Analysis of Current and Future Web Standards for Reducing the Gap between Native and Web Applications

Rocío Rodríguez[1,2], Pablo Vera[1,2], Federico Vallés[2], María Roxana Martínez[1], Daniel Giulianelli[2]

[1]Interamerican Open University (UAI)
Higher education center in information technology
City of Buenos Aires, Buenos Aires, Argentina
{RocioAndrea.Rodriguez; PabloMartin.Vera; Maria.Martinez} @uai.edu.ar

[2]National University of La Matanza
Department of Engeniering and Technological Research
San Justo, Buenos Aires, Argentina
{fvalles; dgiulian}@ing.unlam.edu.ar

Abstract. New web standards are emerging day after day. Some of them are crucial in the improvement of web application for mobile devices. From visualization issues to hardware access, all this new tools can help developers to create native like applications on web environments. HTML5 and CSS3 with css media queries are some examples that are already available to use. Some other standards, such as W3C Devices' APIs, are still in progress but nearly to arrive. This paper shows existing and future standards for mobile web.

Keywords: MOBILE WEB, NATIVE, HTML 5, CSS3, JQUERY MOBILE

1 Introduction

When an application for a mobile device must be develop, the big question is: should we create a native or a web application? Of course the answer is not simple. It is relative to the requirements of that application. Anybody will say that Native applications are more powerful than web applications and it´s true. But new standards are emerging to reduce that gap. Table 1 shows the capabilities of each type of application.

Table 1. Native App vs Web App

	Native App	Web App
Connectivity	Online and offline	Mostly Online but can be offline
Hardware Access	Full	Limited (but growing)
Graphics	Can use all hardware capabilities to create graphical stunning applications like	Limited to the visualization supported by the browser.

	games.	
Look and Feel	Access to native controls allows creating standard applications.	With the use JavaScript frameworks, native controls can be simulated on the browser giving the user the feel of being using a native application.
Portability	Only on the same operating system and restricted by the supported version.	Can be used on any device with any operation system supporting web standards

Next section analyzes each of the features showed on table 1 for mobile web applications. With the goal of analyzing new possibilities that allows reducing the gap between web and native applications. Section 3 shows some related works and the final section present conclusions and future work.

2 Mobile Web Applications

This section will show some strategies for creating more powerful mobile web applications using some new standards.

2.1 Connectivity

Of course web applications were born to be connected but now they can also have very powerful offline capabilities. HTML5 includes a new way of handling cache for web application using a manifest file. This file can control the way each file is treated. Manifest file includes three sections:

- Cache Manifest: Files to be cached
- Network: Files to be always downloaded from the server
- Fallback: Files to be shown if no connection if available

```
An example cache manifest can be:
    CACHE MANIFEST
    # 2012-02-21 v1.0.0
    theme.css
    logo.gif
    main.js
    NETWORK:
    login.asp
    FALLBACK:
    /html5/offline.html
```

Line starting with a # is a comment. It´s a good practice to include the update date of the file so each time the file change the server will download all pages and replace it´s cache. So comments can be used to force the download of new version of the pages.

`AppCache` can be used to create web applications that works completely offline, but with the limitation that if the user clears the browser cache it will also lose access to the application until being online again.

2.2 Storage

Web applications usually store data in the cloud using some server side scripting. But some applications needs to store data locally. HTML5 includes new storage capabilities with `WebStorage` feature.

`WebStorage` allows keeping large volumes of information without affecting the website performance in contrast to cookies where the information travels from server to client on each request. There are two bags to store data:

- `SessionStorage`: keep data while the http session is alive.
- `LocalStorage`: keep data permanently on the device or until the user explicitly deletes browser data.

2.3 Hardware Access

Web applications cannot access all hardware of the mobile devices, but new tools are emerging to be able to do so. A clear example is geolocation capability of HTML5. It allows getting the user current position by means of wireless network and the GPS if it's available.

User location can be determined by using the function `navigator.geolocation.getCurrentPosition`.

Keeping user position when moving can be also made with HTML5. It includes the function `navigator.geolocation.watchPosition` which track user location changes in an asynchronous way. This function has a parameter to enable High Accuracy of geo location that will force to use the device GPS to get a more accurate position.

A set of new standards are being developed to access device's hardware, some of them are:

- Motion Sensors: This specification provides DOM events for retrieving information that describes the physical orientation and motion of the hosting device. Access to attributes like alpha, beta and gamma coordinates can be achieved by registering the `deviceorientation` event on the window object.
- Battery Status: This API is used to determine the battery status of the hosting device. By the creation of the `BatteryManager` object called battery, it is possible to obtain different battery's attributes by adding disparate events to the battery object such as: charging status by adding the `onchargingchange` event, `chargingTime` by appending the `onchargingtimechange` event, `dischargingTime` by attaching the `ondischargingtimechange` event, and the battery level by adhering the `onlevelchange` event.

- Proximity Sensors: This API enables the programmers to access the proximity sensor of mobile devices, this way, it is possible to determine if the mobile device is close to a certain physical object. To achieve this, the API provides an event called `userproximity`, which has to be added to the window object. This is implemented on the `UserProximityEvent` interface whose attribute near (boolean value) is the one that contains the proximity information.
- Ambient Light Sensors: This API makes possible the ambient light measure by accessing the light sensor of a mobile device. To accomplish this, the API provides an event called `devicelight`, which has to be added to the window object. This is implemented on the `DeviceLightEvent` interface whose attribute value (double value) is the one that contains the ambient light value measured in lux. It is important to be aware that "the precise lux value reported by different devices in the same light can be different, due to differences in detection method, sensor construction etc."[1]
- Vibration: The Vibration API enables the access to the vibration mechanism of the hosting mobile device. The Navigator interface possesses a method called `vibrate()`, this method, when invoked makes the mobile device vibrate depending on the value it receives. If the value passed to `vibrate()` is 0, it cancels any existing vibrations.
- Atmospheric Pressure Sensors: This API provides information on atmospheric pressure. The mobile device must be equipped with an atmospheric pressure sensor. The `AtmPressureEvent` interface provides web developers information about the atmospheric pressure levels measured at the hosting device. [2]
  This is achieved by interrogating a barometer or similar detectors of the hosting device.
  Some applications are: a) Altitude detection making use of the relationship between changes in pressure relative to altitude. b) Climatological that use barometric pressure to predict weather conditions like rain is coming. c) Meteorological information.
- Ambient Temperature Sensors: This API provides information on room temperature. Device Measures the temperature in degrees Celsius (°C). The common use is to control the temperature.
- Humidity Sensors: This API provides information of the humidity sensor which measures the relative humidity in percent (%).
- Camera and Microphone: The API, `navigator.getUserMedia()` allows applications to access the microphone and camera user. With `navigator.getUserMedia()`, we can connect to the microphone input and the web camera without any supplement. Access to the camera can just make a call without having to install anything. The data is processed and sent directly to the browser.
- NFC: There is an API to access the hardware subsystem and achieve near-field communications (NFC). Some use cases named in W3.org the API are [3]:
    - Tap to Play: tap your device to another to play a peer-to-peer game, using the `NFCPeer` interface to exchange `NDEF` messages.

- Tap to Share: tap to share some data, e.g. coupons, contacts, using the `NFCPeer` interface to exchange `NDEF` messages.
- Tap to Control: tap to control another device, like a TV remote, using the "handover" capability of the `NFCPeer` interface.
- Tap to Connect: tap to connect via WiFi or Bluetooth, using the "handover" capability of the `NFCPeer` interface.
- Tap to Read: tap to read NFC tags, using the `NFCTag` interface.
- Tap to Write: tap to write NFC tags, using the `NFCTag` interface.

At the moment, Mozilla Firefox is the only browser that supports most of the Devices' APIs developed by the W3C. A detailed explanation and current status of each of the features can be found at [4].

Other features are being developed that will allow accessing user information available in the mobile device like Address book and Calendar Data [5].

2.4 Graphics

HTML 5 includes CAVAS object which creates a drawing area. Inside this are different types of objects can be drawn (circles, rectangle, lines, curves, text, etc). This objects can also be animated. There are some that makes easier the task of using CANVAS, a example is JCanvaScript (http://jcscript.com).

2.4.1 WebGL (Web Graphics Library)

Is a JavaScript API that is used for rendering interactive 3D graphics and 2D graphics within any compatible web browser without using plugin.

WebGL brings 3D graphics to the Web by introducing an API that conforms to OpenGL (ES 2.0) that can be used in `<canvas>` HTML5 elements.

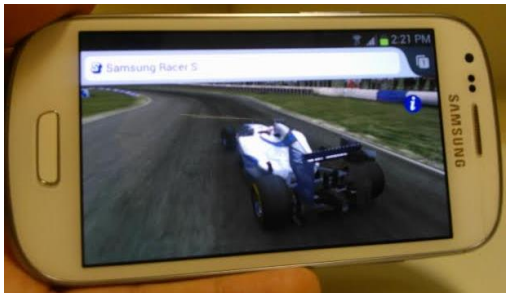

Fig. 1. WebGL Example – Demo http://helloracer.com/racer-s/

2.4.2. Scalable Vector Graphics (SVG)

It is a modular language for describing two-dimensional vector graphics and vector / raster mixed in XML. [6]

It is a good option to add high fidelity visual. Visuals will be easily scalable for both small and simple applications to complex applications in a website without the need

for a plug-in or standalone viewer. Examples of SVG are in the following link W3schools. [7]

2.5 Look and Feel

2.5.1 Responsive Design

A key feature of HTML is the ability to adapt to screen of the device being used, from relative units to complex CSS, it allows creating a unique site that will fit different types of screens. CSS Media queries allows adapting the interface regarding the screen resolution so for example if the site is being used in a tablet it could show more controls if it´s being used in a cell phone. This is called Responsive Design [8], [9]. Some frameworks like JQuery Mobile [10] includes responsive controls available to be used. Figure 2 shows an example of a responsive menu. Section A shows how the menu is expanded in high resolution screens. Section B shows that in a narrower screen the menu will be collapsed and an icon will appear. Section C shows the expanded options of the menu when clicking the icon.

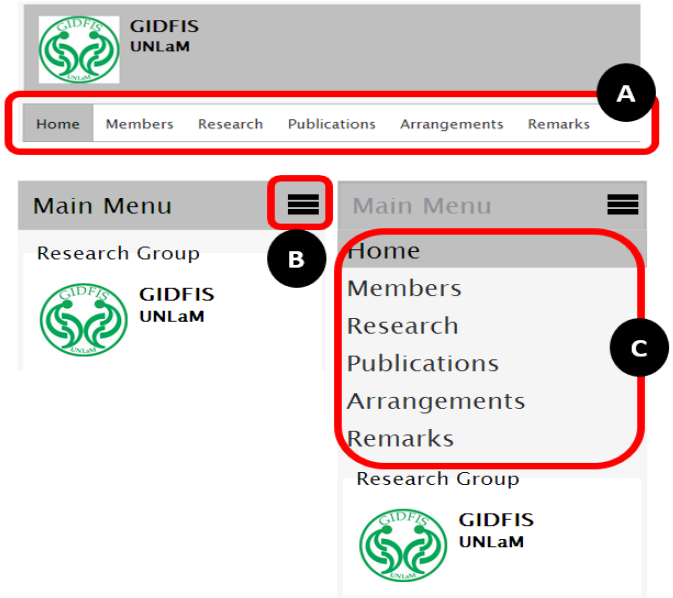

Fig. 2. Example of responsive menu built with css media queries and it´s visualization regarding the screen resolution

There are several ways of using CSS Media Queries [9]:

- Including a query in the link to a css file

```
<link rel= "stylesheet" tyte= "textc/css"
 media= "screen and (max-device-width: 480px)"
href= "customstyle.css" />
```

In this case the css will be applied to the page only if the screen width is lower than 480 pixels.

- Including queries inside the css file.

```
@media screen and (max-device-width: 480px)
{
 .column
   { float: none;}
}
```

- Using an import directive including the query

```
@import url ("customstyle.css") screen and (max-
device-width: 480px);
```

2.5.2 Device Specific user interfaces

Web applications can look like native applications thanks to numerous JavaScript frameworks already available. For example IUI [11] imitates Iphone user interface (figure 3).

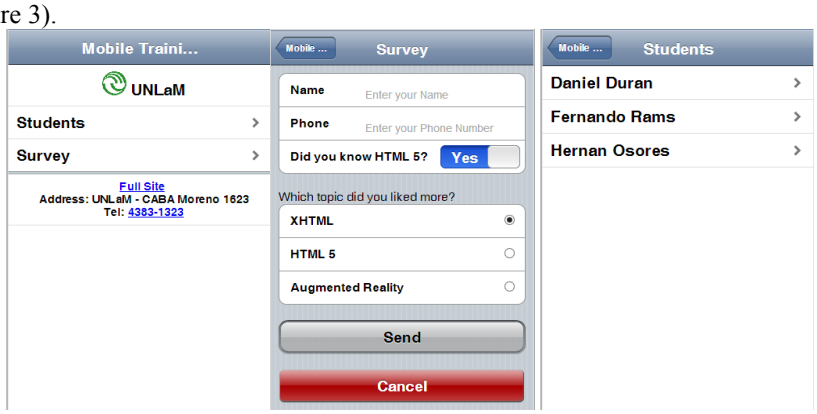

Fig. 3. Examples of a web application developed with IUI

Another usefull framework is JQuery Mobile [10] wich includes a custom framework for mobile applications, with serveral controls, including responsive design. Figure 4 shows some web pages made with jquery mobile.

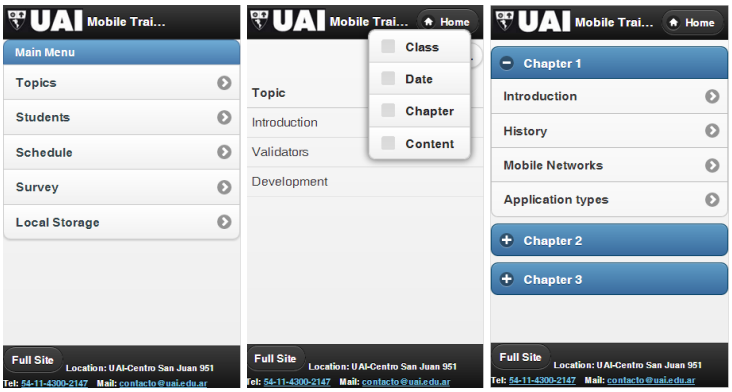

Fig. 4. Examples of a web application developed with JQuery Mobile

Several other javascript frameworks are available some are more powerfull than others, but all of them use javascript and css to generate a rich user interface.

2.6 Portability

The big advantege of web applications is portability. Web applications runs over web browsers so using standards allows developing a unique web application that will run on any existing operating system. But we also need procesing power. New smartphones are powerfull enough to run complex javascript functions, process complex css layouts and access hardware throw new device apis fast enouth to be usefull in or applications. In contrast a native aplication is programed for a particular operating system and also with compatibilities issues over the operating system version, so it`s more difficult to reach different kinds of devices. A third alternative are hybrid applications that generally uses some propietary programming language for building the web application and then allows packing or migrating that application for different platforms [12].

3. Related Work

Several authors are considering that web applications are getting closer to native applications, for example INTEL [13] analyses the development capabilities of the different mobile operation system and concludes that HTML5 is the best choice because it´s common for all of them. So they recommend using it in conjunction with some JavaScript framework to develop a simple application targeting server devices.
CHARLAND Andre, LEROUX Brian [14] compares web and native applications, remarking pros and cons of each technique. Remarking the increasing power of web applications also believing that web gaming will be also possible in mobile environments thanks to the growing implementation of WebGL [15] (a 3D rendering environment for browsers).

CHRIST Adam [16] remarks the advantages of developing web applications over native applications. It also remarks the power of javascript frameworks to create the user interface of new web applications.

FLING Brian dedicates a chapter of his book "Mobile Design and Development: Practical Techniques for Creating Mobile Site and Web Apps" [17] to compare web and native applications. He remarks that if it´s possible for the porpoise of the solution web applications should be preferred over native applications: "I`m a big fan of native application and I feel that there are a lot of great innovative and market opportunities here, but mobile web apps are the only long-term viable platform for mobile content, services, and applications."

4. Conclusions and Future Work

The gap between capabilities of native applications over web applications is narrowing, especially boosted by hardware access web standards. Besides developing a web application ensure it to be multi-platform in contrast to native applications that requires multiplying efforts for developing on each different platform for each mobile operating system. Thus also keeping updated web applications is simpler than maintaining native applications.

Regarding look and fell, there are several JavaScript frameworks that can help web applications to look like native applications, showing native like controls in a web page. Nowadays it begins to be complex distinguishing between a native a web application. Style sheets can now be conditionally applied with CSS Media Query, so the design can match the device screen creating a more usable interface, taking advance of bigger screens and giving a good user experience on small screens without having to develop several web sites.

We are convince that the future is on having applications in the cloud, where user can access then from different devices without losing previously loaded information. HTML5 allows keeping information on the device when it´s disconnected and sync it when internet connection is available.

All this features together, with hardware access capabilities, opens a wide range of opportunities of developing more powerful web applications.

Initially applications were almost entirely native, then began considering hybrid applications (those applications built in a common framework and then compiled in different pre-defined operating system, more powerful than web application but not as powerful as native applications). Nowadays different authors share our view that the future is in web applications.

References

1. W3C. "Ambient Light Events" (2014). http://www.w3.org/TR/ambient-light/#device-light
2. W3C. "Events atmospheric pressure" (2014). http://dvcs.w3.org/hg/dap/raw-file/tip/pressure/Overview.html

3. W3C. “WebNFC API.” (2014). http://www.w3.org/TR/nfc/
4. W3C. “Standards for Web Applications on Mobile: current state and roadmap - Sensors and Hardware Integration” (2013) http://www.w3.org/Mobile/mobile-web-app-state/#Sensors_and_hardware_integration
5. W3C. “Standards for Web Applications on Mobile: current state and roadmap – Personal Information Managemente” (2013) http://www.w3.org/Mobile/mobile-web-app-state/#Personal_Information_Management
6. W3C. “API SVG Simple”. http://www.w3.org/TR/SVG/
7. W3SCHOOLS. “SVG Examples”. http://www.w3schools.com/svg/svg_examples.asp
8. GARDNER Brett S. “Responsive Web Design: Enriching the User Experience”. Sigma noblis, Volume 11, Number 1, pp.13-19 (2011). http://www.noblis.org/media/2dd575c1-2de9-4d92-9bdb-f72ad9fb9a19/docs/SigmaDigEco2011 pdf
9. MARCOTTE Ethan. “Responsive Web Design. A List Apart Magazine: Articles”. (2010). http://www.princeton.edu/~mlovett/reference/A%20List%20Apart-Articles-Responsive%20Web%20Design.pdf
10. JQUERY “JQuery Mobile: jQuery Mobile: Touch-Optimized Web Framework for Smartphones & Tablets” (2013). http://jquerymobile.com/
11. IUI “iUI: User Interface Framework for Mobile Web Devices” (2013). https://code.google.com/p/iui/
12. Seven, D. “What is a hybrid mobile App. Icenium”. (2012). Blog publication at http://bit. ly/OMVQVN accessed, 23(9)
13. INTEL. “Bridging the Gap: from a Web App to a Mobile Device App”. HTML 5 DevCon (2013). http://html5devconf.com/archives/april2013/slides/2013HTML5DevCon_Bridging-the-Gap.pdf
14. CHARLAND Andre, LEROUX Brian. “Web apps are cheaper to develop and deploy than native apps, but can they match the native user experience? - Mobile application Development: Web vs. native”. Vol. 54, No. 5. ACM (2011) http://www.cemfarma.com/images/uploads/urun/pdf/bd1b62bc456d0240f58a8720fa66df1fda24046aS249V3.pdf
15. KHRONOS Group. “WebGL Specification” Version 1.0.2 (2013) https://www.khronos.org/registry/webgl/specs/1.0/
16. CHRIST Adam M. “Bridging the Mobile App Gap”. Sigma noblis, Volume 11, Number 1, pp.27-32 (2011). http://www.noblis.org/media/2dd575c1-2de9-4d92-9bdb-f72ad9fb9a19/docs/SigmaDigEco2011_pdf
17. FLING Brian. “Mobile Design and Development: Practical concepts and techniques for Creating Mobile Sites and Web Apps”. O`Reilly. ISBN 978-0-596-15544-5. Chapter 9: “Mobile Web Apps Versus Native Applications”. pp. 143-150 (2011).