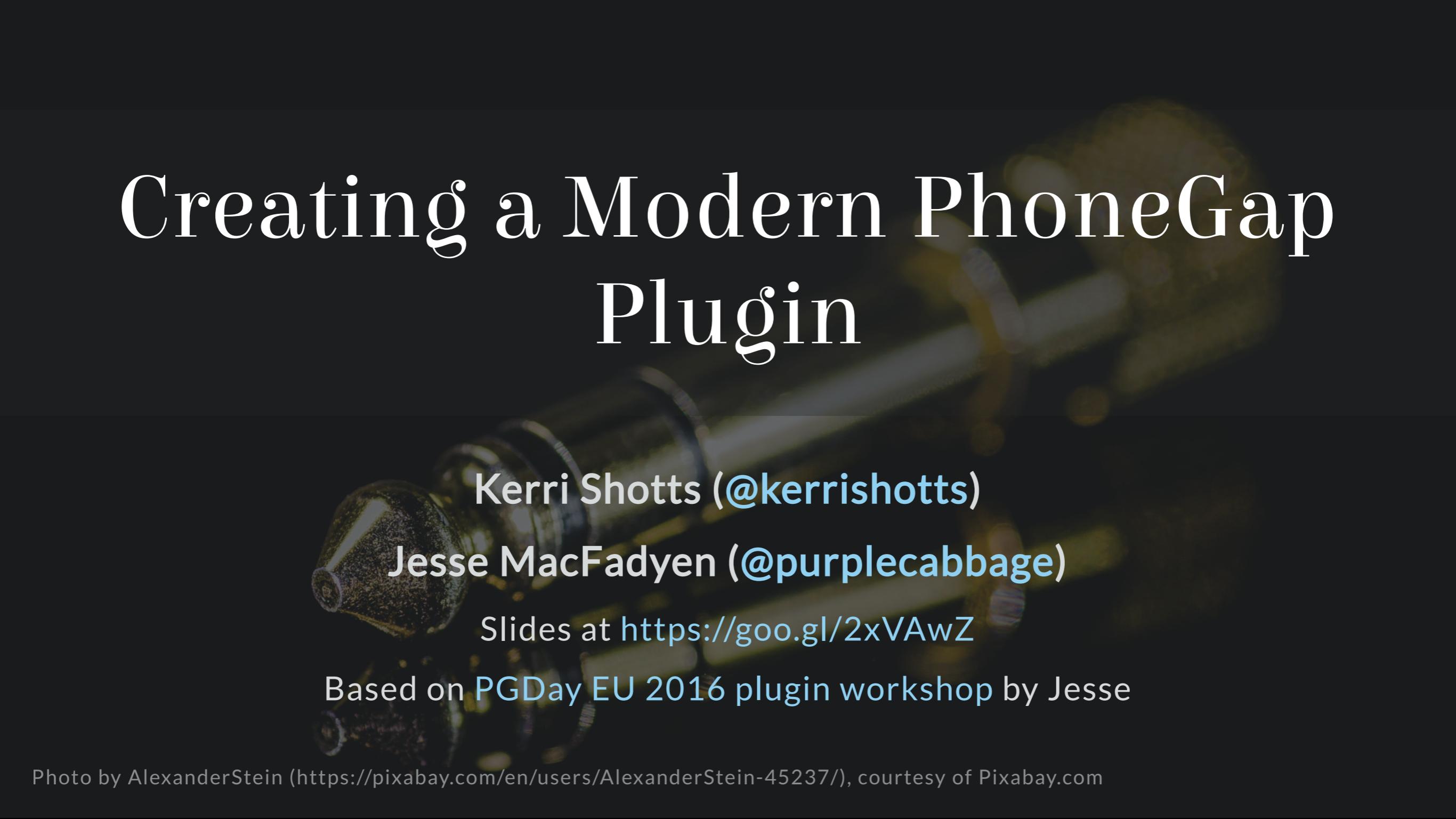


# Creating a Modern PhoneGap Plugin



Kerri Shotts (@kerrishotts)

Jesse MacFadyen (@purplecabbage)

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

Based on PGDay EU 2016 plugin workshop by Jesse

# 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

# 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

---

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	<del>splashscreen</del>	statusbar
vibration	whitelist	

# Community Plugins

---

Developed and supported by the community – like you!

Repository	Plugins
<a href="https://cordova.apache.org/plugins">https://cordova.apache.org/plugins</a>	~2,211 plugins & templates (excl. core)
<a href="http://www.plugreg.com">http://www.plugreg.com</a>	~1,592 plugins (excl. core)
<a href="http://plugins.telerik.com/cordova">http://plugins.telerik.com/cordova</a>	~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
```

---

Package

---

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

---

# Plugin X-ray

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

# The Stuff Plugins are Made of

---

Ingredients	Ingredients
Metadata	Documentation <sup>s</sup>
Native Code *	JavaScript *
Tests <sup>s</sup>	Hooks *
Typings *	TLC * <sup>s</sup>

\* Optional

<sup>s</sup> Optional but highly suggested

Plugin Structure	Description
<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 <i>Please add tests!</i>
<code>tests/</code>	
<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>CDV YourPlugin.h</code>	
<code>CDV YourPlugin.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

# 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)

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <plugin xmlns="http://apache.org/cordova/ns/plugins/1.0"
3   xmlns:android="http://schemas.android.com/apk/res/android"
4   id="cordova-plugin-device" version="1.1.5-dev">
5   <name>Device</name>
6   <description>Cordova Device Plugin</description>
7   <license>Apache 2.0</license>
8   <keywords>cordova,device</keywords>
9   <repo>https://git-wip-us.apache.org/repos/asf/
10    cordova-plugin-device.git</repo>
11   <issue>https://issues.apache.org/jira/browse/CB/
12    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:** Visible platform support on plugin repo is separately controlled ( package.json keywords)

# 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>
```

- Use `<param name="onload" value="true" />` to init at startup

# Manifest Modifications

---

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

```
1 <config-file target="AndroidManifest.xml" parent="/*>
2   <uses-permission android:name=
3     "android.permission.WRITE_EXTERNAL_STORAGE" />
4 </config-file>
1 <config-file target="*-Info.plist"
2   parent="NSLocationWhenInUseUsageDescription">
3   <string>$GEOLOCATION_USAGE_DESCRIPTION</string>
4 </config-file>
```

---

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

# Manifest Modifications (2)

---

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

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

# Dependencies (< cordova@6.1.0)

---

Before cordova@6.1.0, plugin.xml managed dependencies:

- plugin dependencies<sup>1</sup> docs

```
<dependency id="cordova-plugin-file" version="^4.0.0" />
```

- platform & tool dependencies<sup>2</sup> docs

```
<engines>
  <engine name="cordova" version=">=3.1.0" />
</engines>
```

---

**Note:** don't forget about XML entities! So "<" becomes "&lt;"

# Dependencies (cordova@6.1.0+)

---

Now dependencies should be 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 critical; how else will you users know how to use your plugin?

- 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

# 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  
  
$ phonegap-plugin-create isprime IsPrime  
$ cordova-plugin-example-isprime #parms: path name plugin-id  
? 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`

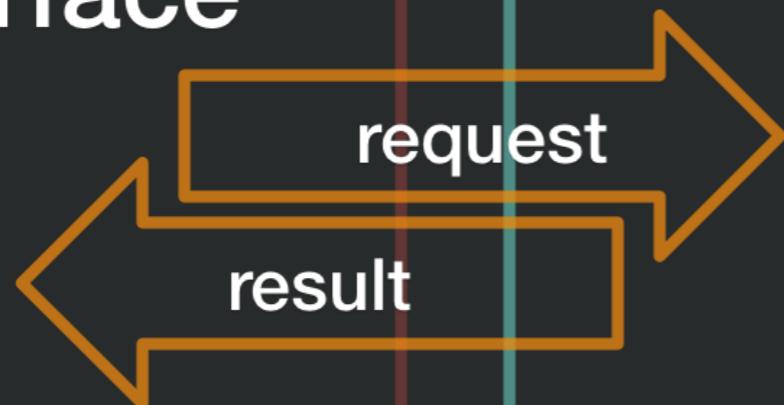
# Your Plugin's JS API

---

```
// www/isPrime.js
var exec = cordova.require("cordova/exec"), SERVICE = "IsPrime";
function tick(fn, thisArg) {
    return function() {
        setTimeout(fn.apply(thisArg, arguments), 0);
    };
}
module.exports = function isPrime(successFn, failureFn, candidate) {
    // ensure the parameters are of the correct types
    if (typeof successFn !== "function") throw new Error("...");
    /*...*/
    var result = { isPrime: false, candidate: candidate, /*...*/ };
    exec(tick(successFn), tick(failureFn), SERVICE, "isPrime", [result]);
}
```

# Native Code

Cordova  
Interface



Plugin Code

Receive request  
Process request  
Return result  
Throw errors

# Your Native Code (iOS)

---

```
#import <Cordova/CDV.h>
@interface CDVIIsPrime : CDVPlugin
@end
@implementation CDVIIsPrime
- (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 {
        /* abracadabra: determine if candidate is prime */
        PluginResult pluginResult=new PluginResult(PluginResult.Status.OK, result);
        callbackContext.sendPluginResult(pluginResult);
    }
}
```

# Your Native Code (Browser / Win)

---

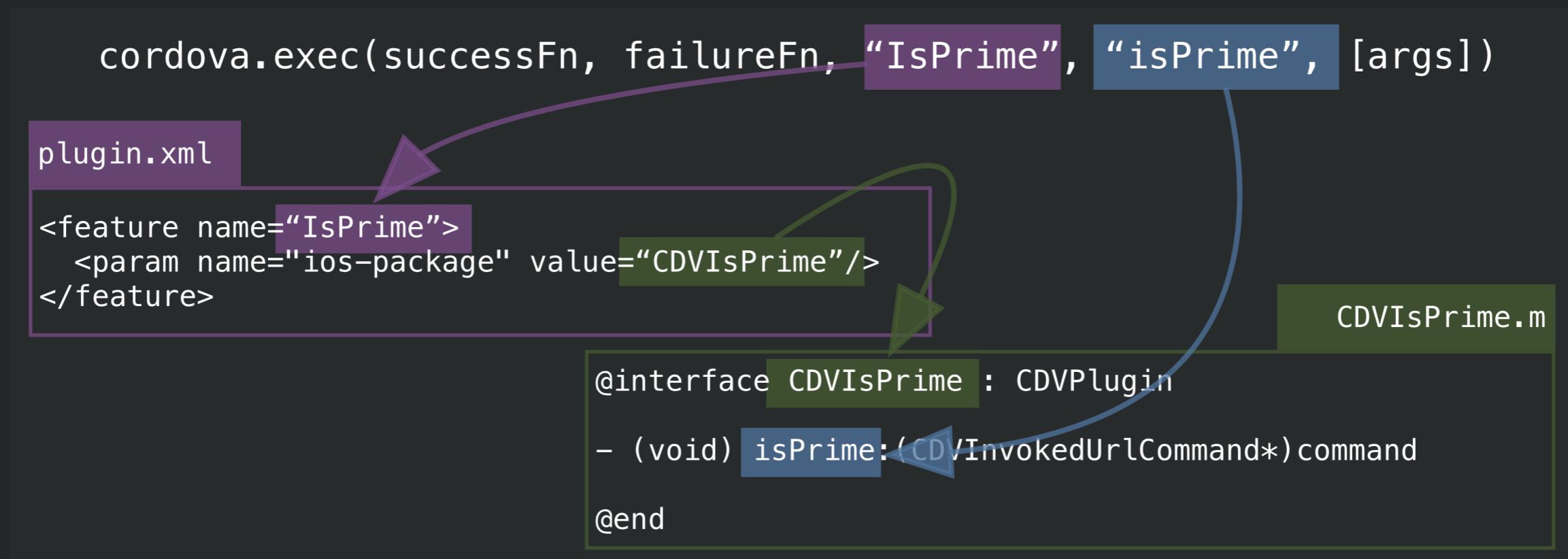
```
//src/[browser|windows]/yourPluginProxy.js
function isPrime(successFn, failureFn, args) {
    var result = args[0],
        candidate = result.candidate;
    /* magic! calculate if candidate is prime */
    successFn(result);
}

module.exports = { isPrime: isPrime };

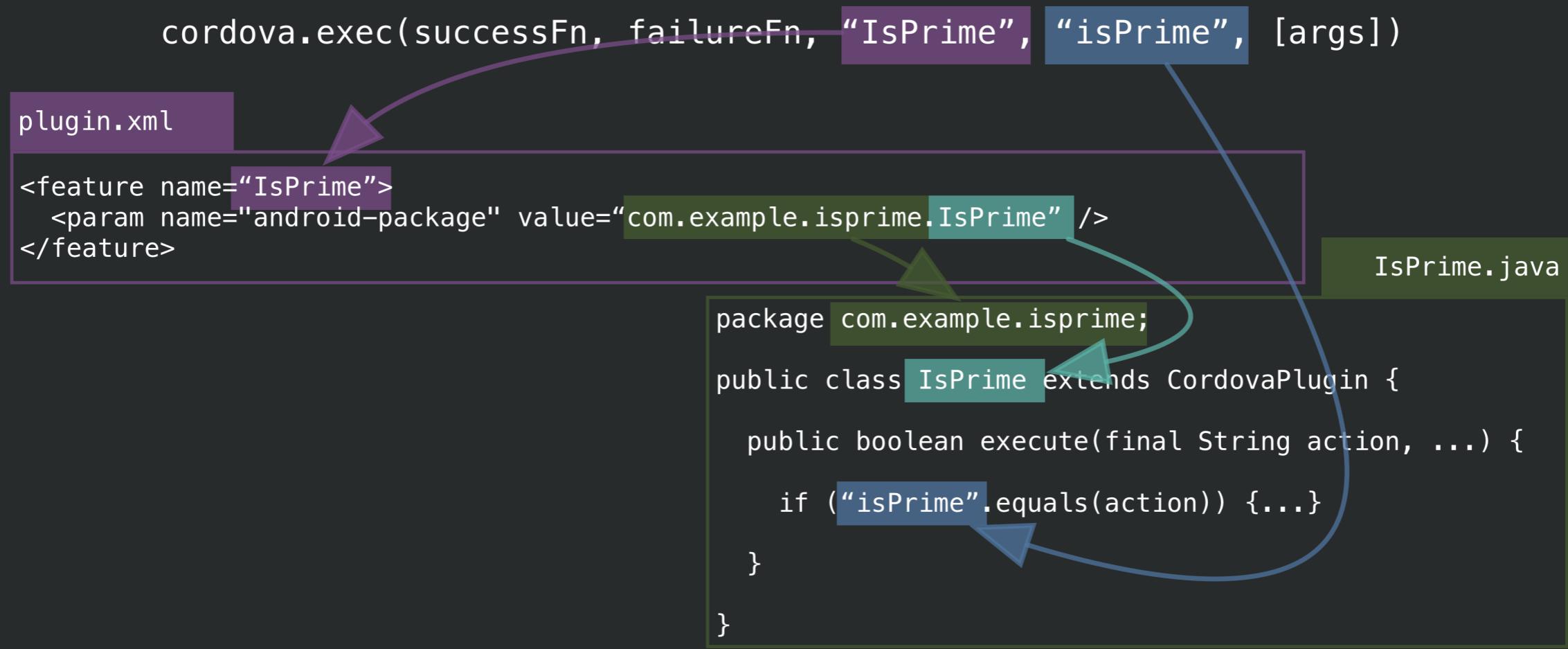
require("cordova/exec/proxy").add("IsPrime", module.exports);
```

# Plugin Class Mapping (iOS)

Remember the JS API's call to `cordova.exec` ?



# Plugin Class Mapping (Android)



# 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});
```

---

iOS StatusBar example

# 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

A photograph of a small, rustic wooden bridge with a curved arch and a flat walkway, situated over a shallow stream or pond in a lush green garden. In the background, there's a colorful wooden playhouse and some large rocks. The foreground is filled with green grass and foliage.

# Crossing the bridges

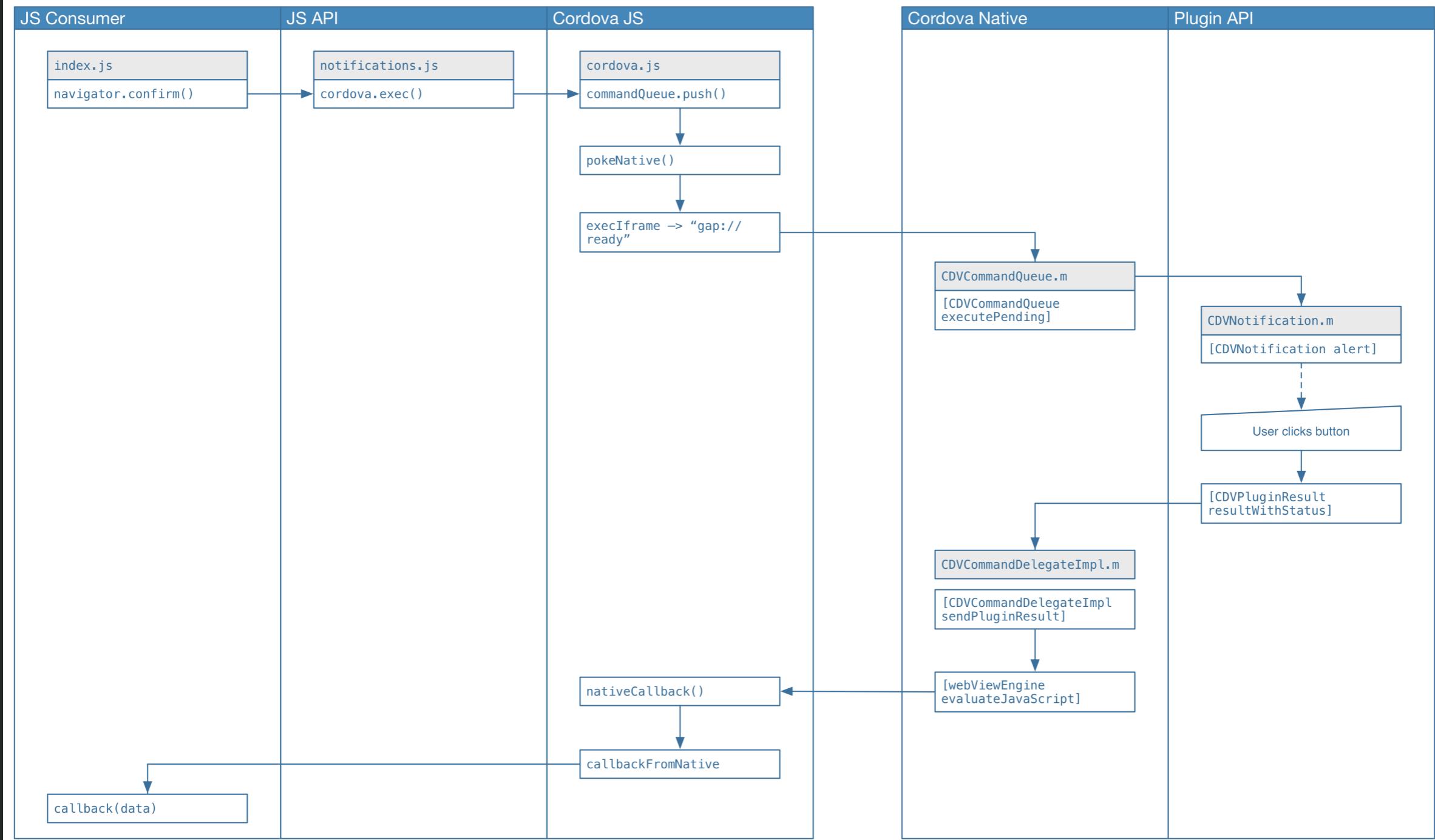
# Know your Bridges

---

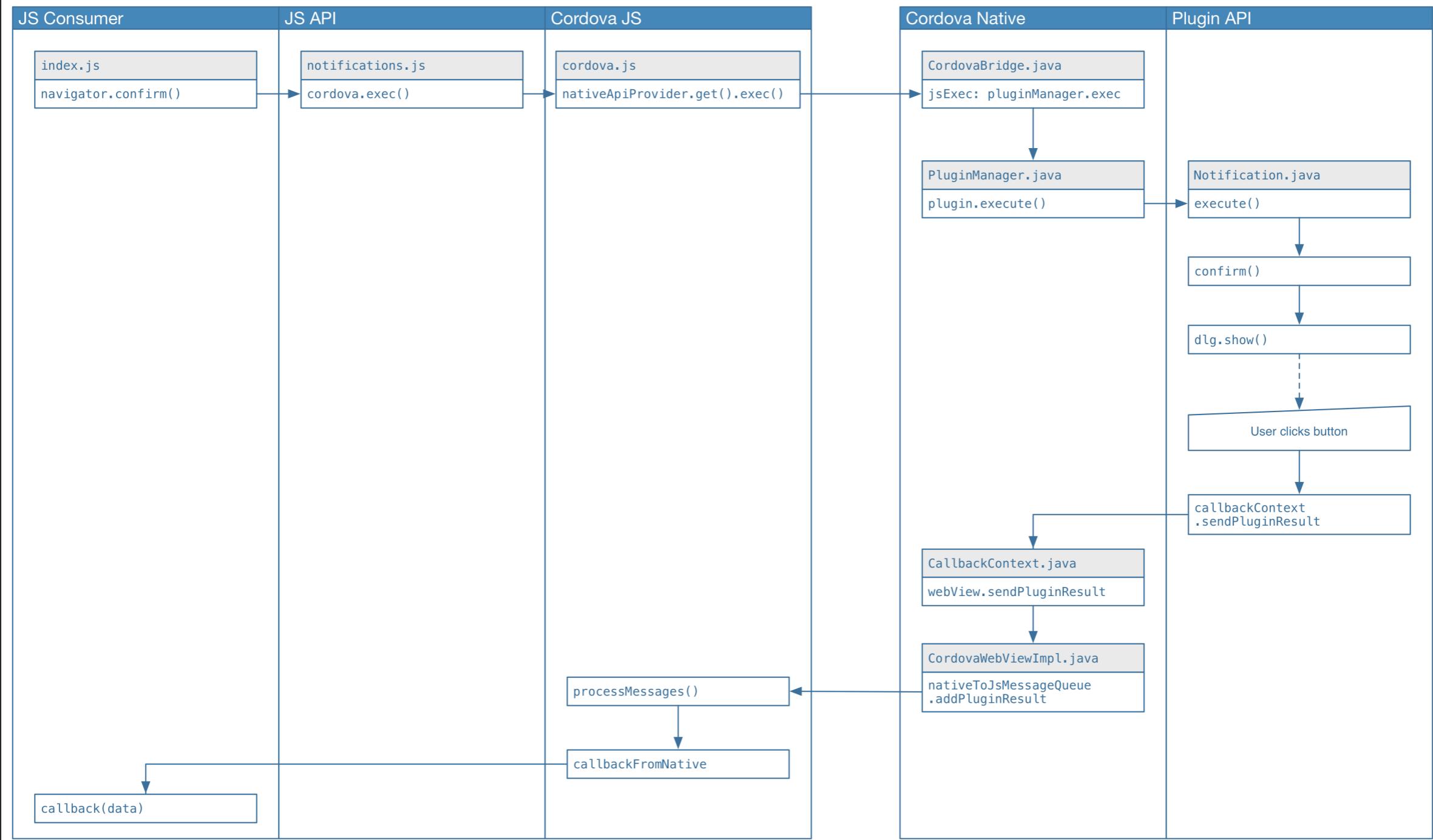
Allows communication between native code and web view contexts.

- 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**

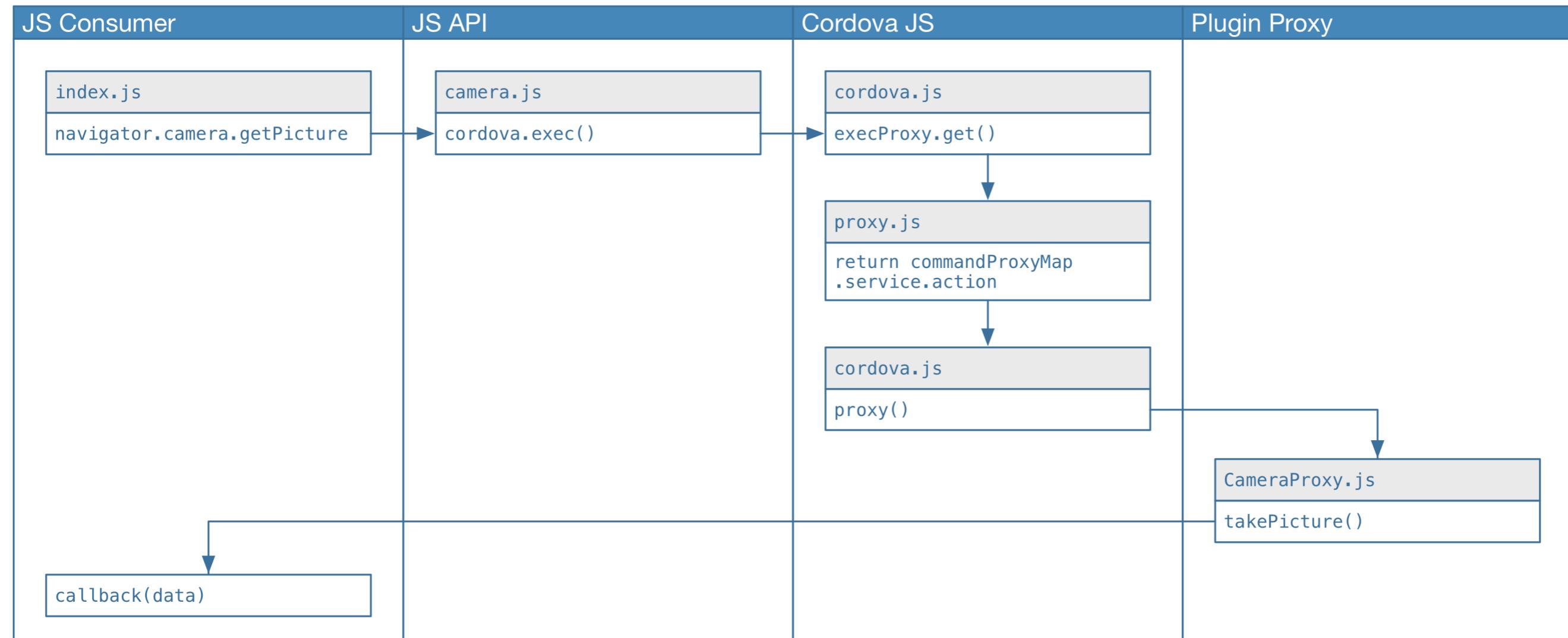
## Cordova iOS Bridge (abridged)



## 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

# 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 install -g cordova-paramedic  
$ | cordova-paramedic --platform ios --plugin .
```

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

# 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 programmatically

A close-up photograph of a beetle's head and thorax. The beetle has a metallic green and blue coloration on its elytra (wing covers). Its antennae are long and segmented, and it has large, compound eyes. The texture of its exoskeleton, with numerous small pits and ridges, is clearly 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

```
1 | cordova create hello com.example.hello hello  
2 | cd hello  
3 | cordova platform add ios android browser  
4 | cordova plugin add --link /path/to/plugin
```

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

- Xcode (macOS) / Safari • Android Studio / Google Chrome • Visual Studio (Windows)

# What gets --linked?

---

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

Exceptions & notes:

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

# Publishing your plugin

---

- If you want to publish to npm , you'll need a package.json
- plugman can generate it based on plugin.xml for you:  
\$ | plugman createpackagejson .
- If you used the PhoneGap Plugin Template, package.json is already there – you'll need to update it.
- Once package.json is correct, publish via:  
\$ | npm publish
- Don't forget .npmignore !

A photograph of a person's hand holding a magnifying glass over a small, round terrarium. The terrarium contains a small green plant growing in soil. The background is dark and out of focus.

# Tips & Tricks

# 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  
`<param name="onload" value="false" />`
- 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`)

# 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

# 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
- 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 instead
    - Try it: <https://github.com/apache/cordova-plugin-globalization>
- The sky's the limit!

# Some more cool plugin ideas

---

- Game controller support
- Apple Pencil / Stylus support (pressure, tilt)
- Audio/video processing
- Faster computation (compared with JavaScript)

# 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