
iPad
AnswerHub
The Enterprise Q&A Platform
Collect, Manage and Share All the Answers Your Team Needs
> Q&A Software for Enterprise Knowledge Sharing
> Mobile-friendly Responsive Design
> Easy to Customize with Themes and Plugins
> Integrates with Your SSO and IT Infrastructure
> Choose Cloud-based or On-premise Hosting
Watch Video
Free Demo
Pricing

DZone Refcardz

CONTENTS INCLUDE:

Getting Started with PhoneGap

By: Raymond Camden

ABOUT ADOBE PHONEGAP

Adobe PhoneGap is an open source framework that lets you build hybrid mobile applications using HTML, JavaScript, and CSS. In 2012, the PhoneGap source became a top-level project under the Apache Software Foundation with the name "Cordova." (You may hear people refer to PhoneGap and Cordova interchangeably.)

PhoneGap provides support for using web standards to creative native applications that run on seven different mobile platforms:

- Android
- iOS
- Windows Phone
- Blackberry
- WebOS
- Symbian
- Bada

This Refcard is intended for experienced developers who have a basic conceptual grasp of mobile app development. The card does not presuppose any real-world mobile dev experience, but experienced mobile developers will probably find the card especially useful.

Along with creating installable mobile web applications, PhoneGap provides a JavaScript bridge to your devices' native hardware. This provides support within your JavaScript for the following features:

Feature	Description
Accelerometer	Allows for tracking the devices' movement, noticing 'shakes' and orientation.
Camera	Allows the user to take a picture for your application or access pictures on the device.
Capture	Allows for video and audio capture.
Compass	Returns compass directions for the device.
Connection	Lets your application determine if there is a connection as well as how strong it is.
Device	Returns basic metadata about the device.
Events	Support for various device related events, including battery level changes.
File	File system access for the device. Includes basic read/write functionality for both files and directories.
Geolocation	Allows you to locate the user.
Globalization	Support for formatting numbers and dates in the user's native language.
InAppBrowser	A way to provide web access to URLs without the user leaving your application.
Media	Audio recording and playback support.
Notification	Various ways to get the user's attention, including sound and vibration-based alerts.
Splashscreen	Lets your application customize the startup experience.
Storage	Provides a single-user database for your application.

GETTING STARTED

Each platform (Android, iOS, etc.) has its own SDK and installation procedures. Read more about them here: http://docs.phonegap.com/en/3.0.0/guide_platforms_index.md.html#Platform%20Guides.

Depending on which platforms you care to build for, you should follow the particular guide's instructions. Some platforms, like iOS, will require a paid program membership.

To make it easier to work with these SDKs, PhoneGap has a command line interface that can be installed via npm (Node Package Manager). NPM is installed with Node.js and may be downloaded here: http://nodejs.org/download. **Note:** you absolutely do **not** need to be a Node developer to use PhoneGap. Simply install Node.js and once you can run npm, use the following command to install the PhoneGap CLI (Command Line Interface):

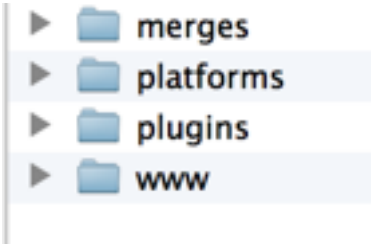
```
sudo npm install -g phonegap
```

The PhoneGap command line can be used to create new projects, create binary builds, and run your code on devices and emulators. Even better, it fully supports your local SDKs or the remote PhoneGap Build service described later in this guide. Most developers will probably prefer to work locally, but downloading and setting up multiple SDKs is not required.

To create a new application, simply run this at the command line:

```
phonegap create somedir org.sample.test test
```

In the example above, a new project will be created in a subdirectory called **somedir**, using the ID of **org.sample.test**, and using an application named **test**. Both the ID and application name values are optional. The directory will contain the following folders:



For now, just concern yourself with the **platforms** and **www** folder.

The **platforms** folder contains a platform-specific install of whatever device types you want to support. So for example, if you wanted to build an app for iOS and Android, there would be two folders underneath it. The folder is empty when you first create a new project.

The **www** folder contains the actual HTML, JavaScript, CSS, and related assets for your application. This is where you will edit your code.

The PhoneGap command line will handle copying your files into the appropriate platform directories.

By default, the www folder already contains some basic assets, so you can immediately test out your project by first building for a specific platform. (**Note:** we are assuming you have either downloaded and setup your SDKs already, or are going to use PhoneGap Build.) To build for iOS:

```
phonegap build ios
```

If everything worked ok you will see:

```
[phonegap] detecting iOS SDK environment...
[phonegap] using the local environment
[phonegap] adding the iOS platform...
[phonegap] compiling iOS...
[phonegap] successfully compiled iOS app
```

To run the application:

```
phonegap run ios
```

Here's the default application running in the iOS emulator:

USING PHONEGAP'S FEATURES

Now that you know how to create a project, build it, and deploy it to a simulator, how do you actually make use of PhoneGap's features?

Each PhoneGap project ships with a JavaScript file, phonegap.js, that provides hooks to native device features not normally available to web pages on the device. The file, phonegap.js, will not be present in the www folder you just created. Yet you can see it included by the default index.html file:

```
<script type="text/javascript" src="phonegap.js"></script>
```

How does this work? When you create a build, the command line tool will copy your HTML, JS, and other assets into the platform folder and automatically include the proper phonegap.js file for each unique platform. The next step is crucial. When your application starts up, it will not be ready to use any special devices features. Instead, it must wait for the application to initialize those hooks and listen for an event called deviceready. Here's a simple example of that in action:

```
document.addEventListener('deviceready', deviceready, false);

function deviceready() {
         alert('Ready!');
}
```

Once the deviceready event is fired, you can then use any of the PhoneGap features your application requires.

USING THE CAMERA API

As an example of using PhoneGap's APIs, let's build a simple application that makes use of the device camera. Before we use the Camera API, we must include support for the camera plugin. Luckily this is easy enough to do via the PhoneGap CLI:

```
phonegap local plugin add https://git-wip-us.apache.org/repos/
asf/cordova-plugin-camera.git
```

All of PhoneGap 3's major features are implemented as plugins. This lets you create a slimmer, simpler application that only supports what you need. Now that you've added the plugin, let's begin with some simple HTML.

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />
        <meta name="viewport" content="user-scalable=no, initial-
scale=1, maximum-scale=1, minimum-scale=1, width=device-width;"
/>
        <link rel="stylesheet" type="text/css" href="css/index.
css" />
        <title>Camera Example</title>
    </head>
    <body>

        <button id="getPicture">Take Picture</button>

        <img id="pictureResult">

        <script type="text/javascript" src="phonegap.js"></
script>
        <script type="text/javascript" src="js/index.js">
</script>
    </body>
```

In order to make the button a bit easier to use, we've added a simple CSS file to make it fatter:

```
button {
    width: 80%;
    height: 30px;
    font-size: 1.1em;
    margin-left: auto;
    margin-right: auto;
    display: block;
}
```

Here is how it looks in the iOS Simulator:

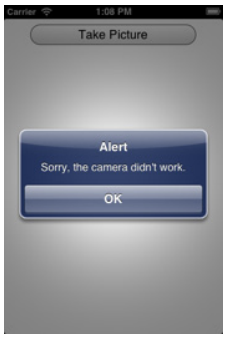
Now that we've got the layout built, let's look at the code.

```
document.addEventListener("deviceready", ready, false);

function ready() {
   var camBtn = document.querySelector("#getPicture");
   camBtn.addEventListener("touchstart", beginCamera, false);
}

function beginCamera(e) {
   e.preventDefault();

   navigator.camera.getPicture(cameraSuccess, cameraError,
     {
        quality:50,
        destinationType: Camera.DestinationType.FILE_URI,
        targetWidth:350,
        targetHeight:350
     });

}

function cameraError(e) {
  navigator.notification.alert("Sorry, camera didn't work.", null);
}

function cameraSuccess(uri) {
   document.querySelector("#pictureResult").src = uri;
}
```

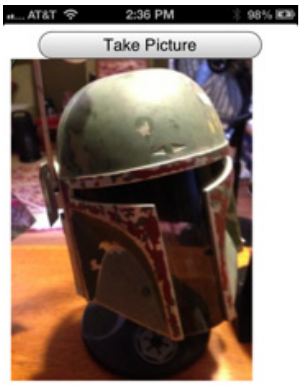
The file begins with a simple event handler for the deviceready event. PhoneGap will fire this automatically when it is safe to use device APIs.

In the beginCamera function you can see the call to navigator.camera.getPicture. The first two arguments are a success and error handler respectively. The third argument is a set of options that let you customize how the camera will be used. In our example, we've specified a quality of 50, a destination type of a file uri, and a target width and height of 350 pixels.

The cameraError function makes use of another part of the PhoneGap API, the notification feature. If the user's device doesn't have a camera (or something else goes wrong), this notification will be displayed:



The cameraSuccess handler is fired after a picture is selected. Since we elected to have a FILE_URI sent, we can simply use that in our DOM. We select the image and assign the URI to the src attribute.



Camera API Overview

Methdod	Description
navigator.camera.getPicture(success handler, error handler, options)	Main method for getting picture data. It also supports getting a picture from the user's existing collection of photos. Other options include desired quality and size as well as encoding type.
navigator.camera.cleanup	A utility to remove photos that may exist in the application's temporary directory.

USING THE CONNECTION API

Another useful feature of the PhoneGap API is the Connection API. PhoneGap apps run locally on a device. Technically, the device does not need to be online to run, so it's important to know connection status. In this code snippet, we inspect the connection device connection status as well as listen for changes:

```
document.addEventListener("deviceready", init, false);

function init() {

          document.addEventListener("online", toggleCon, false);
          document.addEventListener("offline", toggleCon, false);

          if(navigator.network.connection.type == Connection.
NONE) {
                    navigator.notification.alert("Sorry, you are
offline.", function() {}, "Offline!");
          } else {
                    someFunc();
          }

}
```

By using a little bit of code your application can then intelligently respond to changes in the device's connection to the Internet. You can even check the strength of the connection. This would be useful for deciding if it makes sense to stream large files.

Connection Object Values

The following is a list of each constant available in the navigator.network.connection object:

- Connection.UNKNOWN
- Connection.ETHERNET
- Connection.WIFI
- Connection.CELL_2G
- Connection.CELL_3G
- Connection.CELL_4G
- Connection.CELL
- Connection.NONE

USING THE DEVICE API

Another useful feature is the Device API. While not exactly an API, the device object returns values for the name, platform, version, model, as well as the currently running version of PhoneGap. If your code needs to do something slightly different for iOS versus Android, this is what you would use.

```
document.addEventListener("deviceready", init, false);

function init() {
          var deviceDesc = "";
                    deviceDesc += "Device Name: " + device.name +
"<br/>";
                    deviceDesc += "Device Platform: " + device.
platform + "<br/>";
                    deviceDesc += "Device Version: " + device.
version + "<br/>";

                    document.querySelector("#device").innerHTML =
deviceDesc
```

Device Object Values

The following is a list of each constant available in the device object:

- device.name
- device.cordova
- device.platform
- device.uuid
- device.version
- device.model

OTHER APIS

Now let's take a quick tour through the rest of PhoneGap's APIs. This is not a complete list but covers the most important (and useful) features.

Accelerometer

Method	Description
navigator.accelerometer.getCurrentAcceleration(success handler, error handler)	Returns the 'acceleration', or current movement, of the device in X, Y, Z values.
navigator.accelerometer.watchAcceleration(success handler, error handler, options)	Will repeatedly check the device's acceleration. This can be stopped with navigator.accelerometer.clearWatch.

In this example, the watchAcceleration API is used to run a method every three seconds. The data passed to the handler includes movement values over three different axis as well as a timestamp of the reading.

```
function onSuccess(acceleration) {
    console.log('Acceleration X: ' + acceleration.x + ' ' +
          'Acceleration Y: ' + acceleration.y + ' ' +
          'Acceleration Z: ' + acceleration.z + ' ' +
          'Timestamp: '      + acceleration.timestamp);
};

function onError() {
    navigator.notification.alert('onError!');
};

var options = { frequency: 3000 };  // Update every 3 seconds

var watchID = navigator.accelerometer.
watchAcceleration(onSuccess, onError, options);
```

Capture

Method	Description
navigator.capture.captureAudio(success handler, error handler, options)	Allows the user to record one or more audio clips. Could be used to let the user verbally annotate data.
navigator.capture.captureImage(success handler, error handler, options)	For the most part very similar to the Camera API, but allows for multiple captures.
navigator.capture.captureVideo(success handler, error handler, options)	Allows the user to record one or more video clips. You can use the options to specify a max duration.

In this example, the captureAudio method will prompt the user to record one audio file. The success handler is passed an array of file paths. Once you have that file path you can move the file or even upload it using the FileTransfer API.

```
function captureSuccess(mediaFiles) {
    var i, path, len;
    for (i = 0, len = mediaFiles.length; i < len; i += 1) {
        path = mediaFiles[i].fullPath;
        // do something interesting with the file
    }
};

function captureError(error) {
    navigator.notification.alert('Error: ' + error.message, null,
'Capture Error');
};

navigator.device.capture.captureAudio(captureSuccess,
captureError);
```

Compass

Method	Description
navigator.compass.getCurrentHeading (success handler, error handler)	Returns the compass heading of the device as a value from 0 to 359.99.
navigator.compass.watchHeading(success handler, error handler, options)	Will repeatedly check the device's compass direction. This can be stopped with navigator.compass.clearWatch.

In this example, the compass will be monitored and direction data sent to a method. The data sent includes two types of heading values, an accuracy value, and a timestamp as well.

```
function gotHeading (heading) {
    console.log("Current heading is "+heading.magneticHeading);
};

function compassError(compassError) {
    navigator.notification.alert('error: ' + compassError.message,
null, 'Compass Error');
};

var watchID = navigator.compass.watchHeading(gotHeading,
compassError);
```

Contacts

Method	Description
navigator.contacts.create (properties)	Allows for the creation of a new contact.
navigator.contacts.find(fields, success handler, error handler, options)	Performs a search against the device contact database. You can search against multiple fields (like name or email).

In this example, the contact database is searched for people with adobe.com anywhere in their data. Note that the default for find operations is to return one result. Typically you will want multiple results. Be sure to include the multiple=true flag in your options object.

```
function contactSearch(contacts) {
    navigator.notification.alert('Found ' + contacts.length + '
contacts.',null,'Contact Results');
};

function contactError(error) {
    navigator.notification.alert('Error:: '+error.
message,null,'Error');
};

var options = new ContactFindOptions();
options.filter="adobe.com";
options.multiple=true;
var fields = ["displayName", "name"];
navigator.contacts.find(fields, contactSearch, contactError,
options);
```

Events

Type	Description
deviceready	The most important event. Nothing should be used until after this event is fired!
pause/resume	Fired when an application pauses or is resumed.
online/offline	Fired when the device gains or loses network connectivity.
backbutton	For devices that have a back button, this event can be used to detect its use.
batterycritical/ batterylow/ batterystatus	Various battery level events are available. Would be useful to remind the user to save data when the battery is low.
menubutton	For devices that have a menu button, this event can be used to detect its use.
searchbutton	For devices that have a search button, this event can be used to detect its use.
startcallbutton/ endcallbutton	A Blackberry-only set of events related to calls.
volumedownbutton/ volumeupbutton	A Blackberry-only set of events related to the device volume.

One of the best examples of PhoneGap's Event API is listening for changes to online and offline status. Making your application respond intelligently to these events are crucial for building applications that provide the best experience for the user. Even something as simple as recognizing that a user is offline and switching to an older cache on the device is much more useful then throwing an error.

Geolocation

Method	Description
navigator.geolocation. getCurrentPosition (success handler, error handler, options)	Returns the position of the device.
navigator.geolocation. watchPosition(success handler, error handler, options)	Will repeatedly check the device's location. This can be stopped with navigator.geolocation.clearWatch.

While you can constantly monitor the user's location, typically you will only need to get their location once. The result contains values outside of just longitude and latitude. Additional values include the user's altitude as well as an accuracy value for both their location and altitude. Here is an example of that.

```
function gotLocation(pos) {
  console.log("You are at " + pos.coords.longitude+" longitude
and " + pos.coords.latitude + " latitude.");
}

function locationError(error) {
    navigator.notification.alert('Location: ' + error.message,
null, 'Location Error');

}
navigator.geolocation.getCurrentPosition(gotLocation,
locationError);
```

Globalization

Method	Description
navigator.globalization. getPreferredLanguage (success handler, error handler)	Returns the device's default language setting. There is also a globalization.getLocaleName method.
navigator.globalization. dateToString(date, success handler, error handler, options)	Converts a date into a format recognizeable by the device's locale. The reverse method is stringToDate.
navigator.globalization. getDatePattern(success handler, error handler, options) / getNumberPattern / getCurrentPattern.	Returns a "pattern" representing the date format for the device's locale. Also supports numbers and currency via their own methods.
navigator.globalization. getDateNames(success handler, error handler, options)	Returns either an array of month names or week days based on the device's locale.
navigator.globalization. isDayLightSavingsTime(date, success handler, error handler)	Determines if daylight savings is in effect for a particular date.
navigator.globalization. getFirstDayOfWeek(success handler, error handler)	Returns a number between 1 and 7 (where 1 is Sunday) representing the device's locale's initial week day.
navigator.globalization. numberToString(date, success handler, error handler, options)	Converts a number into a format recognizeable by the device's locale. The reverse method is stringToNumber. The method supports decimals, percents, and currency values.

Probably the best example of where users can be confused is with dates. Given a date of 2/3/2013, is that February 3, 2013 or March 2, 2013? Here is an example of how the Globalization API can simplify this. You pass in your date and it worries about formatting it properly for the end user.

```
navigator.globalization.dateToString(
  new Date(2013, 2, 3),
  function (date) {console.log('Date: '+date.value);},
  function () {navigator.notification.alert('Error getting date
string',null,'Error');},
  {formatLength:'short', selector:'date'}
);
```

Notification

Method	Description
navigator.notification.alert(message, callback, title, button)	Creates a visual notification of a simple message.
navigator.notification. confirm(message, callback, title, buttons)	Creates a visual notification that asks for confirmation.
navigator.notification. prompt(message, callback, title, button)	Creates a visual notification that asks the user to enter some data.
navigator.notification.beep(times)	Creates an audio alert.
navigator.notification. vibrate(milliseconds)	Vibrates the device.

We've shown many examples of the notification API in this Refcard, but it bears repeating that while you can use browser-based notifications like alert() and confirm(), you should almost always use the native prompts via the notification API instead. Note that the vibrate method will obviously not work on devices that do not support tactile notifications (like the iPad).

TESTING PHONEGAP APPS

While you can use the command line to build and test PhoneGap apps, there are a few other options you may wish to consider.

Feature	Description
The Ripple Emulator	Ripple is a Chrome extension that lets you test PhoneGap apps in the browser. While not a 100% complete mirror of the PhoneGap functionality, it allows for testing of almost all of the PhoneGap feature set. This extension is free and may be downloaded here: http://bit.ly/10VCTsK
PhoneGap Build	PhoneGap Build (build.phonegap.com) is a free online service. By connection to your GitHub account, or uploading a simple zip file, you can make use of their own compilation services. This allows people to create builds for multiple platforms at once, all in the cloud!

PHONEGAP SUPPORT

PhoneGap offers commercial support subscriptions (http://phonegap.com/support) that lead developers from app design and development through testing and deployment. Subscriptions include access to live online training and knowledge base, weekly chat office hours & a private forum staffed by a professional technical support team.

For community support, PhoneGap Google Group (https://groups.google.com/forum/?fromgroups#!forum/phonegap) is quite active. Topics range from focused, Stack Overflow-style problem-solving to rich discussions on broader questions like 'how do you develop for X resolution'.

ADDITIONAL RESOURCES

Here is a short list of resources that can provide you with additional examples and inspiration for working with PhoneGap.

- The official PhoneGap Blog (http://phonegap.com/blog/) is incredibly useful as it also aggregates content from multiple different developers (including your author!).
- Looking for other PhoneGap developers? Check out the PhoneGap Developer Directory (http://people.phonegap.com).
- The PhoneGap Mobile Application Development Cookbook (http://www.amazon.com/PhoneGap-Mobile-Application-Development-Cookbook/dp/1849518580) by Matt Gifford is an excellent set of real world examples using the PhoneGap framework.
- Finally, if you are interested in attending events featuring PhoneGap topics, check the official PhoneGap events site: http://phonegap.com/event

ABOUT THE AUTHOR

Raymond Camden is a developer evangelist for Adobe. His work focuses on web standards, mobile development, Node.js, and ColdFusion. He's a published author and presents at conferences and user groups on a variety of topics. He is the happily married proud father of three kids and is somewhat of a Star Wars nut. Raymond can be reached via his blog at www.raymondcamden.com or via email at raymondcamden@gmail.com

RECOMMENDED BOOK

This is a cookbook with each section written as a recipe in an informal, friendly style. Each recipe contains the complete code needed to build your applications, and plenty of screenshots showing the completed projects running on devices are included to assist you.

Buy Here

DZone

DZone communities deliver over 6 million pages each month to more than 3.3 million software developers, architects and decision makers. DZone offers something for everyone, including news, tutorials, cheat sheets, blogs, feature articles, source code and more. "'DZone is a developer's dream',' says PC Magazine.

DZone, Inc.
150 Preston Executive Dr.
Suite 201
Cary, NC 27513

888.678.0399
919.678.0300

Refcardz Feedback Welcome
refcardz@dzone.com

Sponsorship Opportunities
sales@dzone.com

ISBN-13: 978-1-936502-80-6
ISBN-10: 1-936502-80-1
50795
9 781936 502806
$7.95

Version 1.0