”

Respondent 3: “For me it feels like I would rather download the app from an app store than bookmark a web app on my home screen. It just feels like you can do more with a native app, like favorite articles and so on.”

Respondent 4: “Apps are more trustable.”

Would it make any difference for you to be browsing new apps through a web link or through an app store?

Respondent 1: “No. As long as the experience is good, then I will use it.”

Respondent 2: “I’d rather download it from an app store. I guess it’s more organized that way.
Respondent 3: “I don’t think it makes any difference. If both web and native apps can be bookmarked to the home screen then it’s equally fast to access them.”
Respondent 4: “it’s more like somebody has to tell you about the web app for you to find it. The app store is good for you random browsing around looking for apps.”

5.4 Summary

The interviews showed that normal smartphone users don’t know what a web application is. Respondents of the interview had heard about them, but mostly referred to them as bookmarks, links or mobile optimized websites. When asked about mobile web app stores, three out of the four respondents considered them to be very important for the future of web apps. Respondent 4 pointed out that without a digital distribution platform, the only way for normal consumers to find new and interesting mobile web apps would be via links shared through friends. Respondent 4 raised the point that if a web app store would not exist and web apps would only be shared as links, scam web app links could easily be generated by viruses and Trojans. A mobile web app store would feel more secure, respondent 2 answered.
All four interviewees thought that app stores are important because they feel secure and the applications found there are generally more reliable because of other user’s reviews and ratings.

All interviewees considered speed as one of the most important factors in a mobile application. Two respondents thought that web apps felt faster and more stable than native apps in general, and that an always updated app is more convenient than a native app that manually needs to be updated.
Regarding the UI and UX on mobile web apps, all respondents agreed that normal consumers would not notice any difference between a well developed web app and a native app. Respondent 3 and 4 especially liked the coherent UI between the Financial Times web app and their main website. They commented that it is good to have familiarities in form of coloring, navigation placement and the use of icons throughout all their medias.

The native Guardian application was considered by all interviewees to feel more stable than the Financial Times app. This was due to better performance, animations and elements more familiar with native applications.

The general response of the interviews show that even though web apps can feel a bit choppy, most of the time they were faster than native apps, and speed is something consumers value the most. As long as the web app looks good and the UX of the web app is good, respondents of the interviews would gladly use web apps over native apps.
The results from the interviews reveal that consumers have interest towards web apps and do not necessarily see them as a poorer choice to native apps, instead they found them useful and faster. For mobile web apps to be able to reach more consumers and let developers start monetizing on their apps, a mobile web app store of some kind has to be made.

6 CONCLUSION

6.1 Summary and findings

HTML5 has been hitting the headlines in various mediums since it was pre-launched in 2009, and that will continue to be the case the following years. The web technology is in a continuous development with new standards being released every two to four years.
Web apps are here to stay, but it will take some time before they will be more widely seen on mobile devices. The growth of PhoneGap powered mobile web applications will be steady and they will help boost the presence of web powered apps on consumer's devices. The technology is not yet at a state that it could compete with native technologies directly, but it is gaining fast in form of performance and functionality. A digital distribution platform is inevitable if mobile web apps are to attract more consumers and developers to start using them.
As the results from the interviews show, people would be ready to use mobile web apps over native apps as long as the experience is good or even better, this confirms the main hypothesis of this thesis.
Development environments and IDEs are continuously improving support for web apps and the PhoneGap framework, a good example is Dreamweaver with its BUILD extension.

Web apps will be able to compete with native applications in the foreseeable future. The hype is evident, a lot of people and corporations are backing the technology and there will be some major improvements and implementations to the framework which will enable the development of even more complicated web applications in the coming years.

Looking back at the choice of research method regarding consumers opinions, instead of conducting four interviews, a more wider spread survey would have brought more value to the research.

6.2 Ideas for further study

With HTML5 based operating systems like Firefox OS and Tizen due in 2013, it will be interesting to see how they will change consumers attitude towards web apps and how the development process of HTML5 will evolve. The Firefox OS is most certainly going to include an integrated mobile web app store that will be using their recently launched Web Payment JavaScript API, which will play an important part of HTML5's future.

REFERENCES

148 Apps. (2013). App store metrics. Available: http://148apps.biz/app-store-metrics/. Available: http://148apps.biz/app-store-metrics/ Last accessed 5.2.2013

Alex Chitu. (2010). Gmail to Use More HTML5 Features. Available: http://googlesystem.blogspot.co.uk/2010/06/gmail-to-use-more-html5-features.html Last accessed 4.3.2013.

Ali Saffari. (2012). Is Webview Slower than the Mobile Browser? (part 1: Canvas Graphics). Available: http://www.codefessions.com/2012/11/is-webview-slower-than-mobile-browser.html. Last accessed 12.3.2013.

Andrew Trice. (2012). PhoneGap advice on dealing with Apple application rejections. Available: http://www.adobe.com/devnet/phonegap/articles/apple-application-rejections-and-phonegap-advice.html. Last accessed 14.3.2013.

Andrew Trice. (2013). My Workflow for Developing PhoneGap Applications. Available: http://www.tricedesigns.com/2013/01/18/my-workflow-for-developing-phonegap-applications/. Last accessed 18.3.2013.

Android Developers. (4.3.2013). Platform Versions. Available: http://developer.android.com/about/dashboards/index.html. Last accessed 2.4.2013.

Apple Corporation. (2012). UIWebView Class Reference. Available: http://developer.apple.com/library/ios/#documentation/uikit/reference/UIWebView_Class/Reference/Reference.html. Last accessed 20.3.2013.

Apple Corporation. (2009). Core Graphics Framework Reference. Available: http://developer.apple.com/library/ios/#documentation/CoreGraphics/Reference/CoreGraphics_Framework/_index.html. Last accessed 26.3.2013.

Apple Corporation. (2013). All Categories: Most recent. Available: http://www.apple.com/webapps/. Last accessed 25.3.2013.

Ariya Hidayat. (2012). For pure #JavaScript, Chrome for iOS is 2x slower than Mobile Safari (iPad 2, iOS 5.1.1). Reason: no JIT #in UIWebView pic.twitter.com/vjANSCHI. Available: https://twitter.com/ariyahidayat/status/218558235530108928. Last accessed 20.3.2013.

Atrice. (2012). Questions and Myths About PhoneGap. Available: http://phonegap.com/2012/06/21/questions-and-myths-about-phonegap/. Last accessed 7.4.2013.

Christian Vasile. (2013). HTML5 Introduction – What is HTML5 Capable of, Features, and Resources. Available: http://www.1stwebdesigner.com/design/html5-introduction/. Last accessed 11.3.2013.

Compuware. (2012). Mobile apps: What Consumers Really Need and Want. (PDF). 1 (1), 11-12.

Erik Kay. (2010). The Chrome Web Store. Available: The Chrome Web Store. Last accessed 7.4.2013.

Frederic Lardinois. (2012). Google Bows To Apple: Chrome On iOS Will Use Apple's Slow UIWebView To Render Web Pages. Available: http://techcrunch.com/2012/06/28/google-bows-to-apple-chrome-on-ios-will-use-apples-slow-uiwebview-to-render-web-pages/. Last accessed 20.3.2013.

Frederick Hirsch. (2013). Device APIs Working Group. Available: http://www.w3.org/2009/dap/ Last accessed 20.2.2013

Global Intelligence Alliance. (2010). Native or Web Application? How Best to Deliver Content and Services to Your Audiences over the Mobile Phone. 1 (1), p1-38. Available: http://www.globalintelligence.com/insights-analysis/white-papers/native-or-web-application-how-best-to-deliver-cont/ Last accessed 7.2.2013.

Greg Avola. (2012). Creating apps with PhoneGap: Lessons learned. Available: http://www.adobe.com/devnet/phonegap/articles/creating-apps-with-phonegap-lessons.html. Last accessed 1.4.2013.

HTML5Rocks. (2013). Why HTML5 Rocks. Available: http://www.html5rocks.com/en/why. Last accessed 8.3.2013.

jQuery Mobile. (2013). How do I need to configure PhoneGap/Cordova?.Available: http://view.jquerymobile.com/1.3.0/docs/faq/how-configure-phonegap-cordova.php. Last accessed 30.3.2013.

Killian Bell. (2011). Steve Jobs Was Originally Dead Set Against Third-Party Apps for the iPhone. Available: http://www.cultofmac.com/125180/steve-jobs-was-originally-dead-set-against-third-party-apps-for-the-iphone/ Last accessed 28.2.2013.

Lauren Indvik. (2013). 'Financial Times' Relaunches Web App With 'Live' and 'Morning' Editions. Available: http://mashable.com/2013/04/03/financial-times-ios-web-app-update/. Last accessed 12.4.2013.

Mary Meeker. (2012). The state of the Web. 1(1), p1-88. Available: http://www.businessinsider.com/mary-meeker-2012-internet-trends-year-end-update-2012-12 Last accessed 4.1.2013.

Matt Asay. (2012). HTML5 isn't Facebook's 'biggest mistake'. Available: http://www.theregister.co.uk/2012/09/14/facebook_html_5_vs_native_apps/. Last accessed 1.4.2013.

Matt Burns. (2013). BlackBerry’s Android Bet Is Paying Off. Available: http://techcrunch.com/2013/03/27/blackberrys-android-bet-is-paying-off/. Last accessed 7.4.2013.

Matt Gemmel. (2011). Apps vs the Web. Available: http://mattgemmell.com/2011/07/22/apps-vs-the-web/. Last accessed 25.2.2013.

Matt Vickers. (2013). Making mobile work. Available: http://blog.xero.com/2013/03/making-mobile-work/. Last accessed 26.3.2013.

Michael Simon. (2012). Face Off: Chrome vs. Safari for iOS. Available: http://www.maclife.com/article/news/face_chrome_vs_safari_ios. Last accessed 20.3.2013.

Mobithinking. (2012). What you can learn from the FT Web app: interview with Steve Pinches. Available: http://mobithinking.com/ft-web-app-interview-steve-pinches. Last accessed 10.4.2013.

Mozilla Developers. (2013). Using the application cache. Available: https://developer.mozilla.org/en/docs/HTML/Using_the_application_cache. Last accessed 8.4.2013.

Netmarketshare. (2013). Market share reports. Available: http://www.netmarketshare.com/. Last accessed 26.3.2013.

Oracle Corporation. (2008). Understanding JIT Compilation and Optimizations. Available: http://docs.oracle.com/cd/E13150_01/jrockit_jvm/jrockit/geninfo/diagnos/underst_jit.html. Last accessed 20.3.2013.

Paul Irish. (2013). WebKit for developer. Available: http://paulirish.com/2013/webkit-for-developers/. Last accessed 26.3.2013.

PhoneGap. Events. Available: http://docs.phonegap.com/en/1.0.0/phonegap_events_events.md.html. Last accessed 28.3.2013.

PhoneGap. (2013). Getting Started Guides. Available: http://docs.phonegap.com/en/2.5.0/guide_getting-started_index.md.html#Getting%20Started%20Guides. Last accessed 1.4.2013.

Sydience. (2013). WebApps. Available: http://www.sydience.com/?p=56. Last accessed 4.4.2013.

W3C. (2012). Plan 2014. Available: http://dev.w3.org/html5/decision-policy/html5-2014-plan.html. Last accessed 13.3.2013.

APPENDICES

Question 1	Do you know what a web app is?
Question 2	What do you like or dislike in the Financial Times web app?
Question 3	What do you like or dislike in the Guardian Native app?
Question 4	Could you see yourself any of these apps daily?
Question 5	What does an app store mean for you?
Question 6	Does an app store bring more value to the app?
Question 7	Would it make any difference for you to be browsing new apps through a web link or through an app store?

Table 3. Interview questions