Native to HTML5: A Real-World Mobile Application Case Study

Rachel Gawley, Jonathan Barr, and Michael Barr

JamPot Technologies Limited, Mobile Computing Research Department,
405 Holywood Road, Belfast, UK
{rachel.gawley,jonathan.barr,michael.barr}@jampot.ie

Abstract. HTML5 is capable of producing media-rich and interactive webpages that function more like desktop applications than webpages. Increasing mobile browser support for the standard has enabled the development of mobile web applications as an alternative to native applications. This paper presents a project, which recreated a cross-compiled native application in HTML5. The experience provides an insight into developing mobile web applications and identifies some of the limitations associated with using HTML5 on mobile devices.

Keywords: HTML5, mobile applications, cross-platform.

1 Introduction

By the end of 2010 there were approximately 5.3 billion mobile subscriptions [1], which was equivalent to 77% of the world population. The sale of mobile phones is increasing with smartphones showing the biggest increase. In Q1 2011, smartphone sales were 85% greater than Q1 2010 [2]. The prevalence of the smartphone, especially in the developed world [1], is creating a culture where a mobile phone becomes an integral part of the user's life as it is also a media player, pocket gaming console, business tool, means to connect to the Internet and even a substitute laptop.

Mobile applications (apps) are also becoming an increasingly common feature in daily life with 43% of American mobile phone users having apps on their phone [3]. In the last 3 years, there have been over 350,000 applications developed [4]. These applications have been downloaded over 10.9 billion times in 2010 and it is predicted that the number of downloads will be 76.9 billion in 2014 [5]. Nevertheless, Gartner analysts [2] noted that

"Every time a user downloads a native app to their smartphone or puts their data into a platform's cloud service, they are committing to a particular ecosystem and reducing the chances of switching to a new platform"

This is an issue for both the end-user and mobile application developers. Ideally, end-users should have the same applications and associated assets available to them regardless of operating system/device. Consequently, application developers are increasingly required to redevelop the same application for different mobile platforms. This is not a trivial task as the skill sets required differ between the various mobile runtime environments.

The ultimate application development goal is to write an application once and deploy it to every screen. This includes desktop devices, tablets, mobile devices and even smart televisions [6]. No complete solution exists [7]. Nevertheless, there are mechanisms that enable the 'write once and deploy to multiple platforms' approach. This is referred to as cross-platform development and often produces cross-compiled code. The most common cross-platform approaches are:

- *Appcelerator Titanium* [8] – a native app builder for iOS and Android using web technologies, for example, HTML, CSS, JavaScript, PHP, Ruby, etc.
- *Mono* [9] – a software development kit that enables the deployment of .Net applications to Unix, Mac OSX, Windows (desktop and phone), iOS and Android as native applications.
- *Phone Gap* [10] – a hybrid app building platform that enables the creation of mobile applications for iOS, Android, BlackBerry, Windows and HP using standard web technologies, i.e. HTML, CSS and JavaScript.
- *Adobe Flash* [11] – an application-building environment used to produce native apps for iOS, Android, BlackBerry Playbook, Windows, Unix and Mac OSX.
- *Corona* [12] – a mobile development platform that enables the creation of native iOS and Android apps.
- *Unity3D* [13] – a development environment that enables the production of 3D game applications for Windows, Mac OSX, iOS, Android and games consoles.

Using a cross-platform approach can decrease development time as a mobile application is written once and deployed to multiple platforms as opposed to developing an individual application for each environment. The cross-platform approach can be used to produce applications that will be accepted into the various mobile applications stores, for example, Apple store, Android Market, Amazon Appstore for Android, BlackBerry App World, etc. Unfortunately, the app stores could, at any time, change their terms and conditions and no longer allow cross-compiled applications. In April 2010, Apple changed their developer agreement to only accept applications written in Objective C, C or C++ into their app store [14]. Any approach that creates final iOS machine code from other programming languages, for example, Adobe Flash, MonoTouch, etc. were no longer accepted. Apple, once again, changed their developer license agreement in September 2010, which relaxed the restriction on development tools used to create iOS applications [15]. Currently, cross-compiled applications are accepted into all major mobile application stores and therefore provide an excellent means of creating and distributing multi-platform mobile applications. However, the issue of app stores controlling the acceptance and distribution of mobile applications raises the question

"Whether the future of mobile applications is on the app store or with an alternative means of implementing and distributing cross-platform apps?"

The development of HTML5 and browser support for the evolving standards has created an environment suitable for developing cross-platform mobile and desktop applications that do not need an app store for distribution. Accessing applications via a browser is naturally cross-platform with desktop, tablets, laptops, mobiles, games consoles and even some televisions providing access to the Internet via a browser. However, HTML5 is an emerging technology as standards are currently being discussed [16]. Each browser uses an underlying layout engine (for example, WebKit, Gecko, Trident, etc.), which all have differing means of processing the HTML5 tags. This discrepancy between layout engines makes it more challenging to provide one solution for all screens. It has been shown that HTML5 provides many features required for media and content rich mobile applications [7]. The potential to write once and distribute everywhere, without an app store, means that HTML5 is a viable alternative to native mobile application development.

The aim of the project, described here, is to provide an insight into the practicalities and issues surrounding HTML5 mobile application development. This is achieved by replicating an existing native application created using cross-platform techniques in HTML5. This paper is structured to address the issues associated with creating an HTML5 mobile application and is organized as follows: section 2 describes the background to the project, including the specification of the existing mobile application; section 3 describes the techniques used to create the online HTML5 application; section 4 discusses the issues relating to going offline; section 5 presents a potential solution to the issue of creating a dynamic offline HTML5 application; section 6 compares the HTML5 app to the native application created using cross-platform techniques; section 7 presents and discusses some open issues relating to the future of HTML5 mobile applications; section 8 provides a brief conclusion.

2 Background

As part of a previous research project, a mobile application, ‘SoundBoom’, was created using cross-platform technologies similar to those mentioned in the introduction. The cross-platform techniques used in the development are proprietary to JamPot Technologies Limited and beyond the scope of this paper. However, the resulting application provides an excellent benchmark to compare other cross-platform techniques such as HTML5.

2.1 SoundBoom Requirements

SoundBoom was created as part of a project aimed at utilising cross-platform technologies to produce a media-rich application that also provides a compelling user experience. To provide a valuable insight, the resulting mobile application had to satisfy four high-level requirements:

- Contain media-rich content
- Utilise device capabilities
- Deliver dynamic content
- Provide the user with the ability to customise the application

2.2 Existing SoundBoom Application

The SoundBoom application was created, using cross-platform technologies, to meet the requirements documented in section 2.1 by providing:

1. A traditional soundboard that plays sounds when image icons are clicked, shown in Fig. 1.
2. A means of creating a custom soundboard using the device capabilities (camera, microphone and asset library).
3. The ability to save a custom soundboard to the cloud via web services.
4. The option to download a custom soundboard from the cloud thus providing dynamic content.

The resulting SoundBoom application was submitted to and is available on the following app stores: iTunes [17], BlackBerry App World [18], Android Market [19], Amazon [20] and Intel AppUp [21]. The application has also been deployed internally on the following platforms: MeeGo [22], Samsung Smart TV [6], WeTab [23], Mac OS X [24], Windows (XP, Vista, 7) [25].

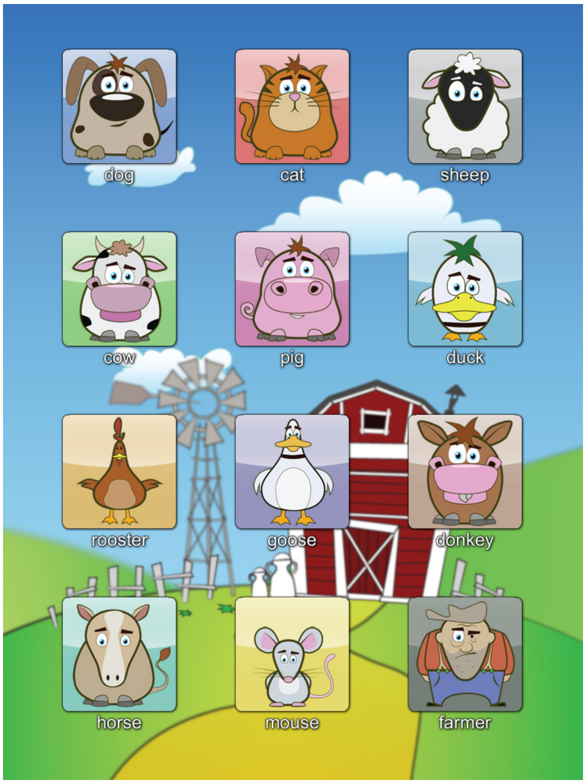

Fig. 1. Screenshot of SoundBoom Application

2.3 HTML5 SoundBoom Specification

The success of SoundBoom provided an ideal foundation to investigate HTML5 as an alternative approach to cross-platform mobile application development. The aim of the project, described in this paper, is to replicate the SoundBoom mobile application in HTML5. Unfortunately, HTML5 mobile applications cannot access device capabilities such as the camera or microphone. It is possible to use a small native application such as photopicker [26], which is launched from the browser, to provide access to the device camera and upload images to a server. As this solution requires a native application that is downloaded from an app store it cannot be utilised for the purposes of this project; it was decided that no native applications or techniques could be used in the solution. The creation of custom soundboards relies heavily on interacting with the device capabilities and, given the current state of HTML5, it would be difficult to recreate accurately the custom soundboard creation functionality. Therefore, the decision was made to create a read-only HTML5 SoundBoom application as version 1.0.

Version 1.0 will provide an interactive farmyard soundboard, which will play sounds when icons are clicked. The application will also provide a means to download custom soundboards, which have been previously created by the original SoundBoom application available on the app stores. Producing a read-only HTML5 version of SoundBoom is not a trivial task as the application is both media-rich and can be populated with dynamic content. The application will stretch some of the key features of the current HTML5 specification to its limits.

2.4 HTML5 as a Runtime Environment

HTML5 is the latest revision of HTML (HyperText Markup Language) and is still currently under development by both W3C [16] and WHATWG [27]. One of the main philosophies behind the development of HTML5 is to enable easier creation of web applications [28]. A web application looks and feels more like a desktop application, for example, a photo-editing tool, mapping utility, etc. Web applications often rely heavily on JavaScript or third-party plug-ins, for example, Adobe Flash [11], Microsoft Silverlight [29], etc. to embed media or enable users to interact with elements of the webpage. With the introduction of HTML5 standards, the traditional browser is evolving towards a runtime environment rather than just a means of rendering HTML.

HTML5 was not specifically created for a mobile environment but smartphones have evolved and most have sophisticated mobile browsers many of which are capable of parsing HTML5. This coupled with the following features make HTML5 mobile application development possible:

- Offline viewing – local storage and the application cache provide a means to store assets and information locally on the device and thus continual connectivity is not required.
- Rich media – <*canvas*>, <*video*> and <*audio*> tags provide a means of adding rich media content without additional third-party plug-ins.

The main problem associated with using HTML5 for mobile applications is the varying levels of browser support. Opera, BlackBerry and the iOS 4.2+ mobile browsers provide the best HTML5 support [30]. An additional issue is capturing and responding to touches on the screen as HTML normally responds to mouse-clicks rather than touch-events. Nevertheless, the current specification of HTML5 and mobile browser support is sufficient to create a read-only version of SoundBoom.

3 SoundBoom HTML5 Application

The aim of the project is to reproduce a read-only version of SoundBoom in HTML5 without any third-party plug-ins. Additional functionality will be provided via the well-known cross-browser JavaScript library, jQuery [31], and CSS3 [32].

The first stage of the project was to create the static part of the application. The term 'static' refers to the HTML pages of the application that utilise assets that are known on the first download/access of the application, even though the assets may be dynamic in when and how they are used in the application. The static part of SoundBoom is the farmyard soundboard, which consists of a background image and 12 images, each associated with an audio file. When an icon is clicked the associated audio file is played. The key features of the static application are:

- Image display
- Orientation specific layout
- Audio playback

3.1 Image Display

The *<canvas>* tag provides a 2D graphical area, which can have graphics, lines, text and animations, added in it. The canvas is particularly useful when creating detailed animations and is therefore suitable for game applications. It is very important to only use the canvas when another existing element will not suffice [33]. This is true for SoundBoom; the canvas element is not required as the image icons can be displayed with traditional *<img>* tag. The application needs to accommodate different screen sizes and in particular smaller screen sizes associated with mobiles. This is achieved using media queries in CSS3, shown overleaf, which enables style customisation based on the user's display [32]. The icons sizes are usually 150px, however, on small screens the size is halved to 75px.

The media query detects screens with a width less than 700px and sets the width of all the divs with class 'SBImage' to 75px. Each image icon is wrapped inside a div with the class 'SBImage'. The min-device-pixel-ratio is used to detect browsers on mobile devices, for example Safari on iOS, and scale the images accordingly.

CSS code to reduce image size

```
@media only screen and (max-width: 700px)
{
   .SBImage
   {
      width:75px;
      height:75px;
   }
}

@media only screen and (-webkit-min-device-pixel-ratio:
1.5), only screen and (-o-min-device-pixel-ratio: 3/2),
only screen and (min-device-pixel-ratio: 1.5)
{
   .SBImage
   {
      width:75px;
      height:75px;
   }
}
```

3.2 Orientation Specific Layout

Traditional webpages and web apps designed for desktop devices have a constant orientation, whereas smartphones are often designed to be used in either portrait or landscape. Therefore, HTML5 mobile applications need to accommodate the change in orientation and its impact on the layout of the application. This can be achieved using the *window.onorientationchanged* event, which is supported by WebKit, the layout engine used by the default browsers on iOS, BlackBerry and Android devices. The JavaScript function, *updateOrientation*, presented overleaf, is called both *onload* and *onorientationchanged.* The function changes the style of the application according to the orientation. When the orientation is 0 or 180, a portrait style is used and when the orientation is -90 or 90 a landscape style is used. Changing the style based on the orientation allows the best use of space in the application. Unfortunately, only WebKit supports the *onorientationchanged* event and orientation value, which means that other mobile browsers, mainly Opera mobile, will not be able to detect the event and change orientation.

Function to update layout to match orientation of device

```
function updateOrientation()
{
   switch(window.orientation)
   {
      case 0:$(".imgSB").removeClass("landscape").
      addClass("portrait");
      break;

      case -90:$(".imgSB").removeClass("portrait").
      addClass("landscape");
      break;
      case 90: $(".imgSB").removeClass("portrait").
      addClass("landscape");
      break;

      case 180:$(".imgSB").removeClass("landscape").
      addClass("portrait");
      break;

      default:$(".imgSB").removeClass("portrait").
      addClass("landscape");
      break;
   }
}
```

When this code was created the BlackBerry PlayBook had not been released. It was only when the application was tested on a PlayBook that a problem was discovered. The default orientation of the PlayBook is landscape opposed to portrait. Consequently, the orientation of 0 and 180 is actually considered to be landscape and not portrait causing the layout to be switched. A simple solution to this, shown below, is used to determine orientation based on width and height. If the width is greater than the height the device is in the landscape position and vice versa.

Function to update layout based on width and height

```
function updateOrientation(){
   var width = window.innerWidth;
   var height = window.innerHeight;

   if (width >= height)
      $(".imgSB").removeClass("portrait").
      addClass("landscape");
   else
      $(".imgSB").removeClass("landscape").
      addClass("portrait");}
```

3.3 Audio Playback

The HTML5 specification originally included an audio codec recommendation; however, Apple and Nokia objected and the codec requirement was dropped from the specification [33]. Consequently, each HTML5 engine uses different codecs and supports different audio files. The most commonly supported audio file types are MP3 and Ogg. MP3 is supported by WebKit based browsers and Internet Explorer, whereas, Ogg is supported by Mozilla. Ideally, both MP3 and Ogg formats would be supported in an application to ensure cross-platform compatibility. For the purposes of this project, it was decided to support MP3 format for the following reasons:

- MP3 files are smaller than Ogg files which makes downloading multiple audio files a better user experience.
- The default browsers on iOS, Android and BlackBerry devices support MP3. Opera mobile browser and Internet Explorer 9 also support the MP3 format.
- MP3 files are used in the native SoundBoom applications.

There is potential to update the application to accommodate other audio files when the HTML5 audio specification is standardised. The <*audio*> tag introduced in HTML5 enables audio files to be played in the browser without additional plug-ins. The code used in the SoundBoom application is shown below. The function plays a sound file from its relative location, which is stored in the *arrAudio* array. The JavaScript function has an integer, *soundPosition*, as a parameter, which indicates the position in the array of the audio file location. The *playSound* function is invoked from an *onclick* event associated with an image icon.

JavaScript to load and play audio files

```
function playSound(soundPosition)
{
   var audioclip;
   if (Modernizr.audio && (Modernizr.audio.mp3 != ""))
   {
      audioclip = arrAudio[soundFile]
      audioclip.load();
      audioclip.play();
   }
   else
      alert("Your browser does not support audio");
}
```

The JavaScript library Modernizr [34] is used to detect browser capabilities and thus provide a custom solution specific to the browser. The Modernizr script detects whether the browser supports the <*audio*> tag and is capable of playing MP3s. Only if these features are supported is an attempt to play the clip made, otherwise an alert is displayed.

4 Going Offline

Offline support is a key feature of the HTML5 specification that supports mobile application development. Offline support enables the download and storage of key assets to ensure the application will remain functional when a connection is not available [16].

4.1 Manifest and Application Cache

The manifest file associated with the HTML page lists the assets to be downloaded and stored locally in the application cache. The file is referenced in the *<HTML>* tag using *manifest="/name.manifest"*. The MIME type of the manifest file must be *'text/cache-manifest'*. The manifest file must begin with *'CACHE MANIFEST'* and include an UTF-8 encoded, line-separated list of assets. The SoundBoom manifest is shown below; for the sake of brevity, only some of the audio and image files are listed.

SoundBoom manifest listing

```
CACHE MANIFEST
# revision 47
CACHE:
js/soundboom.js
js/libs/jquery-1.6.2.min.js
js/libs/modernizr-1.7.min.js
img/h/barn.jpg
img/h/cat.jpg
img/h/cow.jpg
img/h/dog.jpg
...
error.htm
index.htm
sounds/cat.mp3
sounds/cow.mp3
sounds/dog.mp3
sounds/donkey.mp3
...
css/style.css?v=1
NETWORK:
js/blank.js
```

Comment lines in a manifest file start with #. The manifest, listed above, has a comment line with a revision number. Including a revision number comment provides a simple but effective means of ensuring application updates are propagated to devices. The offline cache will only be updated when the manifest changes and not when the files listed change. Within the manifest the inclusion of a revision number, and its subsequent increment when any of the files change, ensures the changes are downloaded when the application is accessed. The files listed under the

'CACHE:' heading are required for the application to work offline and will be downloaded and stored locally. All images, sounds, style sheets, scripts and HTML pages required for the application to run successfully offline are listed in the CACHE. The items listed under the *'NETWORK:'* heading are never cached; a network connection is required to access the files.

4.2 Offline Audio

Unfortunately, audio file caching is not well supported in HTML5 at present. It was observed that audio files listed in the CACHE were not downloaded. One potential solution is to use Base64 encoding to convert the audio file to an ASCII string format, which is stored in an XML file. XML files can be cached if they are listed in the manifest. This offline audio solution only works in desktop browsers, as mobile browsers on iOS and android devices do not support playing Base64-encoded audio. Mobile devices using the HTML5 application cannot play audio when offline.

The best overall solution is to play MP3 audio files when a connection is available and play Base64 audio strings when offline. This at least provides an offline desktop solution. The function [35], shown below, provides a means of determining the connectivity status, which is needed to implement the aforementioned solution.

Function to determine connectivity status

```
function checkNetworkStatus()
{
   if(navigator.onLine)
   {
      $.ajaxSetup({
       async: true,
       cache: false,
       dataType: "json",
       error: function (req, status, ex)
       {
         console.log("Error: " + ex);
         online = false;
       },
       success: function (data, status, req)
       {
         online = true;
       },
       timeout: 5000,
       type: "GET",
       url: "js/blank.js"});
       $.ajax();
     }
     else
       online = false;
}
```

There is a JavaScript function *navigator.onLine* that returns false if there is no Internet connection; however, it is unreliable and often returns true when it a connection is not available. Therefore, this alternative method of determining connectivity is used in the final solution. The JavaScript function *checkNetworkStatus* attempts to return a JSON object stored in the blank.js file. As blank.js is listed in the NETWORK section of the manifest it will not be cached. Therefore, if the JSON in the JavaScript file can be accessed it can be inferred that the device is online.

5 Dynamic Content: Cloud Interaction

A key feature of the original SoundBoom, shown in Fig. 2, is the ability to create user-generated soundboards, upload the assets of the soundboards to the cloud, which can then be downloaded to other devices and platforms running SoundBoom.

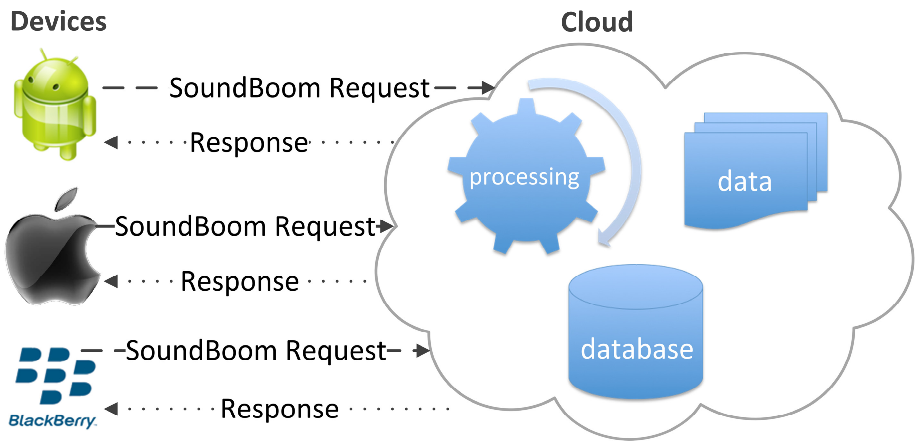

Fig. 2. SoundBoom cloud interaction on multiple devices

Version 1.0 of HTML5 SoundBoom is read-only; and enables users to download custom soundboards and their associated assets. Fig. 3 is a message sequence chart of the interaction between the original SoundBoom application and the cloud as a means of generating dynamic content. A user enters a PIN into the application and selects to load the soundboard associated with the unique PIN. A request is sent to the cloud; if there exists a custom soundboard associated with the PIN the details of the assets (absolute URLs) are returned to the application. A connection is required to make the request and download the assets. Once the assets have been downloaded they are stored for offline access within the application. Only when another custom soundboard is downloaded will the assets be replaced.

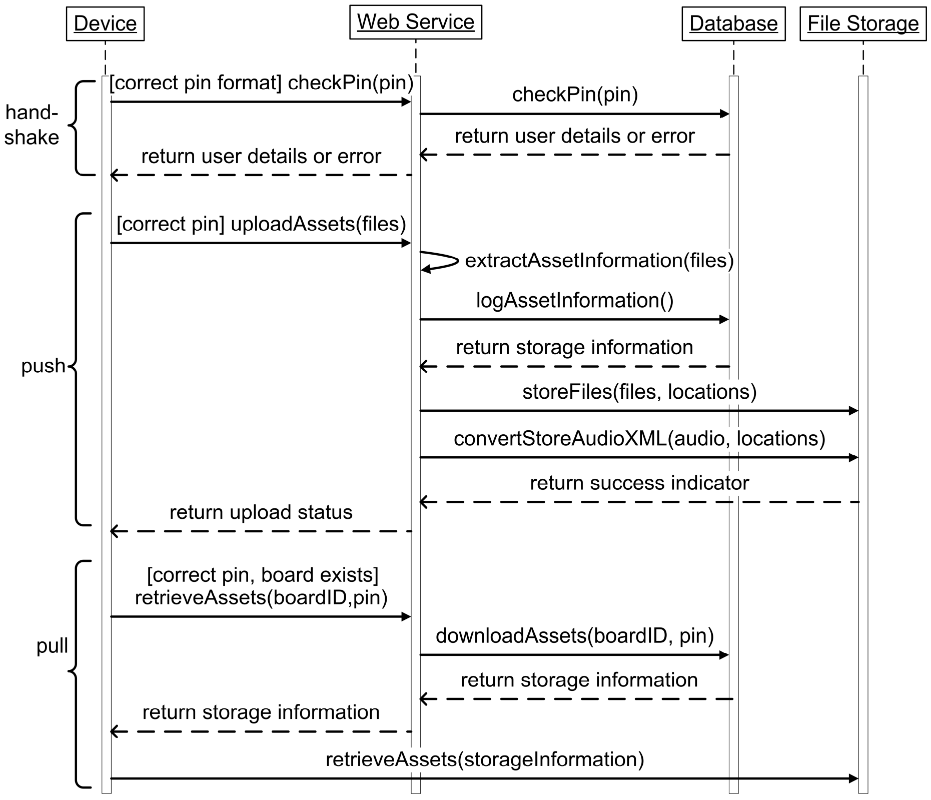
Device
Web Service
Database
File Storage
hand-shake
[correct pin format] checkPin(pin)
checkPin(pin)
return user details or error
return user details or error
push
[correct pin] uploadAssets(files)
extractAssetInformation(files)
logAssetInformation()
return storage information
storeFiles(files, locations)
convertStoreAudioXML(audio, locations)
return success indicator
return upload status
pull
[correct pin, board exists] retrieveAssets(boardID,pin)
downloadAssets(boardID, pin)
return storage information
return storage information
retrieveAssets(storageInformation)

Fig. 3. Message sequence chart of SoundBoom cloud interaction

The key to replicating the original SoundBoom is storing locally the assets of the soundboard for offline viewing. The issues relating to going offline have been discussed in section 4 but caching dynamic content has not been addressed. The problem with caching dynamic content is the fact that the assets are not known when the page is accessed and are not present in the manifest file when the page is loaded. A means of dynamically adding to the application cache after the page has loaded would be a good solution to the dynamic caching issue. The Mozilla browser has suggested the *moz-add* function to add files to the application cache; unfortunately, this is experimental and not part of the official specification.

There is no reliable means of dynamically adding to the manifest at run-time on the client-side, therefore, the problem must be solved on the server side (cloud). The solution implemented in the HTML5 SoundBoom application is shown in the message sequence chart in Fig. 4. When a user accesses the cloud with their unique PIN, a new manifest file is created on the cloud listing all the assets of the custom soundboard associated with the PIN. A new HTML file of the custom soundboard is also created and includes a link to the aforementioned manifest file. Both the HTML and manifest files are named using GUIDs to prevent users accessing the soundboards by guessing the naming convention.

The GUID is linked to a unique instance of a custom soundboard. If the unique soundboard has been accessed previously, the HTML and manifest files will already exist on the cloud and will not be recreated thus reducing the stress on the cloud. On successful competition of the request, the cloud responds with the location of the new HTML file, which is then loaded into the browser. When the page loads, the dynamic content is cached due to the listings provided in the new manifest file.

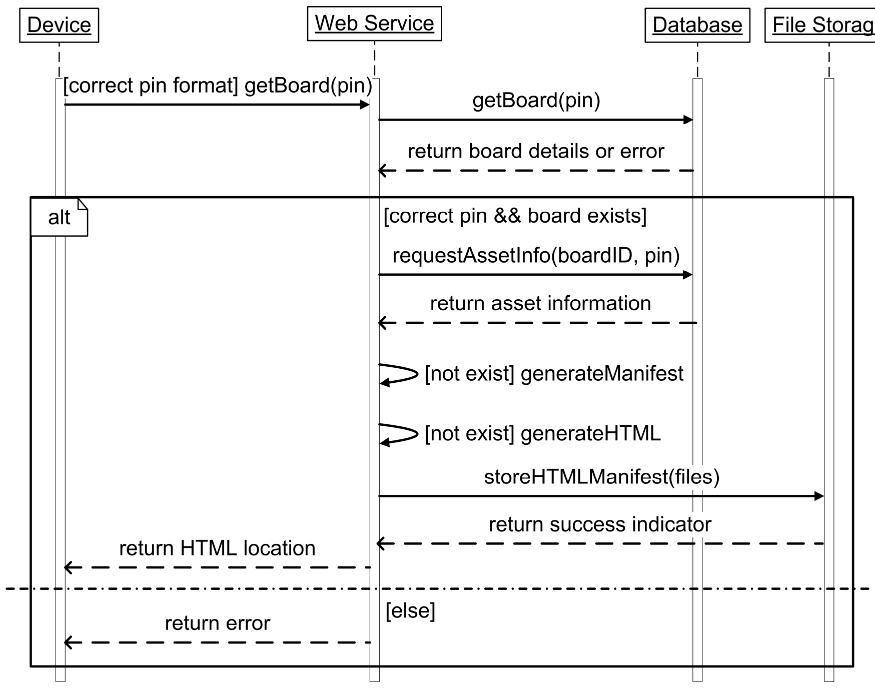

Fig. 4. Message sequence chart of manifest and HTML generation

6 Cross-Platform vs. HTML5

The aim of the project is to provide insight into developing cross-platform mobile applications in HTML5. The SoundBoom application had been developed previously using cross-platform approaches. The result is an application that was coded once, cross-compiled and runs natively on different platforms. Replicating SoundBoom in HTML5 provides an insight into using HTML5 as a means of developing mobile applications. Eight key features were used to compare the two development approaches: development time; cloud interaction; user-interface; media content (audio); multi-platform performance; offline performance; ease of distribution; ease of updating. One mark is awarded to the approach that satisfies the feature the best and zero to the other. If both approaches performed equally they are both awarded one mark. Table 1 presents the results and the overall total for each approach.

Table 1. Cross-compiled native applications vs. HTML5 applications

Comparison Feature	**Native**	**HTML5**
Development time	1	1
Cloud interaction	1	1
User-interface	1	1
Media content (audio)	1	1
Multi-platform performance	1	0
Offline performance	1	0
Ease of distribution	0	1
Ease of updating	0	1
Total	**6**	**6**

The cross-compiled native application and the HTML5 application have a total score of six. The development time for both approaches was approximately the same; the cloud services for the original SoundBoom were reused in the HTML5 version and this was taken into consideration when scoring development time. Interaction with the cloud was easily achieved with cross-platform techniques and HTML5. CSS3 and JavaScript facilitated the creation of a HTML5 interface that replicated the original SoundBoom design. Both approaches had the same issue of finding an audio format that provides a solution for all devices.

The application developed using the cross-platform approach worked consistently on all devices/platforms tested. The HTML5 approach worked on many platforms (iOS, Android, BlackBerry PlayBook, Windows and Mac OSX), however, it did not function consistently across all platforms due to variation in browsers. Once the custom assets were downloaded to the cross-platform SoundBoom there were always available even without connectivity. The HTML5 SoundBoom does not provide a truly offline application as audio files cannot be cached and Base64 encoded audio strings, which can be cached, cannot be played on mobile browsers.

HTML5 applications are very easy to distribute – the page is uploaded on to a server and is instantaneously available to any device with a browser. Cross-compiled applications need to be submitted to many app stores. It is only once the application has been accepted by a store that is available to the public. There is no guarantee that the application will be accepted and the process can take up to three weeks to complete. Similarly, updating requires the resubmission of the application to each store and the update will take time to process. Updating the HTML5 application requires updating files on a server and changing the manifest file. The update is available everywhere instantaneously and will automatically download the next time the user accesses the application when connected to the Internet.

In general, both approaches are comparable in development time, cloud interactivity, user-interface building and the ability to include media. Focusing on where the two approaches differ (one approach is identified as providing a better solution) is essential to providing insight into HTML5. The main areas where the approaches differ are: cross-platform performance; offline performance; distribution and updating of the application.

7 HTML5 Analysis

The SoundBoom mobile application was replicated in HTML5. The results indicated that HTML5 is a valid means of creating cross-platform mobile applications but there are limitations that need addressed. The main issues with HTML5 identified from the SoundBoom project are:

- Consistent cross-platform performance
- Consistent offline performance

These issues can be attributed to the fact that HTML5 is an evolving standard with differing runtime environments provided by browsers. HTML5 is more easily distributed when compared to native applications and this should be viewed as a positive feature. As it differs from the traditional app store approach a different business model is required. It is worth noting that the ease of distribution will affect the protection of Intellectual Property.

7.1 The Evolving Standard

HTML5 is an evolving standard under development by both WHATWG and W3C. The current state of WHATWG is in 'Draft Standard' and W3C is in 'Working Draft'. The specification of HTML5 is an on-going process; it will be many years before reaching the final W3C recommendation state. Even though the specification of HTML5 is not stable, all the major web browsers have committed to support it.

Browsers do not have a rigid specification to follow and therefore browser layout engines differ in their support for HTML5 tags. The differing browser layout engines make it difficult to provide a solution that works consistently across all devices. The HTML5 SoundBoom application targeted the WebKit implementation of HTML5 as it is the layout engine most frequently used by mobile devices.

The debate surrounding HTML5 has also caused previously included specifications to be dropped. For example, audio codecs that were supported were dropped and currently there is no codec specification in HTML5. Also, the delay in deciding on a specification has forced vendors to make a choice without knowing whether or not their solution will become part of HTML5 specification. For example, Safari, Chrome and Opera all use Web SQL [37] as a means of storing data that can be queried. In November 2010, W3C indicated that the specification was "no longer in active maintenance" and that the Indexed Data API is now supported. Three of the main browsers use a HTML5 database implementation that is no longer in the W3C specification. Developers will continue to use this functionality as long as browsers support it. These kinds of situations will continue to cause differences between browser support for HTML5 thus hindering the adoption of HTML5 for mobile application development.

7.2 HTML5 Mobile Application Business Model

HTML5 applications do not require an app store for distribution. The applications can be freely distributed on the Internet and just as easily updated. The ease of distribution is a positive aspect of HTML5 mobile applications but has far-reaching consequences. Distributing mobile applications via an app store simplifies the process of receiving payment and distributing the product. The app creator sets the price, adds a description, details, etc. and the app store does the rest. The app store even provides a target audience interested in mobile applications. Even though the commercialisation mechanisms have been provided by the app store the key to success is getting people to purchase the app.

The HTML5 mobile application distribution model and therefore business model is completely different. HTML5 applications are easily distributed on the web – anyone with an Internet connection can access an HTML5 application. Bypassing the app store and connecting directly to the consumer requires the app developer to assume the billing responsibility for their app. Of course, an ecommerce system could be built around HTML5 applications to require payment to access the application. However, the open-source nature of HTML5 would mean that it would be difficult to protect the code and stop it being copied and distributed freely. One alternative approach could be to embrace the easy distribution by making the application free and adopting a different business model. Another approach would provide free access to the app and require the user to subscribe to a paid service to unlock full functionality. For example, SoundBoom allowed the user to download a custom soundboard via a unique PIN. A user could pay a set fee or subscription and be provided with a PIN to access the customisation/download functionality. An ecommerce system to manage payment/subscription could be easily implemented.

8 Conclusion

The aim of the project, presented here, was to recreate a cross-compiled native application in HTML5. The resulting HTML5 application included much of the functionality of the original. The solution also included a method to download and cache dynamic content. However, there were two features the HTML5 application could not provide: playing audio offline on mobile devices; accessing device capabilities. The latter was not even attempted in this project due to lack of HTML5 support. For HTML5 to provide a real alternative to native application development these issues need to be addressed either by W3C and WHATWG or the browser vendors.

References

1. ITU press release, http://www.itu.int/net/pressoffice/press_releases/2010/39.aspx
2. Gartner press release, http://www.gartner.com/it/page.jsp?id=1689814
3. The Rise of Apps Culture, http://pewinternet.org/Reports/2010/The-Rise-of-Apps-Culture/Overview.aspx
4. Mobile App Internet Recasts The Software And Service Landscape, http://www.forrester.com/rb/Research/mobile_app_internet_recasts_software_and_services/q/id/58179/t/2
5. IDC study, http://www.idc.com/research/viewdocsynopsis.jsp?containerId=225668
6. Samsung Smart TV, http://www.samsung.com/uk/smarttv/
7. Melamed, T., Clayton, B.: A Comparative Evaluation of HTML5 as a Pervasive Media Platform. In: Phan, T., Montanari, R., Zerfos, P. (eds.) MobiCASE 2009. LNICST, vol. 35, pp. 307–325. Springer, Heidelberg (2010)
8. Appcelerator Titanium, http://www.appcelerator.com/
9. Mono, http://www.mono-project.com/Main_Page
10. Phone Gap, http://www.phonegap.com/
11. Adobe Flash Player, http://www.adobe.com/products/flashplayer/
12. Corona, http://www.anscamobile.com/
13. Unity 3D, http://unity3d.com/
14. New iPhone Developer Agreement, http://daringfireball.net/2010/04/iphone_agreement_bans_flash_compiler
15. Statement by Apple on App Store Review Guidelines, http://www.apple.com/pr/library/2010/09/09Statement-by-Apple-on-App-Store-Review-Guidelines.html
16. HTML5 Working Draft, http://www.w3.org/TR/html5/
17. SoundBoom iTunes, http://itunes.apple.com/gb/app/soundboom/id429903896?mt=8
18. SoundBoom BlackBerry App World, http://appworld.blackberry.com/webstore/content/38624?lang=en
19. SoundBoom Android Market, https://play.google.com/store/apps/details?id=air.ie.jampot.SoundBoomPro&feature=search_result#?t=W251bGwsMSwxLDEsImFpci5pZS5qYW1wb3QuU291bmRCb29tUHJvIl0
20. SoundBoom Amazon, http://www.amazon.com/JamPot-Technologies-Ltd-SoundBoom/dp/B004X6GU56/ref=sr_1_1?ie=UTF8&s=mobile-apps&qid=1303284579&sr=1-1
21. SoundBoom Intel AppUp, http://www.appup.com/applications/applications-SoundBoom
22. MeeGo, https://www.meego.com/
23. WeTab, http://wetab.mobi/en/
24. Mac OSX, http://www.apple.com/macosx/
25. Microsoft Windows, http://windows.microsoft.com/en-US/windows/home
26. iPhone Photo Picker, http://code.google.com/p/iphone-photo-picker/
27. WHATWG, http://www.whatwg.org/
28. Lawson, B., Sharp, R.: Introducing HTML5. New riders (2010)

29. Silverlight, http://www.microsoft.com/silverlight/
30. The HTML5 Test, http://html5test.com/results.html
31. jQuery, http://jquery.com/
32. CSS3, http://www.css3.info/
33. Lubbers, P., Albers, B., Salim, D.: Pro HTML5 Programming. Apress (2010)
34. Modernizr, http://www.modernizr.com/
35. Detecting offline status in HTML5, http://ednortonengineeringsociety.blogspot.com/2010/10/detecting-offline-status-in-html-5.html
36. Mobile Browser Stats for (January 2011), http://www.quirksmode.org/blog/archives/2011/02/mobile_browser_5.html
37. Web SQL Database, http://www.w3.org/TR/webdatabase/