Review of PhoneGap APIs Accessing the Native Mobile Platform APIs

Niyigena Jean Pierre and Mukiza Octavien

Abstract—PhoneGap with its JavaScript based APIs has proved that it is very advantageous for developers and this owing to the fact that developers can build mobile native-like applications (known as hybrid applications) with JavaScript, HTML5 and CSS3, web world standard technologies that most web developers typically know. In this paper, we discuss most of PhoneGap APIs and try to show how easy it is from the JavaScript world to access and control the native world of mobile platform with PhoneGap APIs through PhoneGap bridge. The ease of making use of only a couple lines of JavaScript code to access and manage mobile device native APIs with PhoneGap APIs makes obvious that PhoneGap is among the unexcelled mobile applications development frameworks.

Index Terms—Cross-platform, framework, JavaScript, mobile, PhoneGap APIs.

I. INTRODUCTION

Nowadays, smart phones are getting much more popularity notably Android, iPhone, consequently the developers are turning their attention on mobile based application researches. However, Android and iPhone development are based on different programming languages, the former is based on Java while the latter is based on Objective C [1]. To address the latter mentioned bottleneck the cross-platform mobile application development framework PhoneGap, among many others, was invented. The invention of PhoneGap framework made developers not acquire various computer languages to build mobile cross-platform applications. Table I shows the usual requirements for development environments for various mobile platforms.

<table>
<thead>
<tr>
<th>Mobile OS</th>
<th>Operating System</th>
<th>Software/IDEs</th>
<th>Programming Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS</td>
<td>Mac only</td>
<td>Xcode</td>
<td>Objective C</td>
</tr>
<tr>
<td>Android</td>
<td>Windows/ Mac/ Linux</td>
<td>Eclipse/ Java/ ADT</td>
<td>Java</td>
</tr>
<tr>
<td>BlackBerry</td>
<td>Windows mainly</td>
<td>Eclipse/ Java</td>
<td>Java</td>
</tr>
<tr>
<td>Symbian</td>
<td>Windows/ Mac/ Linux</td>
<td>Carbide.c++</td>
<td>C++</td>
</tr>
<tr>
<td>WebOS</td>
<td>Windows/ Mac/ Linux</td>
<td>Eclipse/ WebOS plugin</td>
<td>HTML/ JavaScript/ C++</td>
</tr>
<tr>
<td>Windows 7 Phone</td>
<td>Windows mainly</td>
<td>Visual Studio 2010</td>
<td>C#, .NET, Silverlight or WPF</td>
</tr>
</tbody>
</table>

It is worthwhile noting that applications that are developed using PhoneGap are hybrid applications [4]. These applications are not purely HTML/JavaScript based, nor are they native. Parts of the application, mainly the UI, the application logic, and communication with a server, is based on HTML/JavaScript. The other part of the application that communicates and controls the device (phone or tablet) is based on the native language for that platform.

II. THE WORKING THEORY OF PHONEGAP

The working theory of PhoneGap is not complex. The application’s user interface consists of essentially a unique screen that contains a unique web view that consumes all of the available space on the device’s screen [2].

As illustrated in Fig. 1, when the application launches, it loads the web application’s startup page (typically index.html but easily changed by the developer to something else) into the web view and then passes control to the web view to allow the user to interact with the web application.

As the user interacts with the application’s content (the web application), links or JavaScript code within the application would load. The discussion so far in this paper is just a high-level theory and the exact way a PhoneGap application is implemented can vary significantly depending on the specific use case.

Mukiza Octavien is with the Department of Software Engineering, School of Computer Science and Technology, Beijing Institute of Technology, Beijing, 100081 China (e-mail: mukoctavius@gmail.com).

Niyigena Jean Pierre is with the Department of Computer Networks and Technology, School of Computer Science and Technology, Beijing Institute of Technology, Beijing, 100081 China (e-mail: niyigelnx@gmail.com).

Manuscript received June 27, 2014; revised November 4, 2014.
can load other content from within the resource files packaged with this application or can reach out to the network and pull content down from a web or application server [1].

Fig. 1. Phonegap application architecture.

III. THE DESCRIPTION AND USES OF PHONEGAP APIs

In this section we describe most of the PhoneGap APIs and try to explain how to use the functionality provided by each API.

A. Globalization

PhoneGap globalization API both obtains information and make operations specific to the user's locale and time zone. In addition to having different objects like GlobalizationError, this API provides many various methods such as:

- navigator.globalization.dateToString,
- navigator.globalization.getPreferredLanguage,
- navigator.globalization.getLocaleName,
- navigator.globalization.stringToDate,
- navigator.globalization.getDatePattern,
- navigator.globalization.getDateNames,
- navigator.globalization.isDayLightSavingsTime,
- navigator.globalization.numberToString,
- navigator.globalization.getNumberPattern,
- navigator.globalization.getCurrencyPattern.

As a small example, by using navigator.globalization.dateToString method, if the browser has been set to the en\_US locale, this displays a popup dialog with text similar to date: 29/04/2014 4:21PM using the default options:

```javascript
function () { alert('Error getting dateString\n'); },
{ formatLength: 'short', selector: 'date and time' }
);
```

B. Vibration

This API provides a way to vibrate the device. notification.vibrate vibrates the device for a specified amount of time: To vibrate the device a JavaScript code can be written like this: navigator.notification.vibrate(time), where time is a number representing the amount of time for the device to vibrate expressed in milliseconds.

C. Device Motion (Accelerometer)

Accelerometer API is built on top of existing native motion detection API. It does not only access three dimensions points by accelerometer.getCurrentAcceleration method but also keeps track of interval distances relatively to the current position by accelerometer.watchAcceleration. The interval tracking could be interrupted by Accelerometer.clearWatch.

D. Camera

This PhoneGap Camera API abstracts away the details of native platform camera feature for handling images and host file system navigation by Camera.getPicture and camera.cleanup simple API interface methods. Images data are encoded as base64 string for further processing.

E. Capture

Capture media is at core of mobile devices. The PhoneGap platform is enriched with MIME type files that give it the abstraction layer to access device recording and file manipulation capabilities such as:

- capture.captureAudio
- capture.captureImage
- capture.captureVideo

With global scope capture object architecture and configurable parameters such as type, session length, etc… This makes API extensible and flexible to different media formats.

F. Device Orientation (Compass)

Compass API identify the frame of references in which device is stationary and retrieve the measures in degrees on interval [0 to 360] to which it is pointed.

The main methods to support the functionality are: compass.getCurrentHeading and compass.watchHeading; the latter tracks changes in direction.

G. Network Information (Connection)

Network connection abstraction layer is accessible through navigator connection Object, which is capable to determine the device’s active network status and type of connection.

H. Contacts

Synchronous methods such as contacts.create and contacts.find provide access to manipulate contact database if permission is granted by host device to the PhoneGap application.

I. Device

This framework interfaces native platform to retrieve device characteristic information by key value style device Object. This global object exposes device model, operating
system, Universally Unique Identifier.

J. Events

PhoneGap platform provides a complex event based synchronization through native and JavaScript. A native code loading preexists JavaScript event. Afterwards native code fires deviceready to make Dom loads and available to JavaScript. Event control within application is achieved by attaching an event listener with document.addEventListner method. Most common events are: deviceready, pause, resume, online, offline, backbutton, menubutton and stopwatch.

K. File-Transfer

In this API, the Classical file system operations read, write, modify, metadata retrieval are available through File Object. The file’s content are read and written as UTF-8 encoding data by default but they could be read as text or base64 data. The operations above, in order to be consistent, are coordinated through events scheme.

L. Storage

With Storage API, the implementation of SQL client API by PhoneGap conforms to www.w3.org specifications and built-in that interfaces host device API are all standardized to be reached through consistent and uniform Database objects. The key value storage of local storage object is also accessible.

M. Geolocation

In this API, one has to set up the geolocation system before it can be used. Once the setup is done, it is very convenient to use the provided methods to manipulate geolocation information. You can for instance get the current position of the device, monitor changes in the position of the device and when you are not interested in the devices geolocation information anymore you can stop querying for its state information. Geolocation information is stored inside read only objects such as Position, PositionError and Coordinates.

N. Media

The Media object provides the ability to record and play back audio files on a device. As of this paper writing, media implementation does not adhere to a W3C specification for media capture, and is provided for convenience only. A future implementation will adhere to the latest W3C specification and may deprecate the current APIs.

Set up must get carried out before one can use the Media objects. There is a mechanism to inform the media object of the media file to work on. This is usually specified within the parameters passed when one constructs his object. The media object provides all the functionalities we have grown to expect from modern media handling applications. These may include playing media, advancing or stepping backward as your media file is playing, adjusting the volume, pausing and resuming, recording and the like. Also, methods are provided to check the status on our media object. For example this makes it very convenient to inform the developer when something goes wrong with his media application and he can use this information to take appropriate action.

O. Splashscreen

The splashscreen is the screen that shows up on application startup so the user has something to look at when the system is busy doing heavy initializations. The splashscreen API takes care of that feature in PhoneGap [5].

<table>
<thead>
<tr>
<th>Feature</th>
<th>iPhone, iPad 3G and 4G</th>
<th>iPhone 4S and 5</th>
<th>BlackBerry OS 6.0+</th>
<th>BlackBerry 10</th>
<th>WebOS</th>
<th>Windows Phone 7+</th>
<th>Symbian</th>
<th>Bada</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerometer</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Camera</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Compass</td>
<td>X</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>Contacts</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>File</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Geolocation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>Media</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Network</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Notification (Alert)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Notification (Sound)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Notification (Vibration)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Storage</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

✓ - supported feature
X - unsupported feature due to hardware or software restrictions

With this API, the developer has the capability to show and hide the splashscreen. It is worthwhile noting that some splash screen implementations provide the possibility for the user to click (tap) on the screen to make it disappear. In this splashscreen API, this can be easily implemented using a simple click event on the splashscreen image.

P. Inappbrowser

PhoneGap also shipped-in inappbrowser API provides a web browser view that displays when calling window.open(), or when opening a link formed as `<a target="_blank">` [5]. The API provides all the basic features of a browser, and the most basic one is obviously loading a url within the view. The user can open and close urls. Also dynamic behavior can be added to the view through JavaScript event handling capabilities.

Q. Battery-Status

PhoneGap battery-status API provides an implementation of an old version of the Battery Status Events API [5]. This API gives the developer the power to monitor the current state of the battery and take appropriate action.
He(She) can for example know that the battery is about to die and notify the user of this.

IV. SUPPORTED PLATFORMS

Now that we have discussed all those PhoneGap shipped in APIs, one can ask himself if they are all supported for different mobile operating systems. PhoneGap currently supports development for Apple iOS, BlackBerry, Google Android, LG webOS, Microsoft Windows Phone (7 and 8), Nokia Symbian OS, Tizen (SDK 2.x), Bada, Firefox OS, and Ubuntu Touch. Table II below is a list of supported features for each operating system.

V. CONCLUSION

In this paper, we have described most of PhoneGap APIs and we have explained how JavaScript is used to access mobile native device APIs for almost each them. PhoneGap APIs are convenient to use on grounds of using just a few lines of JavaScript, a technology that most web developers use, to access mobile device APIs for so many different platforms. PhoneGap has other competitive mobile frameworks out there such as Titanium and Corona but PhoneGap still most used today by most developers. Note that even if PhoneGap does support most native features out of the box, there are cases where a native feature may be needed for an application but that feature is not supported by available PhoneGap shipped in APIs. If that is the case, PhoneGap does allow Plugins. Plugins are written for a specific platform in that platform’s native language. For example a plugin for iOS would be written in Objective C and a plugin for Android would be written in Java [6]. Also needed will be a JavaScript component which will expose the plugin to the application. Each platform needs a separate JavaScript component as well.

REFERENCES


Niyigena Jean Pierre was born in Rwanda on 7th September 1984. He held his bachelor’s degree in computer science and technology in 2012 from Beijing institute of technology, China. Currently, he is a master’s student in the Department of Computer Networks and Security, Faculty of Computer Science and Technology at Beijing Institute of Technology.

Mukiza Octavien was born on June 4, 1985. He got his bachelor’s degree in computer science and technology in 2012 from Beijing Institute of Technology, in Beijing China. Currently, he is a master’s student in the Department of Software Engineering in Computer Sciences and Technology at Beijing Institute of Technology.