Mobile web apps – the non-programmer's alternative to native applications

David Sin
School of Information Technology
Charles Darwin University
Darwin, Australia

Erin Lawson
School of Information Technology
Charles Darwin University
Darwin, Australia

Krishnan Kannoorpatti
School of Information Technology
Charles Darwin University
Darwin, Australia

***Abstract*—As the capacity of the mobile web continues to increase with the introduction of new web technologies, a growing number of "non-programming" developers are looking to web-based mobile applications (or "web apps") as an alternative means of creating and deploying content, without the need for code-based programming experience. This paper demonstrates just some of the capabilities of a web-based application built using HTML5 and CSS3. Although this paper focuses on the development process for two web app prototypes built exclusively for the iPhone platform, theoretically, a web application can be presented on other mobile devices with browsers that support HTML5 and CSS3 technologies. It was found that these web apps were capable of producing a comparable standard of user experience as their native application equivalents. The use of the latest web technologies provides "non-programming" developers with a viable alternative for developing apps on mobile devices.**

Keywords-web apps; iOS; mobile; application development; geolocation; CSS3; HTML5; JavaScript; jQuery; iPhone; 3D-transform.

I. INTRODUCTION

Mobile web applications have had a profound impact on the way we access information and communicate ideas. In particular, the area of application development has grown exponentially with the introduction of modern mobile and tablet devices. Recent trends in web development have shown much interest in the development of web-based apps, which are normally built using the HTML5 language in conjunction with CSS3 and JavaScript.

Web apps can be designed to have similar functionality to, and behave like, native applications, whilst still residing on the web. One of the benefits that web apps have over native applications is that they can be designed to be device-independent, and thus can be accessed on a number of different mobile platforms. Providing the mobile or tablet device has a suitable web browser (capable of rendering HTML5 and CSS3 – Mobile Safari, for example) then the application should function as intended across a range platforms. Additionally, the ability to mimic a native application's easy start-up via an icon on the device home screen is more likely to encourage repeated usage, rather than the desktop-originated method of saving and relocating a bookmark in the favourites list of a web browser.

Mobile applications can have an implicit number of purposes, from casual gaming, to data searches, to business-related operations and so on. One particular area that this paper will be looking further into is the design of a web application which utilises a map interface. Map-based apps in general are popular for their ability to present data in an interactive and efficient manner. Mapping applications are also now capable of location awareness with the recent introduction of geolocation capabilities. Designing such an app from the ground up would be a challenge, particularly for non-programmers; however, entities such as Google Maps have assisted in the development process by allowing public access to their APIs for building custom maps.

This paper explains how web development technologies such as HTML5, CSS3 and the jQuery Mobile JavaScript library were used to build a map-based web application prototype. It will also explain how geolocation services and the Google Maps API were used in the development of this mapping application.

II. METHODS (HTML5, CSS3 & JAVASCRIPT)

Two approaches were taken in the development of the web application prototypes. The first approach, given the remoteness of the mapped location in the Northern Territory and possible lack of internet connectivity, saw the application developed for partial offline access and without the use of geolocation. The second method used geolocation services with an interactive Google Maps

978-0-7695-4894-4/12 $26.00 © 2012 IEEE
DOI

interface. Both prototypes used a map interface to highlight places of interest at Charles Darwin University (CDU), and demonstrate the use of interactive and multimedia elements, such as audio and video.

In order to build a web application, it is necessary to understand how HTML5, CSS3 and JavaScript can be used in the development process.

A. HTML5

The recent HTML5 development standard has been modeled with a strong focus on mobile development. Features like offline storage, integrated multimedia and device hardware access are all new elements that can be used across mobile platforms. Whilst HTML5 is still considered a work in progress, the potential it has shown for web app development is likely to lead to an increase in the amount of web apps created in the future.

Currently there are a handful of mobile browsers that offer good HTML5 support, including the iOS default, Mobile Safari: "Mobile Safari on iOS devices like the iPhone and iPad, Opera Mini and Opera Mobile, as well as the Android operating system's web browser all provide strong levels of HTML5 and CSS3 support" (Goldstein et al. 2011).

B. Geolocation

Geolocation – one of the more highly anticipated features introduced in HTML5 – allows users to plot their current position on a map. Using geolocation, it is possible to create location-aware apps – already, this feature has seen many interesting and creative uses, including apps that allow people to "check-in" and meet up with other participants at an event in real time.

C. CSS3

CSS3 is responsible for the presentation side of web-based content, including web app content. The still-developing CSS3 standard has made it possible to create gradients, rounded corners, transitions, and many more effects that are pure CSS, without any use of images or JavaScript.

D. JavaScript

In relation to web app development, JavaScript plays a major part in application behavior and functionality. JavaScript frameworks, examples of which include jQuery Mobile and Sencha Touch, have enabled developers to mimic the functionality of native apps in their own web-based applications.

III. *CDU FLASHBACK* WEB APPLICATION PROTOTYPE - OFFLINE ACCESS, NO GEOLOCATION

A. "Back-end" development

1) Custom icons

Web apps can provide users with a customised icon to imitate the icons normally found on device home screens for native applications. Apple devices running iOS 4.2 and later offer support for detection of multiple icons for different device resolutions. This ensures that the icon with the most appropriate dimensions for the specific device is selected, preventing image distortion.

In order to specify multiple icon targets, each icon resolution must be linked in the header with the "sizes" attribute included. For example, in the *CDU Flashback* project, the following code has been implemented in the header of the index page:

1. Icon sized to 114x114 pixels for the iPhone 4 (high-resolution Retina display):

```
<link rel="apple-touch-icon"
sizes="114x114" href="img/h/apple-
touch-icon.png">
```

2. Icon sized to 72x72 pixels for the iPad (first-generation):

```
<link rel="apple-touch-icon"
sizes="72x72" href="img/m/apple-touch-
icon.png">
```

3. Icon sized to the default 57x57 pixels for iPhone 3GS and earlier (standard non-Retina display), iPod Touch and Android 2.1+ devices:

```
<link rel="apple-touch-icon-
precomposed" href="img/l/apple-touch-
icon-precomposed.png">
```

If the "sizes" attribute is not specified (as in the third code snippet above), the icon size defaults to 57x57 pixels. Additionally, naming an icon with the file name "apple-touch-icon-precomposed.png", will instruct

Mobile Safari to refrain from adding the standard gloss effect to the icon on iOS devices.

2) Splash screen

Similar to a native application, *CDU Flashback* has been configured to display a splash screen upon start-up from the home screen icon (Fig. 1). This has been made possible by invoking the full-screen mode in Mobile Safari, and by including the following code snippet in the header:

```
<link rel="apple-touch-startup-image"
href="img/l/splash.png">
```

3) Full-screen mode

In full-screen mode, the address and toolbars in Safari are removed from view to give the web application a truly native feel, and developers can take full advantage of the maximum screen area without needing to work around browser infrastructure (Fig. 2). Full-screen mode is triggered when the following code snippet has been included in the document header:

```
<meta name="apple-mobile-web-
application-capable" content="yes">
```

Additionally, the visual effect of the status display bar can be minimised by changing the colour and transparency using the following:

```
<meta name="apple-mobile-web-application-
status-bar-style" content="black">
```

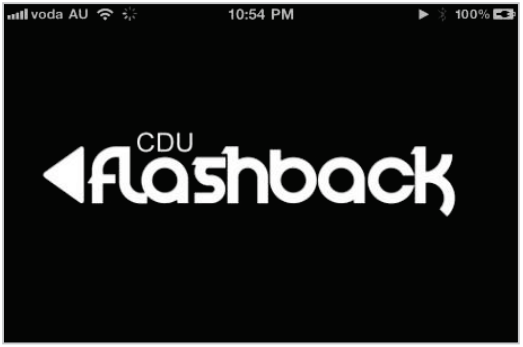

Figure 1. Splash screen on start-up of web app in full-screen mode from home screen icon

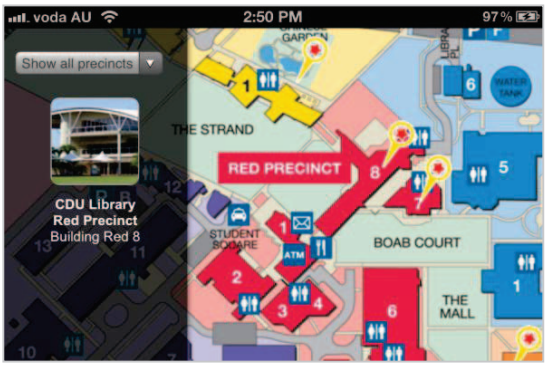

Figure 2. The static map interface presented after loading

4) Cache manifest file

A much-anticipated feature of HTML5 is the ability for web applications to store data on the user's device for offline access. This is achieved through the configuration of a cache manifest – a text-based system file that specifies a list of resources that a web application can have offline access to when disconnected from a network. Use of a cache manifest file also means that each time the user accesses the application, their device is not unnecessarily downloading core resources that do not need to be updated, and would otherwise consume valuable bandwidth. The manifest file is referred to by including the "manifest" attribute in the <html> element:

```
<html manifest="cache.manifest">
```

Included in the cache manifest for *CDU Flashback* are the icon and splash screen images, core JavaScript and CSS files, image files and the HTML pages. The cache manifest also includes a "network" section – referenced here are files that have specifically been defined as only being available if there is a network connection. The network section of the *Flashback* application references the audio and video files – this means that these large files cannot be stored on the user's device and require an internet connection to be viewed.

This feature is particularly useful when in remote communities where an internet connection may not be available. Additionally, this feature helps reduce data download costs.

B. User interaction

1) Mobile Bookmark Bubble

Upon arrival at the app home screen, users are presented with a popup dialogue bubble which prompts for them to add an icon to the home screen, just as a native application does (Fig. 3).

This has been achieved through the implementation of "Mobile Bookmark Bubble", an open-source JavaScript library available through the Google Code developer website which specifically targets the Mobile Safari browser used on iOS devices. In addition to providing the popup, Bookmark Bubble utilises HTML5's local storage capabilities to determine if the bubble has been displayed previously and/or dismissed, in order to prevent this extra feature becoming a nuisance to users.

2) "3D" panorama views

These pannable virtual scenes (partly shown in Fig. 4), do not require any additional plugins to view – instead they use a complex combination of CSS3 3D transforms and JavaScript to arrange 6 images into a three-dimensional cube. The "walls" are then rotated around a centre point by the user to navigate 360° around the scene. The framework code for this effect was sourced from the Apple Safari Technology Demos VR page (<http://developer.apple.com/safaridemos/vr.php>), and the demo images replaced with shots taken by the developer on the CDU Casuarina campus using a photo-stitching application.

3) "Flashback" photo slider

The "then-and-now" photo effect seen on the "Information Centre" *Flashback* page was achieved using JavaScript and CSS combined with a jQuery plugin to map the mouse events to equivalent touch events. Users can view a historical image overlay of the same location over a current image to see how it has changed over time. The "then" photo was sourced from an archived promotional poster, whilst the "now" photo was taken standing at a similar angle in the same area as the original photo (Fig. 5). Each image was placed in a nested div within a container div given the id "#photo", which the JavaScript plugin then uses to place one photo over the other and show/hide respective sections of the image as the slider is moved. Mapping touch events to the original mouse events means that users can tap along the photo length to move the slider rather than drag, which would normally cause the entire page to shift.

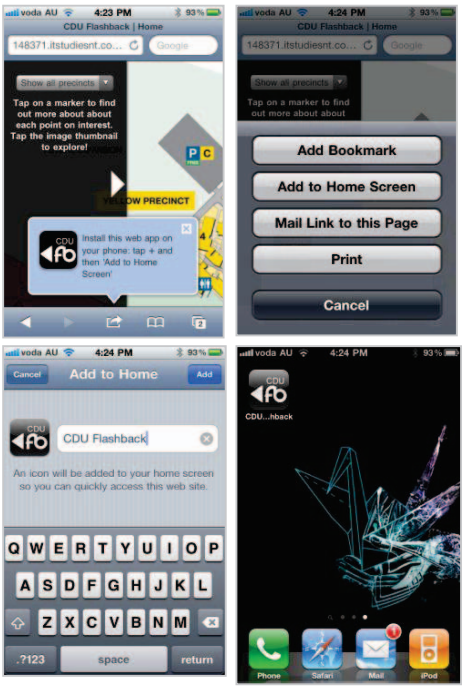

Figure 3. Installation of the home screen icon via the Mobile Bookmark Bubble

Figure 4. 3D panorama on the Indonesian Garden page

Figure 5. The "before-after" photo slider as seen on the Information Centre Flashback page

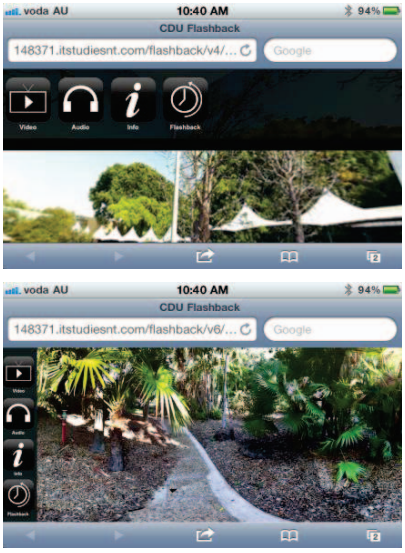

Figure 6. Previous versions of the location page layout – changes to which were as a result of informal user testing

C. User testing and acceptance

There was some informal user testing conducted for this application which resulted in a number of layout changes as shown in Figure 6. However, more extensive levels of user testing were carried out for the second prototype and are discussed in *Section IV: E* of this paper.

IV. CDU MyMap web application prototype – Google Maps interface with geolocation

The development process for the *MyMap* web app followed a process similar to that of regular website construction. The difference lay in what was required to optimize presentation on mobile devices. To ensure this web app functioned as intended on the target device (iPhone), many of the meta tags described in the outline for *CDU Flashback* were also included in the *MyMap* framework, as were a number of additional features relating to the use of Google Maps.

A. Map interface API – Google Maps

In order to offer users a more functional map interface, the *MyMap* developer chose to use the Google Maps API, which provided "a number of utilities for manipulating maps and adding content to the map through a variety of services, allowing one to create robust maps applications" (Google Maps, 2011). In the main, Google Maps API functions are executed via JavaScript and are relatively easy for non-programmers to implement.

Google Maps offers developers a number of options when creating custom maps, including zoom levels, map type (satellite, terrain, roadmap or hybrid) and where to position the map on start-up. Marker objects can also be positioned using basic x- and y-coordinates. The *MyMap* app also demonstrates the overlay of an image onto the map container, which will automatically resize itself upon changes in zoom level. It is also possible to create custom map buttons with their own defined functions, rather than relying on the default ones provided.

B. Google Maps with geolocation

In order to make a web app location-aware, there needs to be a way of obtaining the user's current coordinates on a map. Geolocation in HTML5 works by testing a JavaScript object named "navigator.Geolocation" (Mirkazemi, 2010). This object is used to determine if geolocation on the device is possible to begin with. When the user chooses to allow the device access to their current location, the positions for both latitude and longitude are passed into variables for centering the map.

C. JavaScript framework – jQuery Mobile

CDU MyMap uses the jQuery Mobile framework to structure the app's appearance and functionality, including buttons and page transitions. Once the foundations for the map interface were completed, work on the individual page elements began. A useful aspect of the jQuery Mobile framework is that the individual "pages", which are normally separate HTML files in desktop websites, are created in one single HTML file. jQuery Mobile takes advantage of HTML "div" and "id" attributes to identify different "pages" in a single file, allowing *MyMap* to differentiate between sections dedicated to the main map and pages for each place of interest. Due to the app's simplistic nature and mobile device target, it was beneficial to structure the content this way as it also reduces the amount of HTTP requests to the server.

D. HTML5 – audio and video

Implementing multimedia elements in *CDU MyMap* was made simple with the introduction of HTML5 audio and video tags. Embedding audio-visual media using <audio></audio> and <video></video> respectively will generate a browser-specific player linked to the source file provided in the tag. Not all mobile browsers support this feature as yet, but Mobile Safari (the default browser for the iPhone) will display these media players correctly onscreen (Fig. 7).

Figure 7. HTML5 audio and video players

E. *User testing and acceptance*

In any development project, it is vital to carry out usability testing – a process relating to the targeted end-users and their experience with the app in areas such as ease of use, efficiency and user satisfaction (Nielson, 1993). User testing assists in making decisions on whether a design rework is necessary – in fact, the tests that were conducted on the *MyMap* prototype lead to a number of improvements being made to the original design.

The tests that users performed on *CDU MyMap* related mainly to the map interface, and consisted of a number of tasks specifically designed to test for learnability, errors, efficiency and memorability of the interface (Fig. 8). A questionnaire was also distributed to help determine the level of user satisfaction in navigating the app.

When selecting users for the app testing process, it is important to ensure the tasks are performed by people who fit the targeted demographic. In the case of *CDU MyMap*, the testing group consisted of adult users with a reasonable knowledge of smartphone or touch device operation.

Test criteria were designed to indicate task completion rates, average completion times and users' thoughts about any difficulties in performing that task. Observations were made to ascertain if any tasks were carried out in ways that differed from the developer's intended use.

F. *Results*

After performing usability testing with a small number of users, the results and feedback were collated (Table 1).

The usability testing highlighted the following areas for improvement:

- Some button functions unclear
- Trouble identifying icons
- Number of touches to access information
- Hidden map controls at the bottom of the page
- Text alignment and buttons.

Positive feedback taken from the testing included that users successfully learned to use the map, help section and app shortcut functions. Following the testing, all interface components were made more obvious and were rearranged to fit within the standard iPhone screen dimensions (Fig. 9). The issues identified above were rectified, and the map markers were modified to take users directly to the corresponding information page once touched. Reverse navigation to the map interface was also simplified, with an omnipresent "back" button made available for each marker information page.

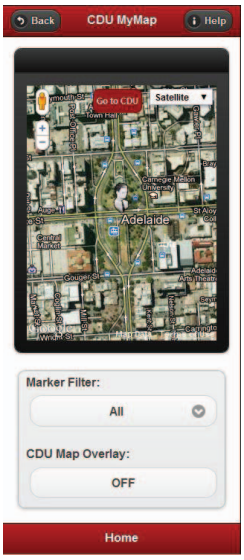

Figure 8. CDU MyMap home screen design

Figure 9. CDU MyMap interface redesign

TABLE I. WEB APP USER TESTING DATA

User Test Results		
	User 1	**User 2**
Scenario 1 – Learnability		
Completed?	Y	Y
Time Taken	1min 11sec	2min 16sec
Observations	User went straight to map page. Seemed to adjust to using the app quickly.	Didn't know how to scroll at first. Got accustomed to control by playing around. Opened the marker.
Scenario 2 – Errors		
Completed?	Y	Y
Time Taken	18sec	27sec
Observations	Easily knew to go to help for information.	Identified help. Map controls sub menu located.
Scenario 3 – Efficiency		
Completed?	Y	Y
Time Taken	2sec	26sec
Observations	Knew about the shortcut button from previous.	Understood that the shortcut was in the map screen.
Scenario 4 – Memorability		
Completed?	Y	Y
Time Taken	34sec	2min
Observations	Knew to look in the map options section. Clicked on video filter. Opened marker for video and used player.	Didn't understand what icons represented Trial and error approach to finding what each button did.

V. DISCUSSION

A number of possible approaches were considered for the development of the application prototypes, of which web-based apps was just one. Development of a web application was considered the most appropriate option because no specific tools or software were required; being similar to website development, the developers were able to work within their chosen software environment(s) across their chosen platform(s). The web application was limited to developing and testing solely on the iPhone; however, the opportunity for cross-platform expansion does remain if the project is progressed further in future.

A. Native vs. web-based apps

1) Advantages and disadvantages

From a web developer's perspective, the advantages of developing a web-based application are obvious. The technologies used to build web apps are already familiar to most in the industry, removing the need to learn multiple new programming languages. The developer can choose any development environment, from the most complex of web integration tools to the simplest of text editors.

Cross-platform compatibility issues are minor, as web apps will run and perform similarly in most mobile browsers, particularly as the majority now use the WebKit rendering engine. One-touch icon launching paired with full-screen capabilities mean that the general "look-and-feel" and behavioural aspects of native applications can be mimicked quite accurately, and as web apps are hosted on a server, there is no need to limit distribution to an often restrictive application store (Godwin-Jones, 2011).

However, this is not to say that web apps are the clear winners in all situations. Depending on the application content, intended target audience and functionality requirements, in some cases, development of a platform-native application would be necessary. A summary of the key benefits and challenges relating to the development of web-based applications is provided in Table 2.

2) Limitations of web apps

Whilst web apps can replicate a large amount of the behaviour of native platform apps, there are some areas where web-based technologies cannot compete with native functionality. Had the applications in this project required integrated use of the iPhone camera or gyroscope, for example, then native development would have been selected, as web apps are currently not able to interact with device hardware in the same way as native applications.

Similarly, the behaviour of web-based technologies cannot be guaranteed to be consistent across all mobile browsers. Not all modern browsers possess the same standard of HTML5 and CSS3 capability, and different rendering engines between mobile browsers can cause layout and performance discrepancies between devices. As an example, the CSS3 and JavaScript used to create the 3D panoramas in the CDU Flashback application will function correctly in both mobile and desktop versions of Safari, but will not display at all in the Mozilla Firefox desktop client, nor its mobile cousin, Fennec.

TABLE II. COMPARISON OF BENEFITS AND CHALLENGES RELATING TO WEB APPLICATION DEVELOPMENT

Benefits	Challenges
For consumers	
Available on all platforms / connected devices Easier discovery from centralized store front + user reviews	Lower usability due to UI design limitations Lower performance due to added browser layer Harder to discover cool apps without store front / user reviews (initially)
For developers	
Low development cost / high ROI – • simpler language (HTML / JavaScript) • shorter / cheaper debugging process • much larger potential reach Low maintenance cost Free from carrier / device manufacturer approval control / revenue share Integrated billing service	Limited HTML5-compliant browser support in the near term Performance-critical apps will likely remain local Unproven discovery (search) / billings support on mobile web Unproven monetization models – advertising / subscription / paid access / virtual goods? Dependent on wireless data connection – no 3G / Wi-Fi = no web application
For platform owners	
Competitive advantage from unique application ecosystems	No competitive advantage

(Morgan Stanley & Co. 2009, p. 163)

VI. CONCLUSIONS

The development of these app prototypes has shown how easy and effective web-based applications are to produce and implement. Making use of new HTML5 features like location tracking and integrated multimedia now enables web developers to create informative and interactive mobile experiences similar to that of native apps. By utilizing existing frameworks like jQuery Mobile and APIs such as Google Maps, building custom map applications for business or other purposes has become much simpler for non-programming developers.

The outcomes derived from the testing and development stages of this project have shown that when building for mobile, it is often better to take a more simplistic approach in terms of design and functionality. As web technologies continue to evolve and expand, it seems very likely that web-based applications will become a more mainstream development platform for future app developers.

The *CDU Flashback* and *MyMap* prototypes demonstrate just one type of app experience that can be achieved by leveraging the skills that those in the web profession are already well familiar with. Developing within the familiar frameworks of HTML5, CSS3 and JavaScript removes the time and skill factors that would be needed to build a platform-native version, and provides developers with increased cross-platform potential.

What has been achieved in this project are working prototypes that not only serve as demonstration models for an intended native iPhone application, but could also, with a few further refinements, be deployed in their current state as web apps to users on a wide range of mobile devices. Depending on the content and intended distribution method, in many cases a web application may be a viable, even preferred option for development. Indeed, it is certainly the case in situations where an experienced native programmer is not available, or when wanting to present a more accurate hands-on experience during the prototyping stage.

REFERENCES

[1] Godwin-Jones, R. 2011, "Mobile apps for language learning", Language, Learning & Technology, vol. 15, issue 2, pp. pp. 2–11, viewed 25 July 2011 via Academic OneFile

[2] Goldstein, A, Lazaris, L & Weyl, E 2011, *HTML5 & CSS3 for the Real World*, 1st edn, SitePoint, Victoria Australia

[3] Google Maps, 2011, *V3: The Solution for Maps Applications for both the Desktop and Mobile Devices*, viewed 5th December 2011, <http://code.google.com/apis/maps/documentation/javascript/>

[4] Mirkazemi, A 2010, *HTML5 Apps: Positioning with Geolocation*, viewed 5th December 2011, <http://mobile.tutsplus.com/tutorials/mobile-web-apps/html5-geolocation/>

[5] Morgan Stanley & Co. 2009, The Mobile Internet Report, accessed 17 August 2011, <http://www.morganstanley.com/institutional/techresearch/pdfs/mobile_internet_report.pdf>

[6] Naraine, R. 2007, "How Far Can Web Apps Take the iPhone?", PC World, vol. 25, issue 10, pp. 18-20, viewed 25 July 2011 via ScienceDirect