

# Creating a Modern PhoneGap Plugin

Kerri Shotts (@kerrishotts)

Jesse MacFadyen (@purplecabbage)

Slides at <https://kerrishotts.github.io/pgday>

Based on PGDay EU 2016 plugin workshop by Jesse

# About Kerri

---

- Used PhoneGap for six+ years
- Author of five books about PhoneGap
- Apache Cordova committer
- One of several moderators:
  - [Adobe PhoneGap Forums](#)
  - [Google Cordova Group](#)
- At Adobe now for nearly 2 months
- [@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
  - 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)

# 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.

# 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!*

# The Core Plugins

---

battery-status	Keep	
camera	Keep	
contacts	Sunset	
console	Merge	Remember to remove it!
device	Keep	
device-motion	Sunset	Migration
device-orientation	Sunset	Migration
dialogs	Keep	
file	Keep	
file-transfer	Sunset	XHR2

# The Core Plugins

geolocation	Keep	
globalization	Keep	
inappbrowser	Keep	
media	Sunset	Browser APIs
media-capture	Keep	
network-information	Keep	
splashscreen	Merge	
statusbar	Merge	
vibration	Keep	
whitelist	Keep	

# Community Plugins

Developed and supported by the community – like you!

<https://cordova.apache.org/plugins>

~2,720 plugins & templates (excl. core)

# Managing Plugins

Plugins are typically downloaded from npm:

```
cordova plugin add cordova-plugin-device
```

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

```
cordova plugin rm cordova-plugin-device # or remove
```

**Note:** As of Cordova 7.x, `--save` is implied, so plugins automatically get saved to your project configuration. Use `--nosave` to disable if needed.

**Important:** Fetching via npm is now the default as of Cordova 7.x; if a plugin doesn't have package.json adding will fail with `--nofetch` for those plugins.

# Git

---

Plugins can also be installed from a Git repository. Typically used for pre-release testing.

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

Specify a branch:

```
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 /path/to/plugin  
cordova plugin rm cordova-plugin-device
```

# 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-plugin-device>  
Cordova Device Plugin  
updated 2 months ago by shazron

# Plugin X-ray

# The Stuff Plugins are Made of

---

Metadata	Documentation +
Native Code *	JavaScript *
Tests +	Hooks *
Typings *	TLC * +

---

\* Optional

+ Optional but highly suggested

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

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

# 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.

# 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: **bold is required**; otherwise optional, but most are used  
Note: package.json can be generated by plugman

# Example Metadata (plugin.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)

```
{ "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": [ "android", "ios", "windows", "wp8", ... ] },
  "repository": { "type": "git", "url": "https://..." },
  "keywords": [ "cordova", "device", "ecosystem:cordova",
    "cordova-ios", "cordova-android", ... ] }
```

# JavaScript Modules

---

Automatically injected into your consumer's `index.html`. [docs](#)

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

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

# Platform Support

---

Use <platform> tags: [docs](#)

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

---

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

# Assets and Native Code

```
<platform name="android">
  <source-file src="src/android/Device.java"
    target-dir="src/org/apache/cordova/device" />
</platform>
<platform name="ios">
  <header-file src="src/ios/CDVDevice.h" />
  <source-file src="src/ios/CDVDevice.m" />
  <framework src="libz.tbd" />
</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<sup>1</sup> [docs](#)
  - Adds elements to manifests / plist or platform config.xml

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

```
<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`<sup>1</sup> 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

---

Plugin Dependencies are managed in plugin.xml: [docs](#)

```
<dependency id="cordova-plugin-camera" version="~2.0.0" />
<dependency id="cordova-plugin-statusbar" version="^2.0.0" />
<dependency id="cordova-plugin-device"
  url="https://github.com/apache/cordova-plugin-device.git" />
```

# 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
```

---

Don't forget .npmignore!

If you used the PhoneGap Plugin Template, package.json is already there – you'll need to update it.

A photograph of a small, rustic wooden garden bridge. The bridge has two arched wooden railings and a central walkway made of wooden planks. It is situated in a lush, green garden with various plants and rocks in the background.

# Crossing the bridges

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

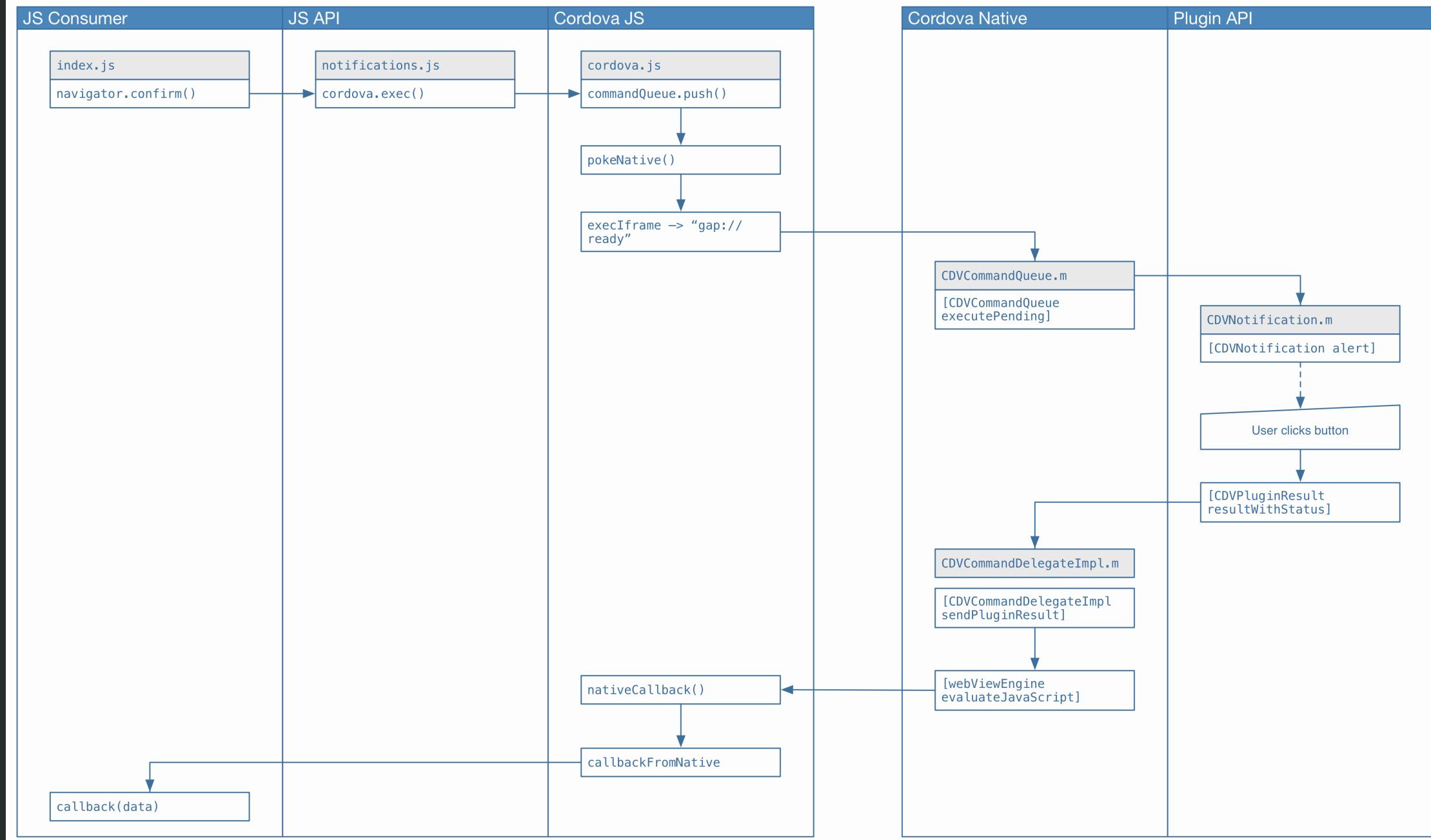
# Know your Bridges

---

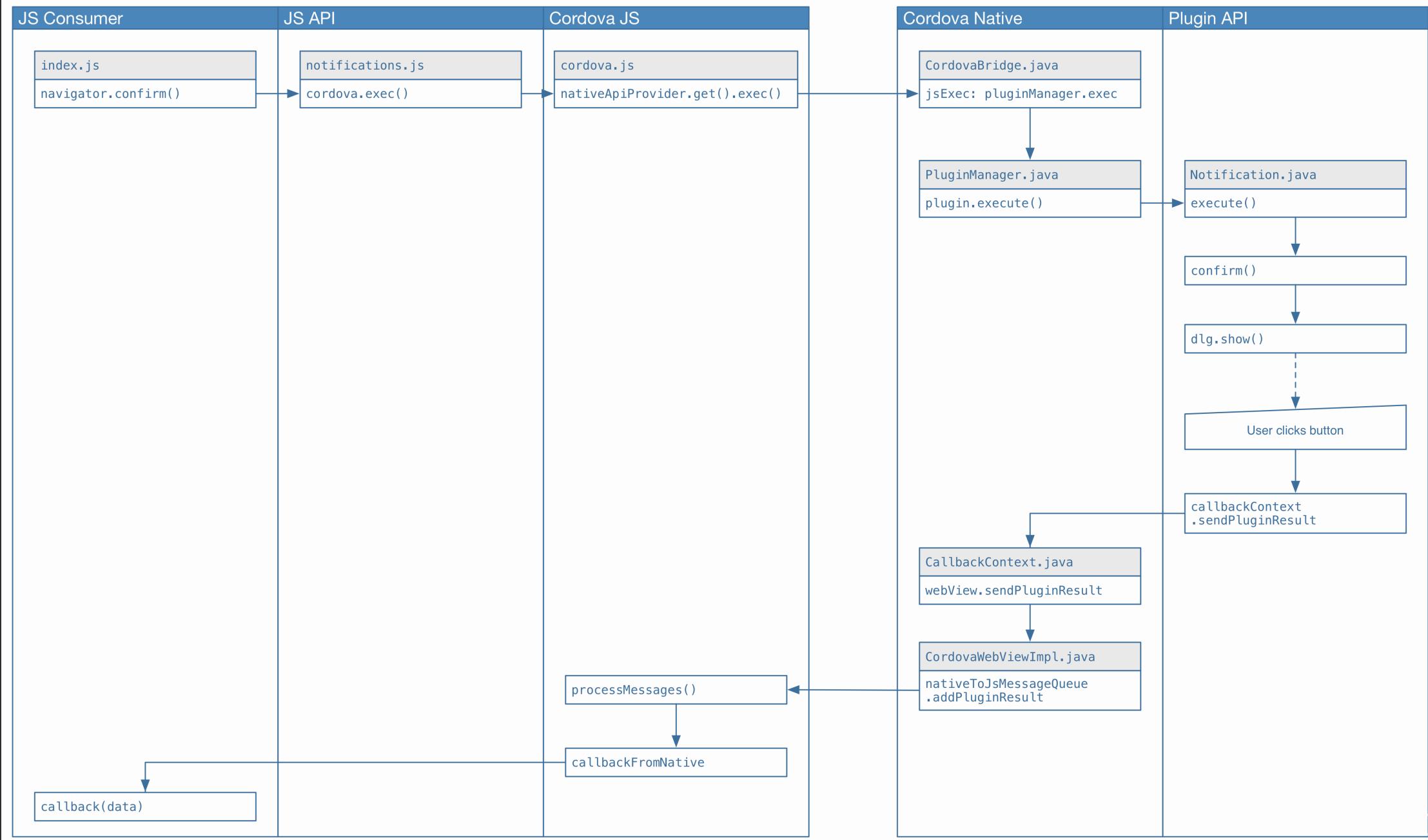
Allows communication between native code and web view contexts.

- 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](#)

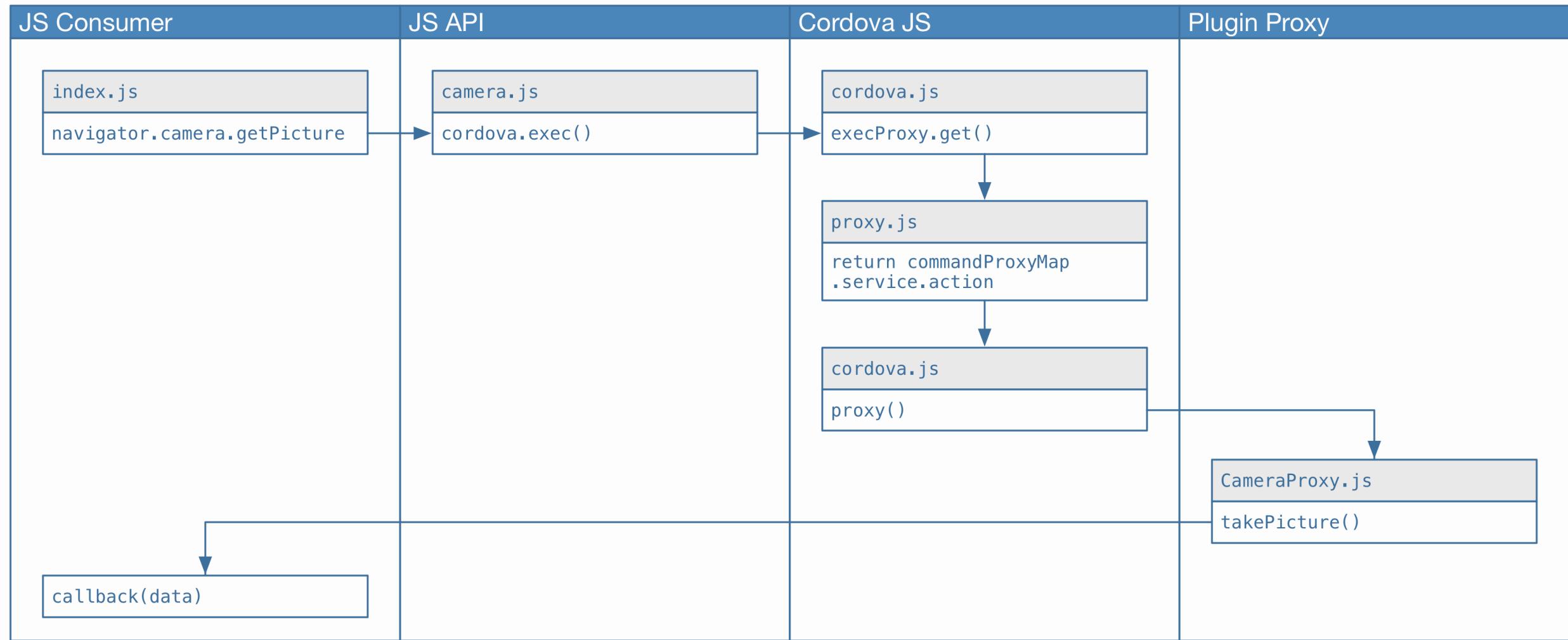
## Cordova iOS Bridge (abridged)



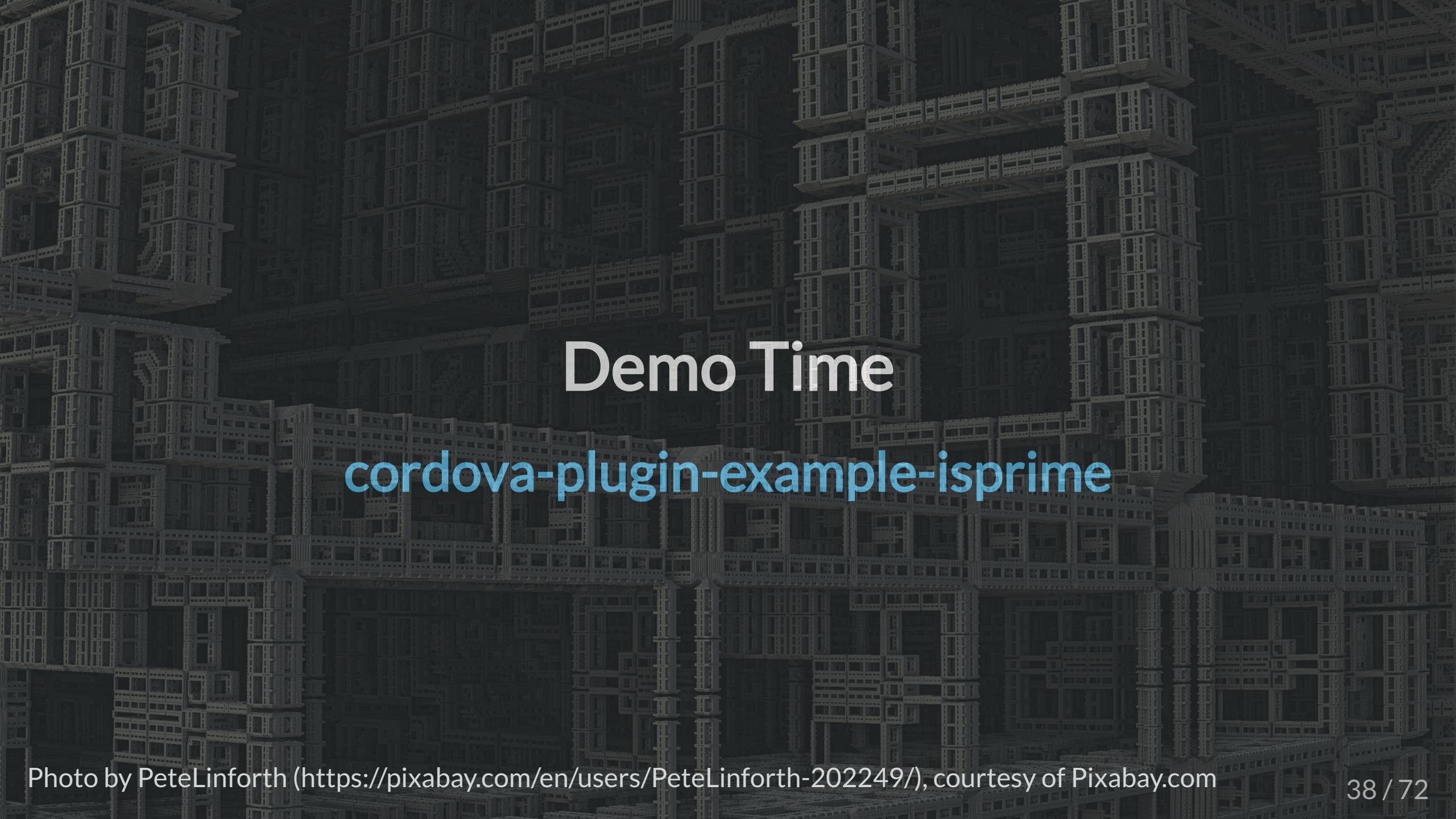
## Cordova Android Bridge (abridged)



## Cordova Windows / Browser “Bridge” (abridged)



# Creating Plugins



# Demo Time

## cordova-plugin-example-isprime

# 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`

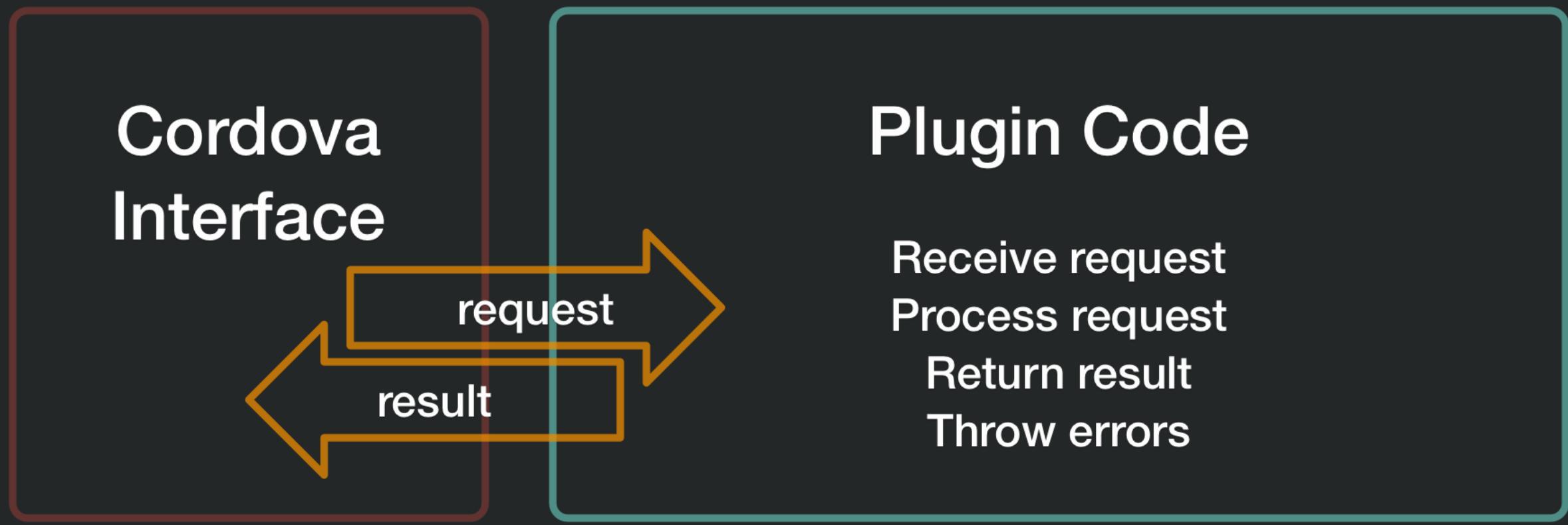
`result`

# 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



# 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);
```

```
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
```

```
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

# Tests

Cordova Test  
Harness

cordova-paramedic  
cordova-plugin-test-  
framework

Test Cases

Your Jasmine tests  
Automatic & Manual

# 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 asynchronously
- 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  
# or  
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

# 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 programmatically
- For native UI, you can use Appium
- [Travis CI example](#) (iOS & Android only)

A close-up photograph of a fly's head and compound eyes. The eyes are large, faceted, and dark, while the surrounding head area is covered in small, light-colored hairs. The fly's mouthparts and antennae are also visible.

# 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 /path/to/plugin
```

- Open your preferred IDE
- Build & run your app

# IDEs & Debugging Demo

- 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
```

# Workflow

---

- Run your project
- If changes need to be made, make them locally
- Once things work, be sure to copy changes back to the original plugin!



# Tips & Tricks

# JavaScript API

---

- Promisify your API <sup>eg</sup>
- 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 `onRequestPermissionsResult` [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`) without wrapping with `setTimeout`

# Miscellaneous

---

- Don't forget the Browser platform! It's real, and useful!
- Don't forget Windows, either
  - JavaScript is a first-class citizen, which makes things even easier.
- Plugins don't just have to be about improving computational performance!
  - You can do pretty cool things by integrating with the native SDK
  - <https://github.com/purplecabbage/phonegap-plugin-multiview>

# 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

# 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

# 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

# Questions?

## Thanks!

Jesse (@purplecabbage) • Kerri (@kerrishotts)

Slides at <https://kerrishotts.github.io/pgday>

Based on Jesse's PG Day 2016 EU plugin workshop

This slide intentionally left blank