

jQuery Mobile

PhoneGap

Lesson 1, Activity 2: PhoneGap

Many mobile devices expose native capabilities like GPS, accelerometer, and orientation via JavaScript. But there's a limit to how much we can access - it would be nice to make use of the phone's camera, for instance, or to tap into (with the user's permission) the contacts in our sites.

Of course, we could write native apps - but that would mean writing separate native code for each targeted device platform and, for many of us, tacking on new skill set(s) to learn Objective-C, Java, or other languages. One promising future development is the work of the World Wide Web Consortium's Device APIs Working Group. As their [Charter website](#) states, the mission of the group is to "create client-side APIs that enable the development of Web Applications that interact with device hardware, services and applications such as the camera, microphone, system sensors, native address books, calendars and native messaging applications." To date, these APIs are not widely available for production use. So what's a poor web developer to do?

(Just to make sure we're all on the same page: an API is an Application Programming Interface, "a specification intended to be used as an interface by software components to communicate with each other." An SDK is a Software Development Kit, "a set of software development tools that allows for the creation of applications for a certain software package [or] software framework...." Please see the Wikipedia entries for [API](#) and [SDK](#) for full definitions.)

A nice solution to these challenges is [PhoneGap](#), an open-source mobile development framework. Either by downloading the PhoneGap SDK or, perhaps more intriguingly, uploading your code to PhoneGap for building for a variety of mobile platforms, you can leverage the same code across a variety of vendor-specific device APIs. In short, you are writing native phone apps with familiar (HTML, CSS, JavaScript) tools: PhoneGap exposes native mobile device APIs and data to JavaScript, offering write-once-run-anywhere development of "native" apps for all popular major platforms.

PhoneGap offers access to the accelerometer, camera, compass, contacts, network, and other phone features - check out the [supported features list](#) on the PhoneGap website for details on specific features and supported platforms.

The PhoneGap API abstracts functionality across the range of supported platforms, allowing us to work with a single JavaScript API - with PhoneGap handling the device-specific implementation details. For instance, to take a photo and retrieve the resulting image file location, we might write code like this:

```
navigator.camera.getPicture(onSuccess, onFail, { quality: 50, destinationType: Camera.DestinationType.FILE_URI });
function onSuccess(imageURI) {
    var image = document.getElementById('myImage');
    image.src = imageURI;
}
function onFail(message) {
    alert('Failed because: ' + message);
}
```

Function `getPicture` both takes the photo and specifies callback functions (as the first and second parameters) for the success and failure cases. Function `onSuccess` - called when a picture is successfully taken - sets the `src` of the DOM element with `id myImage` to be the image we just captured. Function `onFail` - called when the photo wasn't successfully taken - shows the error message as a JavaScript alert.

PhoneGap Build

[PhoneGap Build](#) is a service through which you can upload HTML/CSS/JavaScript apps (written, perhaps, in jQuery Mobile), compile them in the cloud, and "get back app-store ready apps for Apple iOS, Google Android, Windows Phone 7, Palm, Symbian, BlackBerry and more". No need to download an SDK (as is the case with the core PhoneGap offering) - PhoneGap Build returns app-ready code for you, from your uploaded application. A free pricing plan offers unlimited public apps and one free app; various paid options exist for more private apps and larger groups.

To use PhoneGap Build, first [create a free account](#) and log in. Next, you'll create a project and upload to PhoneGap Build for building. As [the docs](#) state, your project will consist of:

- `Index.html` (the main page of your app).
- Any other assets your app uses - JavaScript or CSS files, images, audio, video, and whatnot.
- A `config.xml` file, based on the W3C widget spec, that contains data about your application.
- An app icon image - png files are the widest supported, and your best bet for now.

A good way to start is to download the sample project from its [GitHub repository](#) - clone the repository (if you have a GitHub account) or

download the zipped copy. The sample app isn't written using jQuery Mobile, but (as you'll see) is written in HTML, CSS, and JavaScript. To test it out, try making a quick change to the JavaScript (in [assets/js/app.js](#)) and/or updating the icon ([icon.png](#)) for the project. You might also make some changes to the XML config file ([config.xml](#)) to customize it: update the name, description, version, and/or id to values more specific to you.

Once done with changes to the project, create a new app: when logged in to PhoneGap Build, navigate to the [Apps page](#) and click the "New App" button. Give the app a name, choose "upload an archive or index.html file," browse to an archive (zipped) copy of the app you just modified, and click the "Create" button. Your code will be uploaded. PhoneGap Build will build your app for native use on Android, Windows Phone, BlackBerry, webOS, and Symbian. You'll see a screen similar to the one below, from which you can download the app for specific devices. Note that you will need a certificate and provisioning profile from Apple to build for iOS; you can find [more information on the PhoneGap website](#).

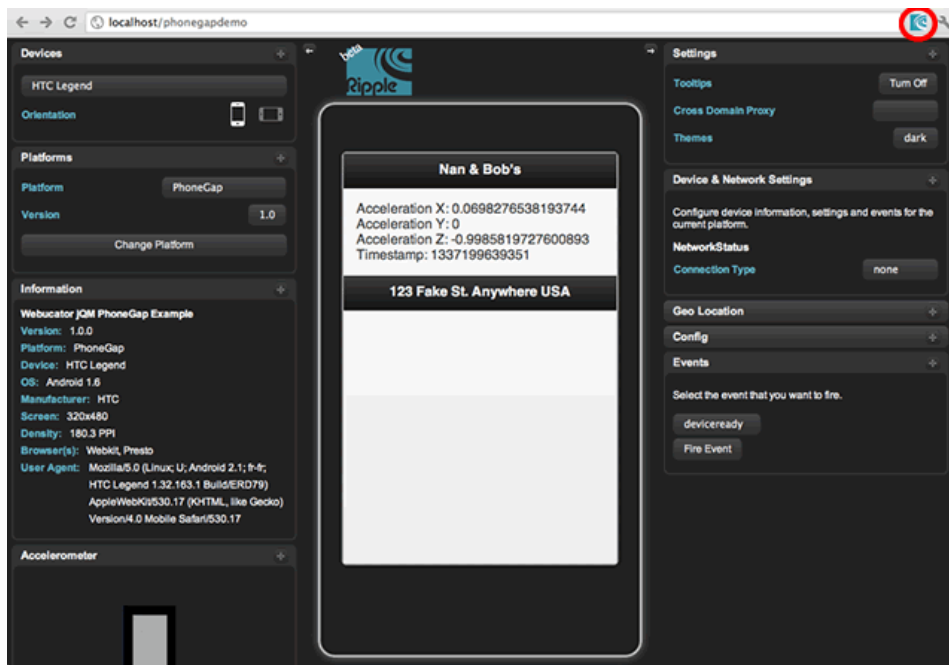
The screenshot shows the PhoneGap Build web interface. At the top, there's a navigation bar with 'PhoneGap: Build BETA', 'Apps', 'Docs', 'Blog', and 'Help'. A user is logged in as 'bhoke@bentleyhoke.com' with a 'sign out' link. The main header says 'PhoneGap: Getting Started' with an 'Edit' button. Below this, it states 'A template for getting started with PhoneGap development and build.phonegap.com' and 'com.phonegap.getting.started at 1.0.0; running on PhoneGap 1.7.0 last built at Jun 11, 11:22 PDT (5 total builds)'. There are 'ADMIN' and 'PRIVATE' tabs, and 'Update code' and 'Rebuild all' buttons. The interface is divided into sections for different platforms: iOS, Android, Windows phone, and BlackBerry. Each platform section has a 'rebuild' button and a QR code. The iOS section has a message: 'You must provide a signing key, fi...' and a 'more info' button. The Android section has a 'Download apk' button. The Windows phone and BlackBerry sections also have QR codes.

Testing with Ripple

Even with the convenience of creating native apps in the cloud with PhoneGap Build, testing the resulting apps on actual devices can be tough - actually getting an app from PhoneGap Build (or PhoneGap) onto an iPhone, Android, Blackberry, or other device can be a difficult process; iPhone apps, in particular, require that you submit a certificate and a provisioning profile from the Apple Developer program. Of course, there's no substitute for thorough testing of that new app on real devices - but it would be nice if there were a convenient, web-based system upon which we could perform first-pass checking of our code.

Luckily, there is Ripple, a free plugin for Google Chrome. Their [website](#) states that "Ripple is a multi-platform mobile environment emulator that runs in a web browser and is custom-tailored to HTML5 mobile application testing. Ripple aims to reduce the challenges being faced by mobile developers caused by today's platform fragmentation in the marketplace." Ripple isn't perfect - it isn't, of course, an actual smartphone device - but, in our experience, it's a useful, convenient tool for first testing of PhoneGap-type apps.

Assuming that you are using Google Chrome, you can download and install Ripple for free from the [Google Chrome Web Store](#). Once installed, you can view any site in Ripple by toggling the blue Ripple icon that will have been added to your browser:



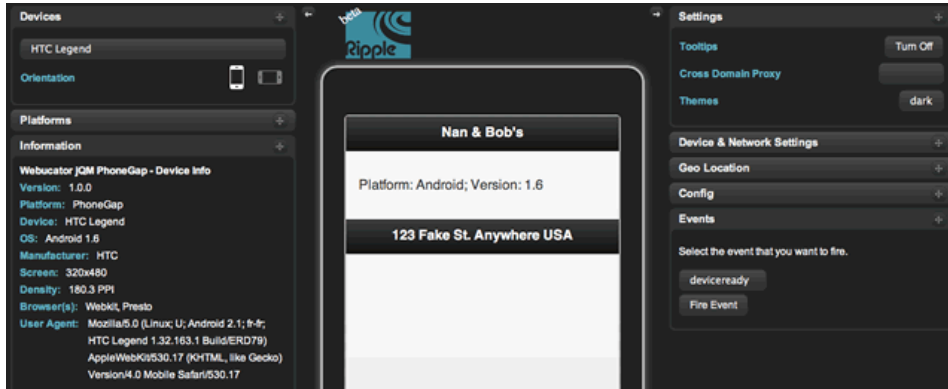
Ripple supports a wide variety of device features when viewing PhoneGap apps, from accelerometer and compass to contacts and geolocation. The Ripple website has a complete [list of supported features](#) from the PhoneGap API.

We'll use Ripple throughout this lesson, so we won't need to build apps through PhoneGap. Supporting the PhoneGap platform as it does, Ripple recognizes the PhoneGap API hooks in our code.

Note that we'll need to use Ripple on pages viewed from a web server - accessing pages through a local file system (as you may have been doing up to this point) won't work. You'll need to publish our examples and your work to a web server via FTP; please refer to the setup instructions for this course for more details.

Lesson 1, Activity 3: A First Example

Let's pick an easy item from the [PhoneGap API docs](#) for our first example: we'll get the operating system and version information from their mobile device. The [device API page](#) lists options available to us - we'll use `device.platform` and `device.version`. Here's a screenshot of the Ripple view of our app:



We'll create a simple, one-page jQuery Mobile site in a file [index.html](#). Since this is now an app - and may be used offline, we will download the various JavaScript and CSS files and include them locally, rather than linking to the CDN-hosted versions.

We also include the file [config.xml](#) in our project - PhoneGap uses this file for configuration details about the app. (Note that you can also set these details in the PhoneGap Build control panel.) The root element of this XML document must be `widget`. The file should reside at the root level of your application. See the [PhoneGap Build docs](#) for more information.

With the housekeeping out of the way, let's look at some code. Open [PhoneGap/Demos/deviceinfo/index.html](#) to check it out.

Code Sample:

[PhoneGap/Demos/deviceinfo/index.html](#)

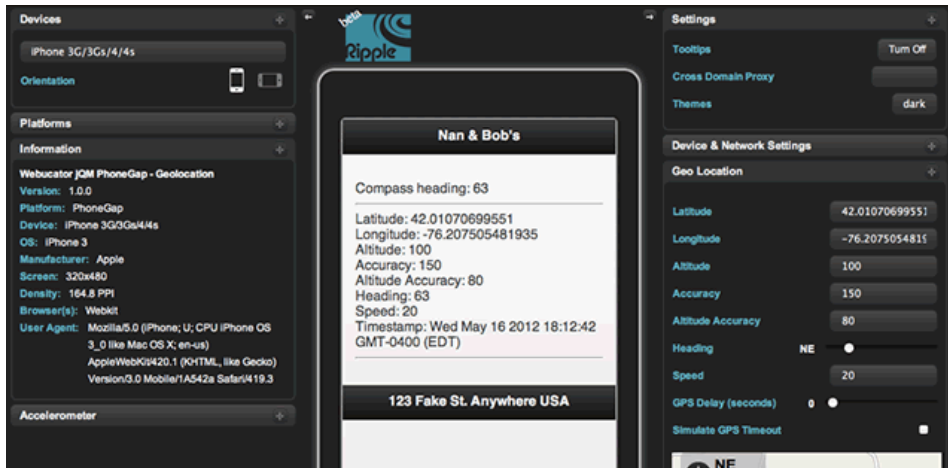
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Nan & Bob's</title>
    <link rel="stylesheet" href="jquery.mobile-1.1.0.min.css" />
    <script src="jquery.min.js"></script>
    <script src="jquery.mobile-1.1.0.min.js"></script>
    <script type="text/javascript">
      document.addEventListener("deviceready", onDeviceReady, false);
      function onDeviceReady() {
        $('#info').html('Platform: ' + device.platform + '; Version: ' + device.version + '');
      }
    </script>
  </head>
  <body>
    <div data-role="page" id="home">
      <div data-role="header">
        <h2>Nan & Bob's</h2>
      </div>
      <div data-role="content">
        <p id="info">Loading info</p>
      </div>
      <div data-role="footer">
        <h3>
          123 Fake St. Anywhere USA
        </h3>
      </div>
    </div>
  </body>
</html>
```

The markup of the page should look pretty familiar - we've only one page that, in its main `content` section, has a single paragraph with `id` `info`. We add an event handler to run on `deviceready`, which fires when PhoneGap is fully loaded - "a very important event," as the [PhoneGap docs](#) note, "that every PhoneGap application should use."

After `deviceready` fires, we use jQuery's `html` method to set the text of the `info` paragraph to a string we construct from `device.platform` and `device.version`. Try changing the "devices" option on Ripple (at upper left) - the page will refresh and the platform/version information will change accordingly.

Lesson 1, Activity 5: **Geolocation**

Ripple allows us to simulate a position - the "Geo Location" panel on the right allows you to set the latitude, longitude, altitude, speed, and other attributes of the simulated device's current location. You can even simulate a GPS failure (the "Simulate GPS Timeout" checkbox), to test how your apps handle the case when no geolocation information is available. Here's a screenshot of our app:



Open PhoneGap/Demos/geolocation/index.html to view an example:

Code Sample:

PhoneGap/Demos/geolocation/index.html

```

---- CODE OMITTED ----

<script type="text/javascript">
  document.addEventListener("deviceready", onDeviceReady, false);
  function onDeviceReady() {
    navigator.compass.getCurrentHeading(onCompassSuccess, onCompassError);
    navigator.geolocation.getCurrentPosition(onGeoSuccess, onGeoError);
  }
  function onGeoSuccess(position) {
    var element = document.getElementById('geolocation');
    $('#info').append('Latitude: ' + position.coords.latitude + '<br />' +
      'Longitude: ' + position.coords.longitude + '<br />' +
      'Altitude: ' + position.coords.altitude + '<br />' +
      'Accuracy: ' + position.coords.accuracy + '<br />' +
      'Altitude Accuracy: ' + position.coords.altitudeAccuracy + '<br />' +
      'Heading: ' + position.coords.heading + '<br />' +
      'Speed: ' + position.coords.speed + '<br />' +
      'Timestamp: ' + new Date(position.timestamp) + '<br />');
  }
  function onGeoError(error) {
    alert('code: ' + error.code + '\n' + 'message: ' + error.message + '\n');
  }
  function onCompassSuccess(heading) {
    $('#info').append('Compass heading: ' + heading + '<hr />');
  }
  function onCompassError() {
    alert('onError!');
  }
</script>
</head>
<body>
  <div data-role="page" id="home">
    <div data-role="header">

```

```

        <h2>Nan &amp; Bob's</h2>
    </div>
    <div data-role="content">
        <p id="info"></p>
    </div>
---- C O D E   O M I T T E D ----

```

On `deviceready`, we invoke the `compass.getCurrentHeading` and the `geolocation.getCurrentPosition` PhoneGap methods. Each method works in a similar manner: the two parameters supplied refer to methods to handle the success and failure cases of the method, respectively. We name the methods ourselves - we just need to be consistent in passing the names of our methods as parameters to the PhoneGap API methods.

For the Geolocation method, we append to the `info` paragraph details about the current position; these details are exposed through the `position.coords` parameter. Here are the available properties, along with units:

Position Properties

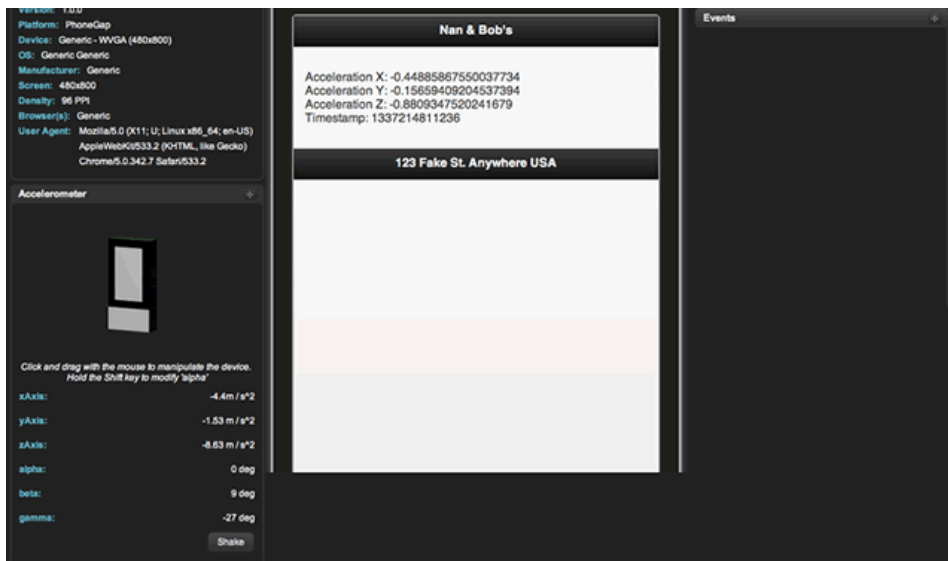
Property	Units
<code>coords.latitude</code>	degrees
<code>coords.longitude</code>	degrees
<code>coords.altitude</code>	meters (may be null)
<code>coords.accuracy</code>	meters
<code>coords.altitudeAccuracy</code>	meters (may be null)
<code>coords.heading</code>	degrees clockwise (may be null)
<code>coords.speed</code>	meters/second (may be null)
<code>timestamp</code>	a date and time

Similarly, we use the compass to find the current heading and append it to the `info` paragraph - not needed, really, since we get it from the GPS data.

Refer to the PhoneGap API docs on [Geolocation](#) and [Compass](#) for more information.

Accelerometer

Lots of cool stuff available to us via the [accelerometer API](#): "The accelerometer is a motion sensor that detects the change (delta) in movement relative to the current device orientation. The accelerometer can detect 3D movement along the x, y, and z axis." We can both capture the current state of the device or implement a watch that checks the acceleration at a specified interval. Here's a demo - open up [PhoneGap/Demos/accelerometer/index.html](#) to see the code.



Code Sample:

PhoneGap/Demos/accelerometer/index.html

```

---- C O D E   O M I T T E D ----

<script type="text/javascript">

    document.addEventListener("deviceready", onDeviceReady, false);
    function onDeviceReady() {
        startWatch();
    }

    function startWatch() {
        var options = { frequency: 1000 };
        watchID = navigator.accelerometer.watchAcceleration(onSuccess, onError, options);
    }

    function stopWatch() {
        if (watchID) {
            navigator.accelerometer.clearWatch(watchID);
            watchID = null;
        }
    }

    function onSuccess(acceleration) {
        $('#info').html('Acceleration X: ' + acceleration.x + '<br />' +
            'Acceleration Y: ' + acceleration.y + '<br />' +
            'Acceleration Z: ' + acceleration.z + '<br />' +
            'Timestamp: ' + acceleration.timestamp + '<br />');
    }

    function onError() {
        alert('onError!');
    }

</script>
</head>
<body>
    <div data-role="page" id="home">
        <div data-role="header">
            <h2>Nan & Bob's</h2>
        </div>
        <div data-role="content">
            <p id="info">Waiting for accelerometer...</p>
        </div>
    </div>

---- C O D E   O M I T T E D ----

```

On `deviceready`, our code calls method `startWatch()`, which in turn calls `navigator.accelerometer.watchAcceleration`. A successful capture of the accelerometer information invokes method `onSuccess`, from which we update the contents of the `info` paragraph for display.

Note that you can simulate movement on Ripple from the lower-left "Accelerometer" panel, dragging or shaking the phone.


```

$('#addButton').click(function() {
    //when the "addButton" button is clicked, create a new contact
    var contact = navigator.contacts.create();
    //set the displayName to the value of the #fullname field
    contact.displayName = $('#fullname').val();
    //set the nickname to the value of the #nickname field
    contact.nickname = $('#nickname').val();
    //save the contact
    contact.save(onContactAddSuccess,onContactAddError);
});

function onContactAddSuccess(contact) {
    $('#fullname').val('');
    $('#nickname').val('');
    alert('Contact Added');
}

function onContactAddError(contactError) {
    alert("Error = " + contactError.code);
}

function onSuccess(contacts) {
    if (contacts.length == 0) {
        $('#contacts').html('<li><em>No contacts matched your search</em></li>');
        navigator.notification.vibrate(100);
    } else {
        $('#contacts').html('');
        for (var i=0; i<contacts.length; i++) {
            $('#contacts').append('<li>'+contacts[i].displayName+'</li>');
        }
    }
}

function onError() {
    alert('onError!');
}

</script>
</head>
<body>
    <div data-role="page" id="home">
        <div data-role="header">
            <h2>Nan & Bob's</h2>
        </div>
        <div data-role="content">
            <ul id="contacts"></ul>
            <input type="text" name="search" id="searchBox">
            <button type="submit" id="searchButton">Search</button>
            <hr>
            <input type="text" name="text" id="fullname" placeholder="Full Name">
            <input type="text" name="text" id="nickname" placeholder="Nickname">
            <button type="submit" id="addButton">Add</button>
        </div>
    </div>
    ---- C O D E   O M I T T E D ----

```

`$('#searchButton').click(function() {` adds a click handler for the "search" button; we use similar code to add a click handler for the "add" button.

We use `$('#searchBox').val()` to get the value of the `searchBox` input; the same code gets the values of the `fullname` and `nickname` fields.

If adding a contact is successful, we set the `html` of the two "add" fields to "" and generate an `alert` with a success message.

Upon a successful search, if there are no matches we display a message in the `#contacts` list and shake the phone with `navigator.notification.vibrate(100)` - that is, vibrate the phone for 100 milliseconds. For a search that returns at least one result,

we loop over the `contacts` and append a list item to `#contacts`, showing the `displayName` of each contact.