Creating a Modern PhoneGap Plugin

Kerri Shotts (@kerrishotts)

Jesse MacFadyen (@purplecabbage)

Slides at https://goo.gl/2xVAwZ

Based on PGDay EU 2016 plugin workshop by Jesse

Photo by AlexanderStein (https://pixabay.com/en/users/AlexanderStein-45237/), courtesy of Pixabay.com

About Kerri

- Used PhoneGap for six+ years
- Author of five books about PhoneGap
- IT Consultant for eight years
- Apache Cordova committer
- One of several moderators:
  - Adobe PhoneGap Forums
  - Google Cordova Group
- @kerrishotts

About Jesse

- PhoneGap Developer since 2008
- Apache Cordova committer
- At Adobe for nearly 6 years now
- @purplecabbage

What is a Cordova Plugin?

noun A mystical collection of machine incantations which grant access to amazing and magical capabilities

ahem...

noun A module consisting of code and settings extending the essential functionality of Cordova with the goal of providing access to device capabilities, enhancing existing capabilities, or improving the developer's workflow

What can plugins do?

- Anything native code can do
- Active in the following contexts:
  - run time
  - build time
  - install time
- Two sources of Plugins
  - Core — used to be built in pre-3.x (Undergoing audit: CB-12708)
  - Community — people like you!

Plugins at Run Time

Full access to the native SDK and device features. Some examples:

- Push Notifications: PhoneGap, Pushwoosh, AeroGear, OneSignal
- Storage Plugins: Native Storage, SQLite, SQLite 2
- Social Plugins: Email, X SocialSharing
- Audio Plugins: DBMeter, Native Audio, Media Picker
- Misc: Barcode Scanner, In App Purchase, Google Maps, Vuforia (AR), Microsoft ACE (native controls), Tesseract (OCR, iOS)
- Creative Cloud: Auth, Asset Browser, Image Editor, Send to Desktop

Photo by skeeze (https://pixabay.com/en/users/skeeze-272447/), courtesy of Pixabay.com

Plugins at Build Time

Full access to the build-time environment and Cordova project. Some examples:

- Transpile and Bundle ES2015+: Webpack & Transpiler plugin
- Pre-process CSS files (SASS, less, auto-prefixer)
- Check code quality (eslint, tslint, jshint)
- Etc.

Photo by Pexels (https://pixabay.com/en/users/Pexels-2286921/), courtesy of Pixabay.com

Plugins at Install Time

Full access to the Cordova project and environment at install time. Some ideas:

- Configure the project environment
- Bundle other plugins
- Provide tests for another plugin...
  - cordova-plugin-test-framework

Plugin-ception 🎇

Photo by PublicDomainPictures (https://pixabay.com/en/users/PublicDomainPictures-14/), courtesy of Pixabay.com

The Core Plugins

Core Cordova Plugins (used to be built-in pre-3.x) (Audit: CB-12708)

battery-status	camera	console
contacts	device	device-motion
device-orientation	dialogs	file
file-transfer	geolocation	globalization
inappbrowser	media	media-capture
network-information	~~splashscreen~~	statusbar
vibration	whitelist	

Community Plugins

Devoloped and supported by the community — like you!

Repository	Plugins
https://cordova.apache.org/plugins	~2,211 plugins & templates (excl. core)
http://www.plugreg.com	~1,592 plugins (excl. core)
http://plugins.telerik.com/cordova	~77 plugins

Managing Plugins

npm

Plugins are typically downloaded from npm:

```
$ cordova plugin add [--save] cordova-plugin-device

$ cordova plugin ls                                  # or list
  cordova-plugin-device 1.1.1 "Device"

$ cordova plugin rm [--save] cordova-plugin-device # or remove
```

Note: `--save` persists the plugin to `config.xml` so that plugins can be easily restored (done at `prepare`-time)
7.x: `--save` is the default in `cordova@7.0.0`; `--nosave` will turn it off
7.x: Fetching via npm is now the default; if a plugin doesn't have `package.json` adding will fail. Use `--nofetch` for those plugins.

Git

Plugins can also be installed from a Git repository.

```
$ cordova plugin add http://github.com/apache/cordova-plugin-device

$ cordova plugin rm cordova-plugin-device
```

Specify a branch: (useful for testing pre-release/edge plugins):

```
$ cordova plugin add http://github.com/apache/cordova-plugin-device
    #branch
```

Note: Use the plugin's identifier when removing — not the URL.

Local Filesystem

Or install from the local file system — very useful for plugin development.

```
$ cordova plugin add [--save] [--link] /path/to/plugin

$ cordova plugin rm [--save] cordova-plugin-device
```

Note: `--link` is useful when developing plugins

Tip: Adding a plugin to a child project (relative to the plugin) may automatically symlink the plugin
Note: Careful with parent plugins and child projects — easy to get circular references in the file system
Windows: May need to be admin to create links

Finding Plugins

- Cordova Plugin Search: https://cordova.apache.org/plugins
- npm: https://www.npmjs.com/search?q=ecosystem:cordova
- Or, if the CLI is more your thing:

```
$ npm install -g npms-cli
$ npms search cordova-plugin device --size=5

┌──────────────
```

Crossing the bridges

Photo by kaboompics (https://pixabay.com/en/users/kaboompics-1013994/), courtesy of Pixabay.com

Know your Bridges

Allows communication between native code and web view contexts.

- iOS
- Android
- Browser/Windows is an exception...
  - Careful, the bridge is a **mirage**! 🏜️
  - JavaScript is **native** 🌟
  - `cordova.exec` uses a proxy to keep things consistent
  - full docs

Cordova iOS Bridge (abridged)

Cordova Android Bridge (abridged)

Cordova Windows / Browser “Bridge” (abridged)

Tests

Cordova Test Harness

cordova-paramedic
cordova-plugin-test-framework

Test Cases

Your Jasmine tests
Automatic & Manual

Plugin X-ray

Photo by dcondrey (https://pixabay.com/en/users/dcondrey-122249/), courtesy of Pixabay.com

The Stuff Plugins are Made of

Metadata	Documentation [s]
Native Code *	JavaScript *
Tests [s]	Hooks *
Typings *	TLC * [s]

* Optional

[s] Optional but highly suggested

Plugin Structure	Description
cordova-plugin-*your-plugin*/	Plugin root
package.json	npm metadata
plugin.xml	Plugin metadata and configuration
README.md	English documentation
doc/*locale*	Documentation other than English
tests/	*Please add tests!*
types/	TypeScript typings
src/*platform*	Platform-specific native code
android/	Native Android code
YourPlugin.java	
ios/	Native iOS code
CDV*YourPlugin*.h	
CDV*YourPlugin*.m	
www/	JavaScript code & web assets
yourPlugin.js	API for JavaScript consumers

(representational; not every file is included here); Ex: Device Plugin

Metadata

plugin.xml

id, **version**, author, license, name, description, repo, issue, keywords, platform (& assets), dependencies, engines, preferences, hooks, info, etc.

package.json

name, **version**, author, license, description, repository, issue, keywords, platforms, dependencies

Note: `package.json` can be generated by `plugman`

Example Metadata (plugin.xml)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<plugin xmlns="http://apache.org/cordova/ns/plugins/1.0"
  xmlns:android="http://schemas.android.com/apk/res/android"
  id="cordova-plugin-device" version="1.1.5-dev">
  <name>Device</name>
  <description>Cordova Device Plugin</description>
  <license>Apache 2.0</license>
  <keywords>cordova,device</keywords>
  <repo>https://git-wip-us.apache.org/repos/asf/
    cordova-plugin-device.git</repo>
  <issue>https://issues.apache.org/jira/browse/CB/
    component/12320648</issue>
```

npm Metadata Example (package.json)

```
1  { "name": "cordova-plugin-device",
2    "author": "Apache Software Foundation",
3    "license": "Apache-2.0",
4    "version": "1.1.5-dev",
5    "description": "Cordova Device Plugin",
6    "types": "./types/index.d.ts",
7    "cordova": {
8      "id": "cordova-plugin-device",
9      "platforms": ["android", "ios", "windows", "wp8", ... ] },
10   "repository": { "type": "git", "url": "https://..." },
11   "keywords": ["cordova", "device", "ecosystem:cordova",
12                "cordova-ios", "cordova-android", ... ],
```

Device Plugin's package.json

JavaScript Modules

Automatically injected into your consumer's `index.html`. docs

```
<js-module src="www/device.js" name="device">
  [<children/>]
</js-module>
```

Children	Description
`<clobbers target="device" />`	overwrites `window.device`
`<merges target="device" />`	merges with `window.device`
`<runs />`	runs, but doesn't export

- Unless necessary, target `cordova.plugins.yourPlugin`

Platform Support

Use `<platform>` tags: docs

```
<platform name="android">
  ...
</platform>
<platform name="ios">
  ...
</platform>
```

Note: You should also indicate platform support in `packge.json`: `cordova.platforms`

Assets and Native Code

```
1  <platform name="android">
2    <source-file src="src/android/Device.java"
3      target-dir="src/org/apache/cordova/device" />
4  </platform>
5  <platform name="ios">
6    <header-file src="src/ios/CDVDevice.h" />
7    <source-file src="src/ios/CDVDevice.m" />
8    <framework src="libz.tbd" />
9  </platform>
```

Other asset tags: `asset`, `resource-file`, `lib-file`; full docs
Note: You can include third-party libraries; iOS supports Cocoapods, and Android supports AARs with Gradle.
Bug: On iOS hidden (dot) files may not be copied. See CB-10135

Plugin Class Mapping

- Android (Geolocation)

```
<config-file target="res/xml/config.xml" parent="/*">
  <feature name="Geolocation">
    <param name="android-package"
      value="org.apache.cordova.geolocation.Geolocation" />
  </feature>
</config-file>
```

- iOS (Geolocation)

```
<config-file target="config.xml" parent="/*">
  <feature name="Geolocation">
    <param name="ios-package" value="CDVLocation"/>
  </feature>
</config-file>
```

Manifest Modifications

- `config-file` [1] docs
  - Adds elements to manifests / plist or platform `config.xml`

```xml
<config-file target="AndroidManifest.xml" parent="/*">
  <uses-permission android:name=
    "android.permission.WRITE_EXTERNAL_STORAGE" />
</config-file>
```

```xml
<config-file target="*-Info.plist"
  parent="NSLocationWhenInUseUsageDescription">
  <string>$GEOLOCATION_USAGE_DESCRIPTION</string>
</config-file>
```

1: android, file transfer; ios, geolocation; windows, geolocation

Manifest Modifications (2)

- `edit-config` [1 docs]
  - Edits attributes of existing elements in manifests

```
<edit-config file="AndroidManifest.xml"
  target="/manifest/application/activity     \
    [@android:name='MainActivity']"
  mode="merge">
  <activity android:theme="@style/AppTheme" />
</edit-config>
```

Dependencies (cordova@6.1.0+)

Dependencies are managed in `package.json`: docs

```
"engines": {
  "cordovaDependencies": {
    "2.0.0": { // plugin version (applies to any ver 2+)
      "cordova-plugin-console": ">1.0.0",
      "cordova": ">6.0.0" // cordova-cli above version 6
    }
  }
}
```

Documentation

Documentation is absolutely critical!

- Location of documentation
  - English goes in `README.md` (plugin root)
  - Other languages in `docs/[locale]/README.md`
- Provide examples, constants, errors that can be thrown, etc.

Creating and Publishing Plugins

💶 And getting rich, maybe? 💶

Or maybe not...

Demo Time

cordova-plugin-example-isprime

Photo by PeteLinforth (https://pixabay.com/en/users/PeteLinforth-202249/), courtesy of Pixabay.com

plugman

plugman is a `node` library that manages plugins in your projects. `cordova-cli`, `phonegap-cli`, etc., use `plugman` internally.

It is also used to create an initial plugin project:

```
$ npm install -g plugman
$ mkdir isprime
$ plugman create --name IsPrime
                 --plugin_id cordova-plugin-example-isprime
                 --plugin_version 0.0.1
                 --path .
```

phonegap-plugin-template

Or, use PhoneGap's plugin template to create a plugin:
https://github.com/phonegap/phonegap-plugin-template

```
$ npm i -g https://github.com/phonegap/phonegap-plugin-template

  #parms: path name plugin-id
$ phonegap-plugin-create isprime IsPrime
    cordova-plugin-example-isprime
  ? license[MIT] [enter]
```

Creates `docs`, `src/android`, `src/ios`, `www`, `plugin.xml`, `package.json`, and `README.md` (as well as some dot files)

JavaScript Code
Consumer API
Consumer-facing functions and methods
cordova.plugins.yourPlugin
Cordova Internals
cordova.exec
result

Your Plugin's JS API

```
// www/isPrime.js
var exec = cordova.require("cordova/exec"),
    SERVICE = "IsPrime";
module.exports = function isPrime(successFn, failureFn,
  candidate) {
  // ensure the arguments are of the correct types
  if (typeof successFn !== "function") throw new Error("!");
  /* etc. */
  var arg = { isPrime: false, candidate: candidate, ... };
  /* pass the call over the cordova bridge */
  exec(successFn, failureFn, SERVICE, "isPrime", [arg]);
}
```

Native Code
Cordova Interface
request
result
Plugin Code
Receive request
Process request
Return result
Throw errors

Your Native Code (iOS)

```
#import <Cordova/CDV.h>
@interface CDVIsPrime : CDVPlugin
@end
@implementation CDVIsPrime
- (void)isPrime:(CDVInvokedUrlCommand*)command {
  NSMutableDictionary* result = [[command argumentAtIndex:0] mutableCopy];
  NSMutableArray* factors = result[@"factors"];
  int64_t candidate = [result[@"candidate"] longLongValue];
  /* let there be a miracle: calculate if prime is a candidate */
  CDVPluginResult* r = [CDVPluginResult
    resultWithStatus:CDVCommandStatus_OK messageAsDictionary: result];
  [self.commandDelegate sendPluginResult:r callbackId:command.callbackId];
}
@end
```

Your Native Code (Android)

```
package com.example.isprime; /* omitting imports */
public class IsPrime extends CordovaPlugin {
  @Override
  public boolean execute(String action, JSONArray args,
    CallbackContext callbackContext) throws JSONException {
    if ("isPrime".equals(action)) {
      this.isPrime(args.getJSONObject(0), callbackContext);
    } else { return false; }
    return true;
  }
  private void isPrime(JSONObject result, CallbackContext callbackContext)
    throws JSONException {
    /* magic incantation: determine if candidate is prime */
    PluginResult pluginResult = new PluginResult(PluginResult.Status.OK, result);
    callbackContext.sendPluginResult(pluginResult);
  }
}
```

Your Native Code (Browser / Win)

```
//src/[browser|windows]/yourPluginProxy.js
function isPrime(successFn, failureFn, args) {
  var result = args[0],
      candidate = result.candidate;
  /* magic! calculate if candidate is prime */
  successFn(result);
}

module.exports = { isPrime: isPrime };

require("cordova/exec/proxy").add("IsPrime", module.exports);
```

Plugin Class Mapping (iOS)

Remember the JS API's call to `cordova.exec` and plugin class mapping in `plugin.xml`?

```
cordova.exec(successFn, failureFn, “IsPrime”, “isPrime”, [args])
```

plugin.xml

```
<feature name=“IsPrime”>
  <param name="ios-package" value=“CDVIsPrime”/>
</feature>
```

CDVIsPrime.m

```
@interface CDVIsPrime : CDVPlugin

- (void) isPrime:(CDVInvokedUrlCommand*)command

@end
```

Plugin Class Mapping (Android)

```
cordova.exec(successFn, failureFn, “IsPrime”, “isPrime”, [args])
```

plugin.xml

```
<feature name=“IsPrime”>
  <param name="android-package" value=“com.example.isprime.IsPrime” />
</feature>
```

IsPrime.java

```
package com.example.isprime;

public class IsPrime extends CordovaPlugin {

  public boolean execute(final String action, ...) {

    if (“isPrime”.equals(action)) {...}

  }

}
```

Triggering callback more than once

```
// iOS
CDVPluginResult* r=[CDVPluginResult resultWithStatus:CDVCommandStatus_OK
  messageAsDictionary:result];
[r setKeepCallbackAsBool:YES];

// Android
PluginResult r = new PluginResult(PluginResult.Status.OK, result);
r.setKeepCallback(true);

// Browser / Windows
successFn(result, {keepCallback: true});
```

Things your plugin should do

- If your plugin does a lot of work, use a background thread
- You should respond to `pause` and `resume` events
  - Respect the Android lifecycle! docs
- You should respond to `onDestroy` and `onReset` as well
  - `onDestroy` occurs when the plugin is about to go away
  - `onReset` occurs when the web view is about to navigate
  - Great for cleaning up background operations

Testing plugins

`cordova-medic` is a test tool designed to run all the core Cordova plugin tests as part of Cordova's continuous integration system

- Tests are written in Jasmine 2.0
- Tests run asynchonously
- Plugins have a dependent test plugin which is installed separately (usually in `/tests` by convention)
- Many of these pieces of `cordova-medic` are reusable, so Jesse spun them into another purpose-based tool...

cordova-paramedic

n. provides advanced levels of care at the point of illness or injury, including out-of-hospital treatment, and diagnostic services

```
$ npm i -D cordova-paramedic
$ npm i -D https://github.com/apache/cordova-paramedic.git
```

Then:

```
$ ./node_modules/cordova-paramedic/main.js --platform ios
    --plugin .
```

Repo & docs: https://github.com/apache/cordova-paramedic

cordova-paramedic

Or, just use an npm script in `package.json`:

```
"scripts": {
    "test:ios": "cordova-paramedic --platform ios     \
      --plugin .",
}
```

And then:

```
$ | npm run test:ios
```

Automates Jasmine Tests

- Creates a new project (in temporary location)
- Adds the platform specified (`ios` , `android` , `windows` , etc.)
- Installs the `cordova-plugin-test-framework` plugin
- Installs the plugin specified (in `.`) (current working directory)
- Installs the plugin's tests (in `./tests`)
- Sets start page to `cordova-plugin-test-framework` 's test runner
- Creates a local server to listen for results
- Exits with success/fail based on results

Note: Only supports npm-published platforms

How to write tests

- Copy a core plugin's tests – we all do it!
- Create a `tests` folder in your plugin's repository
- Add a `package.json` file (shouldn't be complex)
- Add a `plugin.xml` file (doesn't need to be complex) eg

```
<plugin xmlns="http://apache.org/cordova/ns/plugins/1.0"
xmlns:android="http://schemas.android.com/apk/res/android"
id="cordova-plugin-statusbar-tests" version="2.2.3-dev">
  <name>Cordova StatusBar Plugin Tests</name>
  <license>Apache 2.0</license>
  <js-module src="tests.js" name="tests"></js-module>
</plugin>
```

Testing Tips

- Automate as much as you can (`exports.defineAutoTests`)
- For tests that can't be automated, use manual tests (`exports.defineManualTests`)
- Don't forget to accept & call `done` in your `it` tests when working with callbacks and promises.
- If you've got similar tests, you can build them programatically
- For native UI, you can use Appium
- Travis CI example (iOS & Android only)

Debugging & Iterating

Photo by ROverhate (https://pixabay.com/en/users/ROverhate-1759589/), courtesy of Pixabay.com

Debugging & Iterating

- Create an example app that uses your plugin

```
cordova create hello com.example.hello hello
cd hello
cordova platform add ios android browser
cordova plugin add --link /path/to/plugin
```

  - `--link` — can simplify dev workflow
  - If you change your example's `www`, be sure to `prepare`

IDEs & Debugging

- iOS: Xcode / Safari

```
$ open ./platforms/ios/*.xcworkspace
```

- Android: Android Studio / Google Chrome

```
$ open -a "Android Studio" "./platforms/android/"
```

- Windows (universal): Visual Studio

```
> start .\platforms\windows\CordovaApp.sln
```

What gets --linked?

- app's `plugins/<your-plugin>` is symlinked to your plugin
- Native code in symlinked in app's `platforms/`

Exceptions & notes:

- `cordova clean android` if the build complains
- `plugin.xml` changes require a `plugin rm` & `add`
- Changes to your plugin's `www` require an `rm` & `add` as well
- `platform rm` & `add` won't preserve `--link`s (CB-12597)

Photo by bernswaelz (https://pixabay.com/en/users/bernswaelz-1728198/), courtesy of Pixabay.com

Publishing your plugin

- Generate / update `package.json`
  - `plugman` can generate it based on `plugin.xml` for you:

```
$ plugman createpackagejson .
```

- Once `package.json` is correct, publish via:

```
$ npm publish
```

IMPORTANT:

- Don't forget `.npmignore`!
- If you used the PhoneGap Plugin Template, `package.json` is already there — you'll need to update it.

Tips & Tricks

Photo by Pexels (https://pixabay.com/en/users/Pexels-2286921/), courtesy of Pixabay.com

JavaScript API

- Promisify your API
- Preprocess arguments in JavaScript
  - convert to appropriate types
  - throw type-mismatch errors, etc.
- Transpile ES2015+ to ES5
- Stick to the `cordova.plugins` namespace
  - Unless creating a polyfill; **window** is crowded!
- Return useful error messages to error callbacks

Native

- Return useful error information
- Use background threads for processing
  - iOS documentation
  - Android documentation
- Avoid init at app startup unless necessary, but if you need to start up at start, you can:

```
<param name="onload" value="true" />
```

- Override `onReset` to clean up when web view navigates eg ios android

Native (Android)

- Override `pluginInitialize` for plugin initialization logic code
- Runtime Permission Requests (Marshmallow) docs
  - `cordova.requestPermission()` code
  - `cordova.hasPermission()` code
  - Override `onRequestPermissionResult` code
- Don't forget Android activity lifecycle docs code

Native (iOS)

- Use `pluginInitialize` for plugin initialization logic eg code
- If memory is getting low, `onMemoryWarning` is called code
- If app is going to be terminated, `onAppTerminate` is called code
- You can respond to `pause`, `resume`, etc. code, but you have to register for notifications in `pluginInitialize`
- If you need to handle URLs, override `handleOpenURL` code
- Never, *ever* call JavaScript that triggers blocking UI (e.g. `alert`)

Miscellaneous

- Don't forget the `browser` platform!
  - Useful when testing on the desktop
  - Mock results if no equivalent browser support

Hook

noun A piece of code that hooks into a Cordova process in order to perform some action on behalf of the plugin; see dev guide.

Possibilities:

- Create entitlements as needed
- Transform code (transpile, version # replacement, etc.)
- Create launch images and icons
- Check plugin versions and warn if out-of-date
- **Note:** NOT supported by PhoneGap Build

Photo by Tama66 (https://pixabay.com/en/users/Tama66-1032521/), courtesy of Pixabay.com

Hook Tips

- **Don't be evil!** Your hook executes on your user's machine!
- `before_prepare` plugin hooks not run on discovery; run the `cordova` command again
- `events.emit("verbose", ...)` and `--verbose` are your friends when troubleshooting

Photo by Tama66 (https://pixabay.com/en/users/Tama66-1032521/), courtesy of Pixabay.com

Homework

- Create a new plugin and add it to a Cordova project.
  - Apple Pencil / Stylus support (pressure, tilt)
  - Audio/video processing
  - Faster computation (compared to JavaScript)
- Extend and/or improve an existing plugin
  - Core plugins should adhere to specs when they are available
  - Translate API docs if you know a language other than English
  - Add and improve tests and examples

Photo by geralt (https://pixabay.com/en/users/geralt-9301/), courtesy of Pixabay.com

Questions?

Thanks!

Jesse (@purplecabbage) • Kerri (@kerrishotts)

Slides available at: https://goo.gl/2xVAwZ

Based on Jesse's PG Day 2016 EU plugin workshop

This slide intentionally left blank