Using Phonegap as a Mobile App Platform

Adrian Kosmaczewski

2013-11-04

This report will provide an overview of the challenges and opportunities brought by PhoneGap when building cross-platform mobile applications for touchscreen smartphones and tablets.

(Download ZIP file[1] with PDF and EPUB version of this report.)

It will be based in the experience of the Trifork team while building real-life applications using this tool.

Introduction

PhoneGap[2] is a mobile development framework produced by Nitobi, purchased by Adobe Systems. It enables software programmers to build applications for mobile devices using JavaScript, HTML5, and CSS3, instead of device-specific languages such as Objective-C or Java. The resulting applications are hybrid, meaning that they are neither truly native (because all layout rendering is done via web views instead of the platform's native UI framework) nor purely web-based (because they are not just web apps, but are packaged as apps for distribution and have access to native device APIs).

The software underlying PhoneGap is Apache Cordova. The software was previously called just "PhoneGap", then "Apache Callback". Apache Cordova is open source software.

History

First developed at an iPhoneDevCamp event in San Francisco, PhoneGap went on to win the People's Choice Award at O'Reilly Media's 2009 Web 2.0 Conference and the framework has been used to develop many apps. Apple Inc. has confirmed that the framework has its approval, even with the new 4.0 developer license agreement changes. The PhoneGap framework is used by several mobile application platforms such as ViziApps, Worklight, Convertigo, and appMobi as the backbone of their mobile client development engine. Adobe officially announced the acquisition of Nitobi Software (the original developer) on October 4, 2011. Coincident with that, the PhoneGap code was contributed to the Apache Software Foundation to start a new project called Apache Cordova. The project original name, Apache Callback, was viewed as too generic. Then it also

[1] report.zip

[2] At the time of this writing, the current stable version of PhoneGap is version 3.1.

appears in Adobe Systems as Adobe PhoneGap and also as Adobe Phonegap Build.

Early versions of PhoneGap required a person making iOS apps to have an Apple computer, and a person making Windows Mobile apps to have a computer running Windows. After September 2012, Adobe's PhoneGap Build service allows programmers to upload HTML, CSS and JavaScript source code to a "cloud compiler" that generates apps for every supported platform.

As presented by Adobe, PhoneGap provides two complementary services:

- A *wrapper*; PhoneGap packages HTML, CSS and JavaScript files to be deployed and distributed through mobile app stores.
- A *bridge*; PhoneGap also provides mechanisms to augment HTML5 web applications, allowing them to access and consume information and services otherwise only available to native applications, such as the local address book, the notification system, sounds, and other utilities.

Supported Platforms

At the time of this writing, PhoneGap supports the following mobile platforms:

- Apple iOS
- Android
- BlackBerry
- webOS
- Microsoft Windows Phone
- Symbian
- Bada
- Tizen

Supported Features

As a bridge, PhoneGap allows developers to access the following features:

- Accelerometer
- Address Book
- Camera
- Compass
- File system
- Geolocation
- Media
- Network
- Notifications
- Storage

Not all features are supported in all the mobile platforms in which PhoneGap runs; the chart reproduced in the table below shows the features available in each platform. As a rule of thumb, *iOS, Android, Windows Phone, Tizen and BlackBerry 10 support the complete feature set offered by PhoneGap.*

Supported PhoneGap Features:

Feature	iOS	Android	WP	BB	Bada	Symbian	webOS	Tizen
Accelerometer	x	x	x	x	x	x	x	x
Camera	x	x	x	x	x	x	x	x
Compass	x	x	x	x	x		x	x
Contacts	x	x	x	x	x	x		x
File	x	x	x	x				x
Geolocation	x	x	x	x	x	x	x	x
Media	x	x	x	x				x
Network	x	x	x	x	x	x	x	x
Notifications	x	x	x	x	x	x	x	x
Storage	x	x	x	x		x	x	x

Design Rationale

The core of PhoneGap applications use HTML5 and CSS3 for their rendering, and JavaScript for their logic. Although HTML5 now provides access to underlying hardware such as the accelerometer, camera and GPS, browser support for HTML5-based device access is not consistent across mobile browsers, particularly older versions of Android. To overcome these limitations, the PhoneGap framework embeds HTML5 code inside a native WebView on the device, using a foreign function interface to access the native resources of the device.

PhoneGap is also able to be extended with native plug-ins that allow for developers to add functionality that can be called from JavaScript, allowing for direct communication between the native layer, and the HTML5 page. PhoneGap includes basic plugins that allow access to the device's accelerometer, camera, microphone, compass, file system, and more.

However, the use of web-based technologies leads many PhoneGap applications to run slower than native applications with similar functionality. Adobe Systems warns that applications built using PhoneGap may be rejected by Apple for being too slow or not feeling "native" enough (having appearance and functionality consistent with what users have come to expect on the platform).

Basic Usage

To create a native application using PhoneGap usually involves the following steps:

1. Create the web application using HTML5, JavaScript and CSS.
2. Install the PhoneGap framework in the development machine.
3. Create a PhoneGap application using the command line tools.
4. Add the required HTML, CSS and JavaScript files.
5. Build the application using the command line tools.
6. Deploy to a device for testing, or to a mobile app store for distribution.

Experience

As advanced as it may be, any tool has limitations, which the developer must be aware of, and strenghts to exploit and take advantage of. This chapter will provide a short summary of highlights and drawbacks encountered while using PhoneGap in real-life projects.

Installation

The first contact with PhoneGap is through the installation of the developer tools. As a toolkit, PhoneGap is entirely dependent on the pre-existence of the native developer tools for each target platform. For example, to create iOS applications, the developer machine should be bundled with Xcode, and so on[3].

PhoneGap is bundled through the `npm` package manager system, originally built with the open source project `Node.js`. Through `npm` developers can install PhoneGap in the developer machines. `npm` also takes care of providing updates to the core PhoneGap libraries as they are released.

Command Line Tools In an effort to streamline and simplify the creation and management of PhoneGap systems, Adobe provides a series of command-line tools that can be used to create and build PhoneGap applications.

For example, to create a new application, a developer would write the following command:

```
$ phonegap create --name "AppName" --id com.trifork.AppName foldername
```

The command above would create an application named `AppName` in a subfolder of the current working directory called `foldername`.

Once the application is created, developers can perform several different operations on it; typically, a certain number of plugins would be required, which can be added with the following commands:

```
$ phonegap local plugin add org.apache.cordova.dialogs
$ phonegap local plugin add org.apache.cordova.console
```

These commands not only include the required code in the project (both native and JavaScript), they also modify the permissions and other settings in the platform-specific projects for each target[4].

Similarly, the command line utilities can be used to package and build the native application for a specific platform[5]:

```
$ phonegap build ios
[phonegap] detecting iOS SDK environment...
[phonegap] using the local environment
[phonegap] adding the iOS platform...
[phonegap] compiling iOS...
```

[3]This report will not deal with PhoneGap Build, the cloud-based build system provided by Adobe, as for confidentiality reasons some customers might not be comfortable knowing that their applications are being built elsewhere.

[4]It is strongly recommended to install at least the console plugin, which provides a direct connection from the web `console.log()` statements to the native logging mechanism, like the Android logcat or the iOS console.

[5]Unfortunately, at the time of this writing, the PhoneGap Android developer build system does not work when launched from directories that contain a space. Developers should make sure that they create their applications using full paths without spaces.

```
[phonegap] successfully compiled iOS app

$ phonegap build android
[phonegap] detecting Android SDK environment...
[phonegap] using the local environment
[phonegap] adding the Android platform...
[phonegap] compiling Android...
[phonegap] successfully compiled Android app
```

Dependencies

As stated above, building a native PhoneGap application requires the following tools installed in the local machine (this text assumes that the reader is using a Mac computer running the latest version of OS X and wishes to create iOS and Android applications using PhoneGap):

- Xcode.
- Xcode command line tools (installed separately from Xcode.)
- Android SDK (can be installed through Homebrew.)
- Apache Ant

Challenges

This section will highlight major issues and challenges faced by developers when creating PhoneGap applications, organized in three major groups:

- JavaScript.
- Development time.
- Documentation.
- Debugging.
- Integration.

JavaScript PhoneGap applications are built using JavaScript, a language recognized as particularly difficult to master, particularly given the fact that it bears a misleading name. JavaScript is a functional, class-less language with objects, with a single-threaded, run-loop execution model. This description places immediate constraints in the layout and organization of the source code, and many developers, even after having been exposed to the language for years, are not familiar with them.

It is recommended that developers be exposed to a certain level of training (usually one or two days is enough) in JavaScript before delving into PhoneGap applications; this will increase their productivity and will help them debug the resulting applications faster.

This becomes extremely important when dealing with supplementary UI frameworks such as jQuery Mobile or Sencha Touch, which can be run inside PhoneGap applications and provide a higher-level abstraction for creating complex web applications in a shorter amount of time.

Development Time The biggest advantage of having a single code base for all the platforms covered by a mobile solution is, without any doubt, the reduced time required to write and maintain the application. In general, not only is the development time reduced, but also the time spent to fix a bug is also amortized accross platforms.

This principle implies that the true economy of scale is only achieved when several platforms are required for the same source code base. The development time used to write a mobile app in JavaScript is around 60% to 70% the time required to write the same solution in Objective-C, C# or Java. Once this is done, adding a new platform to the PhoneGap solution is a matter of hours.

However, PhoneGap (or any other cross-platform technology, for that matter) does not reduce the time required for marketing, submission to the App Store, customer relationship management, and other related tasks; for each platform, a substantial overhead is expected in non-development tasks, and this overhead is comparable to that of native solutions.

All in all, PhoneGap represents an interesting opportunity as soon as there are more than one platforms required for a mobile solution; otherwise it is impossible to maximize the advantage provided by this approach.

Documentation The documentation provided by Adobe for PhoneGap is probably the weakest point of the whole platform, particularly regarding the exact syntax of the command line options. These have changed through the past releases (particularly between versions 2 and 3.0, and again in 3.1) which makes it quite difficult to move forward in stable ground. The author hereby wishes that the whole toolkit will remain stable for the time being.

In many cases, the syntax of the `phonegap` commands is simply wrong, and it even changes from page to page in the same documentation set, with errors sprinkled all over. Needless to say, this greatly diminishes the appeal of the whole platform, and makes the development process needlessly complex.

The other part of the documentation, which basically describes the various APIs available to connect to the different device subsystems, however, has been stable for at least 2 major releases, and for the past 3 years; this section of the documentation is solid and clear, and includes lots of useful examples.

The experience shows that these APIs work as intended, even if sometimes it is not very clear that plugins are required to make them work properly. Without those plugins, the application fails silently without any feedback to the developer.

Debugging Debugging PhoneGap applications is another delicate point. While most of the development of a web application happens on a web browser, where the tools are these days acceptably good, solving issues in PhoneGap applications remains a complex task.

Debugging a mobile web application involves the same tasks as in any other platform:

- Setting breakpoints and exploring the behavior of the application at runtime, including variables and call stacks.

- Inspecting the layout of the application, in this case composed by the HTML and rendered using CSS statements.
- Watching the log statements produced by the application immediately.
- Being able to observe the state of the file system or other storage media.
- Performing all the tasks enumerated above in a device, in real time.

Most web browsers these days include excellent developer tools, like Chrome, Safari, Firefox and even the latest releases of Internet Explorer. The problem remains when using native APIs, like the camera or the accelerometer, where browsers do not emulate those capabilities, at least not at the moment of this writing.

In particular, since the release of OS X Mavericks and iOS 7, Safari 7 for Mavericks is apparently unable to properly debug web applications wrapped in PhoneGap containers and running on devices connected via USB. Several other developers have raised similar concerns, and apparently there is no solution for this problem at the moment.

Another solution for in-device debugging involves using Adobe Edge, a system proposed by Adobe which specifically targets this problem; using Edge, developers can inspect the runtime of their applications directly on the device, getting access to log statements, but without the ability of setting breakpoints in the code, which is a major drawback at the moment.

Integration The final complex point of PhoneGap has to do with the integration with build systems or continuous integration platforms.

In this case, the fact that PhoneGap is bundled as a set of command-line tools helps developers greatly, since the build and deployment of PhoneGap applications can be directly done from build scripts or similar tools (Ant, GNU Make, shell scripts, etc.)

The most important thing to take into account when setting up the build environment is the dependencies; for example, PhoneGap requires Ant and the Android SDK to be able to build Android applications, and so on.

Conclusion

PhoneGap provides an interesting approach to the creation of mobile applications; however there are inherent problems due to the youth of the system and the changing priorities of the corporation behind it.

In the opinion of the author of this lines, it is strongly recommended to avoid PhoneGap for performance-sensitive or complex tasks, like games or productivity applications; the strength of PhoneGap is in the fast delivery of cross-platform business solutions, particularly in the case of form-based applications, used for reporting and data mining.

PhoneGap also requires developers to be comfortable with the JavaScript language and its associated execution model (functional, run loop based, single-threaded, etc.) This skill is usually not found, even in developers having been exposed to the language during years.

Finally, PhoneGap provides substantial benefits when an application targets 3 or more platforms; true economies of scale can be achieved from that point forward, both in development and maintenance times.