Fantastic Plugins & How to Make Them

Kerri Shotts (@kerrishotts)

Jesse MacFadyen (@purplecabbage)

https://github.com/kerrishotts/pgday/2017/fantastic-plugins-and-how-to-make-them

Based in part on http://purplecabbage.github.io/slides/pgd16Plugins/index.html

About Kerri

- Used PhoneGap for six+ years
- Author of five books about PhoneGap
- Working on several video series about PhoneGap
- IT Consultant for eight years
- Apache Cordova comitter
- @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, and/or improving the developer's workflow

What can plugins do?

- Anything you can do with native code in various contexts:
  - run time
  - build time
  - install time

Plugins at run time

- Expose native device features
  - push notifications, native social network sharing
- Use native UI Widgets
  - Microsoft ACE
- Quality assurance, Logging, etc.
- Analytics
- Faster computations (compared to JS)

Plugins at build time

- Could transpile ES2015+ or Typescript to ES5
- SASS/less pre-processing
- Image inlining / webpack
- Code coverage, linting and quality checks
- Or be used as "proofs of concept" in Cordova dev:
  - cordova-plugin-ios-launch-screen before adding to `cordova-ios@4.3.1`

Plugins at install time

- Could bundle other plugins
- Or, could provide tests for another plugin...

Plugin-ception 🎆

Who has used plugins?

- Everyone
- Cordova device integration provided by **core** plugins
- Two categories
  1. Core
  2. Community

The Core Plugins

Core Cordova features (used to be built-in)

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

Extensions provided by the community — like you!

Repository	Plugins
https://cordova.apache.org/plugins	~1,960 plugins (– core)
http://www.plugreg.com	~1,592 plugins (– core)
http://plugins.telerik.com/cordova	~77 plugins

Managing Plugins

or, finding fantastic 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)

GitHub

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

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

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

Local Filesystem

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

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

- Use `--link` when developing plugins
  - Changes are reflected automatically!
  - No need to `rm` and `add` again.
  - Automatically linked if a parent (`../`)

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
```

```
Package

cordova-plugin-device • https://github.com/apache/cordova
Cordova Device Plugin
updated 2 months ago by shazron
```

Plugin Autopsy 

or, what's inside these things?

ref: cordova-plugin-device

Plugin Structure

```
cordova-plugin-device/          # plugin root
    doc/<locale>                # documentation other than English (convention)
    src/<platform>              # Platform-specific native code
    |   android/
    |   +   Device.java         # Native Android code
    |   ios/
    |   |   CDVDevice.h         # Native iOS header
    |   +   CDVDevice.m         # Native iOS code
    |
    tests/                      # Please add tests!
    types/                      # Types for Typescript
    www/                        # Web assets
    +   device.js               # API for JavaScript consumers
    package.json                # npm metadata
    plugin.xml                  # plugin metadata and configuration
    README.md                   # English documentation
```

(representational only; not every file is included here)

Metadata

All plugins have metadata and settings in `plugin.xml`

- Unique plugin ID for registration, discovery, and management
- Version number, author, repository, etc.
- Supported platforms, engines, OS versions
- Native headers, source files, resources, JavaScript files
- Configuration preferences, permissions
- JavaScript API (if exposed to webview)
- Hooks

Example Metadata (plugin.xml)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<plugin xmlns="http://apache.org/cordova/ns/plugins/1.0"
    xmlns:rim="http://www.blackberry.com/ns/widgets"
    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>
```

JavaScript API Entry

In cordova-plugin-device's plugin.xml:

```
<js-module src="www/device.js" name="device">
    <clobbers target="device" />
</js-module>
```

Indicating Platform Support

In cordova-plugin-device's plugin.xml:

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

Specifying headers, frameworks, etc.

```
<platform name="android">
    <config-file target="res/xml/config.xml" parent="/*">
        <feature name="Device" >
            <param name="android-package" value="org.apache.cordova.device.Device"/>
        </feature>
    </config-file>
    <source-file src="src/android/Device.java" target-dir="src/org/apache/cordova/device" />
</platform>
<platform name="ios">
    <config-file target="config.xml" parent="/*">
        <feature name="Device">
            <param name="ios-package" value="CDVDevice"/>
        </feature>
    </config-file>
    <header-file src="src/ios/CDVDevice.h" />
    <source-file src="src/ios/CDVDevice.m" />
    <framework src="libz.tbd" />
</platform>
```

Note: Can include third-party libraries too. iOS supports Cocoapods too!

npm Metadata Example

Plugins need npm metadata too, so they can be published.

```
{
  "name": "cordova-plugin-device",
  "author": "Apache Software Foundation",
  "license": "Apache-2.0",
  "version": "1.1.5-dev",
  "description": "Cordova Device Plugin",
  "types": "./types/index.d.ts",
  "cordova": {
    "id": "cordova-plugin-device",
    "platforms": ["firefoxos", "tizen", "android", ... ]
  },
  "repository": {
    "type": "git",
    "url": "https://github.com/apache/cordova-plugin-device"
  },
  "keywords": ["cordova", "device", "ecosystem:cordova", ... ],
```

npm Metadata Example (2)

```
  ...
  "scripts": {
    "test": "npm run jshint",
    "jshint": "node node_modules/jshint/bin/jshint www && node node_modules/jshint/bi
  },
  "engines": {
    "cordovaDependencies": {
      "2.0.0": {              // plugin version (applies to any ver 2+)
        "cordova": "> 1.0.0" // cordova-cli above version 1
                              // could be Cordova platform (cordova-ios,...)
                              // or another plugin (eg cordova-plugin-%)
      }
    }
  },
  "devDependencies": {
    "jshint": "^2.6.0"
  }
}
```

Creating and Publishing Plugins

or, the “making” bit of the title...

💶 And getting rich, maybe? 💶

Or maybe not...

Who has used "plugman"?

- Everyone! You just might not know it.
- `plugman` is a `node` library that manages plugins in your projects
- `cordova-cli`, `phonegap-cli`, etc., use `plugman` internally

```
$ npm install -g plugman

$ plugman install --plugin cordova-plugin-device \
    --platform ios --project .
```

"plugman" can create plugins, too!

```
$ plugman create --name PluginName \
                 --plugin_id cordova-plugin-plugin-name \
                 --plugin_version 0.0.1 \
                 --path .

# or, if you have created a repo already:
$ plugman create --name plugin-directory \
                 --plugin_id cordova-plugin-plugin-name
                 --plugin_version 0.0.1
                 --path .. # <-- parent directory! ;-)
```

- Can pass `--variable-name=value` pair string to define additional data like author, etc.

Dependencies

- Plugins can depend upon other plugins (and platforms, too)

```
<!-- plugin.xml -->
<dependency id="cordova-plugin-device" />
<dependency id="cordova-plugin-console" version="^1.0.0" />
// or in package.json
"engines": {
    "cordovaDependencies": {
        "2.0.0": { //plugin version (applies to any ver 2+)
            "cordova-plugin-console": "> 1.0.0"
        }
    }
}
```

Wiring it all up...

```
// in plugin.js
cordova.exec(successFn, failureFn, "PluginName",
             "pluginMethod", args<Array>);
<!-- in plugin.xml -->
<feature name="PluginName">
    <param name="ios-package" value="CDVPluginClass" />
    <param name="onload" value="true" />
</feature>
// in CDVPluginClass.m
- (void) pluginMethod:(CDVInvokedUrlCommand*)command {
    // do something useful and optionally
    // return results across the "bridge"
}
```

StatusBar Example

```
// in statusbar.js
cordova.exec(null, null, "StatusBar", "styleDefault", []);
<!-- in plugin.xml -->
<feature name="StatusBar">
    <param name="ios-package" value="CDVStatusBar" />
    <param name="onload" value="true" />
</feature>
// in CDVStatusBar.m
- (void) styleDefault:(CDVInvokedUrlCommand*)command {
    [self setStyleForStatusBar:UIStatusBarStyleDefault];
}
```

Returning data back to JavaScript

```
// in CDVStatusBar.m
(void)fireTappedEvent {
    if (_eventsCallbackId == nil) { return; }

    NSDictionary* payload = @{@"type": @"tap"};

    CDVPluginResult* result = [CDVPluginResult
        resultWithStatus:CDVCommandStatus_OK
        messageAsDictionary:payload];

    [result setKeepCallbackAsBool:YES]; // default is NO

    [self.commandDelegate sendPluginResult:result
        callbackId:_eventsCallbackId];
}
```

Follow the yellow brick bridge?

or, a look at the code behind the curtain!

Lots of bridges

- iOS
- Android
- Windows
  - Careful, the bridge is a **mirage**! 🏜️
  - JavaScript is **native** 🌟
  - `cordova.exec` uses a proxy

Publishing your plugin

- `npm` is the home of all core Cordova plugins
- If you want to publish to `npm`, you'll need a `package.json`
- `plugman` can do that for you too!

```
$ plugman createpackagejson .

$ npm publish
```

A cool plugin demo

Testing your plugins

or, the art of making sure it works like it should

and improving the lives of developers who use your plugin 😄

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

```
$ cordova-paramedic --platform ios --plugin .
```

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

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 `plugin.xml` file (doesn't need to be complex)

Debugging

or, mastering the dark art of reading your computer's mind

Debugging

- Debugging native code in Xcode
- Debugging native code in Android Studio
- Debugging Windows JS Code in Visual Studio
  - (note to self: start your VM!)

Docs

- You should include documentation so that users know how to use your plugin
- Look at any of the "core" plugins for best practices
- Convention:
  - English docs in the root `README.md` file
  - Translations in the `docs/` folder

Hooks

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

Possibilities:

- Create entitlements as needed
- Code transformations
- Create launch images and icons
- Check plugin versions and warn if out-of-date

Some more cool plugin ideas

- Optical Character Recognition using Tesseract
- Game controller support
- Apple Pencil, anyone?
- iOS Storage providers
- Audio/video processing

Tips & Tricks

or, wisdom from those who have gone before

and face-palmed for you in your stead...

JS API

- Promisify your API
- Preprocess arguments in JavaScript
  - convert to appropriate types
  - throw type-mistmatch errors, etc.
- Transpile ES2015+ to ES5 if you want to use ES2015
  - not all targets understand native ES2015 yet

Native

- Return useful error information
- Use background threads!
- Be respectful of other plugins
- Lazy load?
- Init events?

Homework

- Extend and/or improve a plugin
- For example, the globalization plugin's API is asynchronous, which is really irritating.
  - All the formatting / globalization information could be determined up-front, and then the API could be synchronous!
  - https://github.com/apache/cordova-plugin-globalization
- The sky's the limit!

Questions?

Thanks!

Jesse (@purplecabbage)

Kerri (@kerrishotts)

https://github.com/kerrishotts/pgday/2017/fantastic-plugins-and-how-to-make-them

Based in part on http://purplecabbage.github.io/slides/pgd16Plugins/index.html