ngCordova

Description:
ngCordova is not a plugin itself, but a Wrapper for native Cordova plugins.
It makes the life a lot easier.

With ngCordova, instead of calling Cordova plugins directly and having to figure out the proper object or plugin name, or to check if the plugin is actually installed, you can just call a simple AngularJS service like this:

```
$cordovaCamera.getPicture(options)
  .then(function(imageData) {

        // Process camera data

  }, function(error) {

        // Show an error to the user

  });
```

A brandnew and maybe even better wrapper is ionic-native.
The description:
Ionic Native is a curated set of wrappers for Cordova plugins that make adding any native functionality you need to your Ionic, Cordova, or Web View mobile app easy.
https://github.com/driftyco/ionic-native

How we included it:

Put it in your index.html

```
<script type="text/javascript" src="lib/ngCordova/dist/ng-cordova.js"></script>
<script type="text/javascript"
src="lib/ng-cordova-ble/ng-cordova-bluetoothle.js"></script>

// The ng-Cordova scripts need to be included BEFORE the cordova.js script.
Otherwise it will fail.
<script type="text/javascript" src="cordova.js"></script>
```

BluetoothLE

Description:
This plugin allows you to interact with Bluetooth LE devices on Android, iOS, and partially on Windows.

Requirements

- Cordova 5.0.0 or higher
- Android 4.3 or higher, Android Cordova library 5.0.0 or higher, target Android API 23 or higher
- iOS 7 or higher
- Windows Phone 8.1 (Tested on Nokia Lumia 630)
- Device hardware must be certified for Bluetooth LE. i.e. Nexus 7 (2012) doesn't support Bluetooth LE even after upgrading to 4.3 (or higher) without a modification

What we changed:
nothing

Why we chose it:

We first used https://github.com/don/cordova-plugin-ble-central
But it was hard to interact with Angular.

A better way would be to use the ngCordovaBluetoothLE Plugin in the project to wrap the plugin.
Would save some trouble too we had in the beginning.

How to use it:
Add it to your controller:
.controller('TagCtrl', function($scope, $cordovaBluetoothLE)

Have a closer look to app.js and tag.js
The code is well documented.

Measuring decibel

- it does not work in a browser(obviously)
- it works on blackberry, ios, android
    - even in background
    - even in standby
    - even when playing music from another app.

What we changed:
nothing

How have you included it?
The cordova dbmeter plugin offers a variable `DBMeter` in global scope:

```
DBMeter.start(function(dB) {
    // this is called periodically whenever a new dB value was retrieved until
DBMeter.stop() is called.
    // so the next line will countinously log the loudness
    console.log('loudness:' + dB + 'dB');
});
DBMeter.stop();
```

DeviceMotion

Description:
This plugin provides access to the device's accelerometer. The accelerometer is a motion sensor that detects the change (delta) in movement relative to the current device orientation, in three dimensions along the x, y, and z axis.

Install:
cordova plugin add cordova-plugin-device-motion

Supported Platforms

- Amazon Fire OS
- Android
- BlackBerry 10
- Browser
- Firefox OS
- iOS
- Tizen
- Windows Phone 8
- Windows

Android Issue:
The accelerometer is called with the SENSOR_DELAY_UI flag, which limits the maximum readout frequency to something between 20 and 60 Hz, depending on the device. Values for period corresponding to higher frequencies will result in duplicate samples. More details can be found in the Android API Guide.

iOS Issue:
iOS doesn't recognize the concept of getting the current acceleration at any given point.
You must watch the acceleration and capture the data at given time intervals.
Thus, the getCurrentAcceleration function yields the last value reported from a watchAccelerometer call.

Why have you chosen it?
ngCordova offers this Plugin to meassure the Acceleration.
As ngCordova is the recommended way to use a plugin (without trouble) we chose this one.

What we changed:
nothing

How have you included it?
Include it in your Controller
.controller('TagCtrl', function($scope, $rootScope, $q, $cordovaDeviceMotion) {

Example Usage:

```
// Define your Success Function
function onSuccess(acceleration) {
	alert('Acceleration X: ' + acceleration.x + '\n' +
	'Acceleration Y: ' + acceleration.y + '\n' +
	'Acceleration Z: ' + acceleration.z + '\n' +
	'Timestamp: '	+ acceleration.timestamp + '\n');
}

// Define your Error Function
function onError() {
	alert('onError!');
}

// Create an options Object
var options = { frequency: 3000 };  // Update every 3 seconds (not more options
available yet)

// Start to watch the accelerometer data with the frequency of [options]
var watchID = navigator.accelerometer.watchAcceleration(onSuccess, onError,
options);

// Deregister from accelerometer again. Use the watchID you got from
watchAcceleration()
navigator.accelerometer.clearWatch(watchID);
```

Dialogs

Description:
This plugin provides access to some native dialog UI elements.

Supported Platforms

- Amazon Fire OS
- Android
- BlackBerry 10
- Browser
- Firefox OS
- iOS
- Tizen
- Windows Phone 7 and 8
- Windows 8
- Windows

Installation:
cordova plugin add cordova-plugin-dialogs

How to use:

navigator.notification.alert
navigator.notification.confirm
navigator.notification.prompt
navigator.notification.beep

Our "Activate Bluetooth" with Translation Example:

```
navigator.notification.confirm($translate.instant("PROMPT.TURN_ON_BLUETOOTH
"), function (buttonIndex) {
  if (buttonIndex == 1){
   enableFunction();
  }else if(buttonIndex == 0 || buttonIndex == 2){

navigator.notification.alert($translate.instant("PROMPT.APP_ONLY_WORKS_WITH
_BT"), function () { navigator.app.exitApp(); });
  }
```

}, $translate.instant("PROMPT.HEADER_ENABLE_BT"),
[$translate.instant("PROMPT.ACCEPT"), $translate.instant("PROMPT.CANCEL")]);

Why did we choose it:
We use it to interact with the user before the App is completly loaded.

For example:
We need to activate the BLE. Therefore we need a interaction with the user. The plguin helps us to do this in a native way.

Globalization

Description:
This plugin obtains information and performs operations specific to the user's locale, language, and timezone. Note the difference between locale and language: locale controls how numbers, dates, and times are displayed for a region, while language determines what language text appears as, independently of locale settings.

Installation:
cordova plugin add cordova-plugin-globalization

How to use:
```
// Get it in your Controller
module.controller('MyCtrl', function($cordovaGlobalization) {
  $cordovaGlobalization.getPreferredLanguage().then(
        function(result) {
        // result
        },
        function(error) {
        // error
  });

  $cordovaGlobalization.getLocaleName().then(
        function(result) {
        // result
        },
        function(error) {
        // error
  });

.....
```

Why we chose it:
Part of ngCordova

SQLite-Storage

Description:
Native interface to sqlite in a Cordova/PhoneGap plugin for Android, iOS, and Windows, with API similar to HTML5/Web SQL API.

Source: https://github.com/brodysoft/Cordova-SQLitePlugin.git

Install the Plugin:
cordova plugin add https://github.com/litehelpers/Cordova-sqlite-storage.git

Include it in your Controller:
.controller('TagCtrl', function($scope, $cordovaSQLite)

Angular Issue:
It was not possible to use a native cordova Plugin within a service.
So you cannot provide the openDB, createDB, queryDB methods in a service.
We used rootScope for a workaround.

iOS Issue:
We need to specify a location where to store the data, but only in IOS.
Why?
From A User´s iCloud Storage is Limited:
DO store the following in iCloud:
[other items omitted]
Change log files for a SQLite database (a SQLite database's store file must never be stored in iCloud)
DO NOT store the following in iCloud:
[items omitted]

So:

- Use the location or iosDatabaseLocation option in sqlitePlugin.openDatabase() to store the database in a subdirectory that is NOT backed up to iCloud

Options to choose:

- default: Library/LocalDatabase subdirectory - NOT visible to iTunes and NOT backed up by iCloud
- Library: Library subdirectory - backed up by iCloud, NOT visible to iTunes

- Documents: Documents subdirectory - visible to iTunes and backed up by iCloud

Example Usage:

```
// Open/Create a Database called my.db
// You need to specifiy a iosDatabaseLocation
var db = $cordovaSQLite.openDB({name: "my.db", iosDatabaseLocation: 'default'});

 $scope.execute = function() {
        var query = "INSERT INTO test_table (data, data_num) VALUES (?,?)";
        $cordovaSQLite.execute(db, query, ["test", 100]).then(function(res) {
        console.log("insertId: " + res.insertId);
        }, function (err) {
        console.error(err);
        });
 };
```

What did we change:
nothing

Why we chose it:
Included in ngCordova.
But, we first tried to use http://lokijs.org/#/ with an adapter for persistent storage.
Advantage: Have a fast in-memory-database for use.
Adapter: Have a wide range of choices how you want to store the data persistent.
Problem was: It did not run with our project setup. Even the default example threw errors.
But it is worth a look!