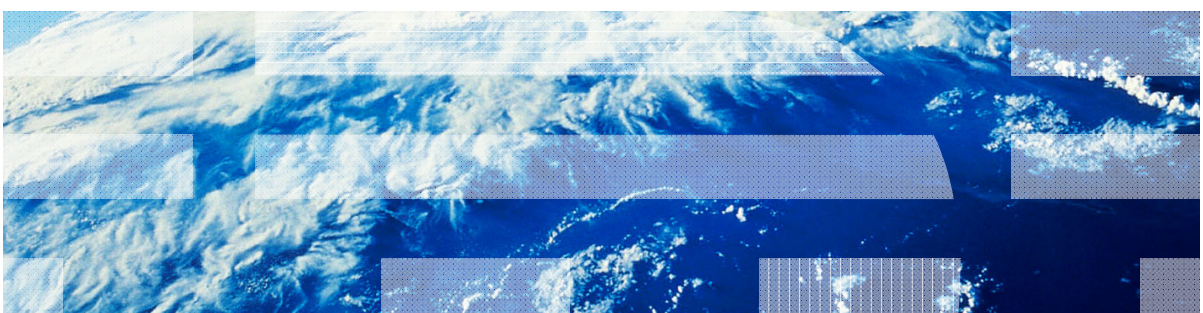
IBM Worklight V5.0.5
Getting Started

Module 8.3 – iOS Development Using the Apache Cordova Plug-in



18 January 2013

Trademarks

- IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Worklight is a trademark or registered trademark of Worklight, an IBM Company. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.
- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
- Other company products or service names may be trademarks or service marks of others.
- This document may not be reproduced in whole or in part without the prior written permission of IBM.

About IBM®

- See http://www.ibm.com/ibm/us/en/

Agenda

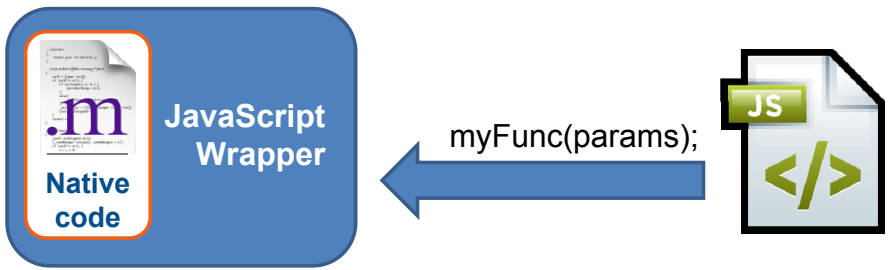
- Apache Cordova plug-in overview
- Implementing an Objective-C code plug-in
- Adding a plug-in to DOM
- Invoking a plug-in from JavaScript

Apache Cordova plug-in overview

- Occasionally within a Worklight® application, developers need to use a specific third-party native library or a device function that is not yet available in Apache Cordova.
- Apache Cordova allows developers to create custom native code blocks and invoke them using JavaScript.
- This technique is called an Apache Cordova plug-in.
- This module demonstrates how to create a simple Apache Cordova plug-in and how to use it in your code.
- More samples can be found in the Apache Cordova documentation at https://github.com/phonegap/phonegap-plugins.

Apache Cordova plug-in overview

- An iOS Apache Cordova plug-in consists of two parts:
  - An Objective-C code which runs natively in iOS.
  - A JavaScript wrapper .
- When both parts are implemented, you can call native code from JavaScript in a simple and familiar way.

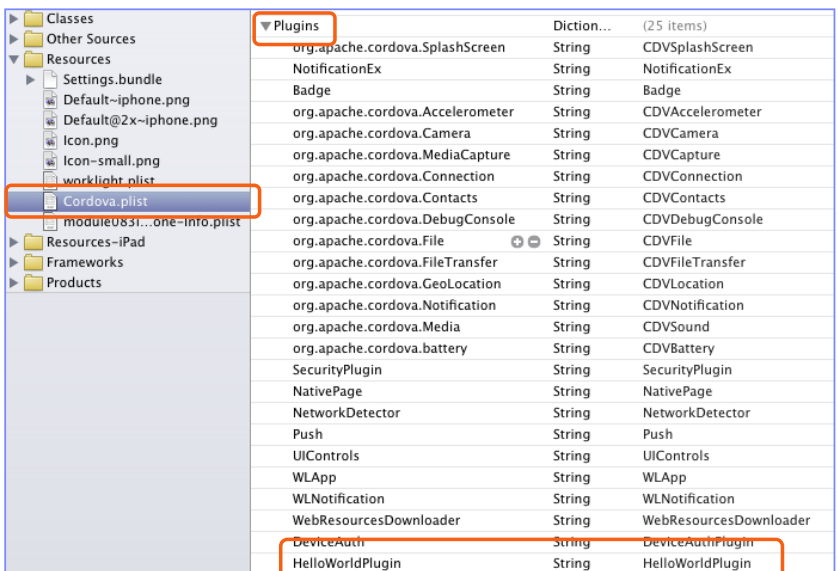


Agenda

- Apache Cordova plug-in overview
- Implementing an Objective-C code plug-in
- Adding plug-in to DOM
- Invoking plug-in from JavaScript

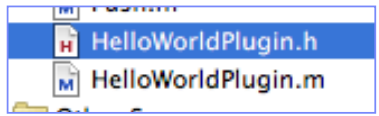
Implementing an Objective-C Code plug-in

- Add your plug-in definition to the Plugins section of the `Cordova.plist` file.
- Use a custom name for the key and the class name of the plug-in for the value.



Implementing an Objective-C Code plug-in

- Start by creating an Objective-C class for a plug-in. Call it HelloWorldPlugin.
- Import the `CDVPlugin.h` and inherit the CDVPlugin class. Add a property for a `callbackId` and required the custom methods.




```
#import <Foundation/Foundation.h>
#import <Cordova/CDVPlugin.h>

@interface HelloWorldPlugin : CDVPlugin{
    NSString *callbackId;
}
-(void)sayHello:(NSMutableArray*)arguments withDict:(NSMutableDictionary*)options;

@property (nonatomic, copy) NSString* callbackId;

@end
```

Implementing an Objective-C Code plug-in

- Implement the method:

```
#import "HelloWorldPlugin.h"

@implementation HelloWorldPlugin

@synthesize callbackId;

-(void)sayHello:(NSMutableArray*)arguments withDict:(NSMutableDictionary*)options{

    self.callbackId = [arguments pop];

    NSString *responseString = [NSString stringWithFormat:@"Hello World, %@", [arguments
        objectAtIndex:0]];

    CDVPluginResult *pluginResult = [CDVPluginResult resultWithStatus:CDVCommandStatus_OK
        messageAsString:responseString];

    [self writeJavascript:[pluginResult toSuccessCallbackString:self.callbackId]];
}

@end
```

Implementing an Objective-C Code plug-in

- Implement the method:

```
#import "HelloWorldPlugin.h"

@implementation HelloWorldPlugin

@synthesize callbackId;

-(void)sayHello:(NSMutableArray*)arguments withDict:(NSMutableDictionary*)options{

    self.callbackId = [arguments pop];

    NSString *responseString = [NSString stringWithFormat:@"Hello World, %@",[arguments
        objectAtIndex:0]];

    CDVPluginResult *pluginResult = [CDVPluginResult resultWithStatus:CDVCommandStatus_OK
        messageAsString:responseString];

    [self writeJavascript:[pluginResult toS
}

@end
```

Function arguments are received as a standard NSMutableArray object.

Implementing an Objective-C Code plug-in

- Implement the method:

```
#import "HelloWorldPlugin.h"

@implementation HelloWorldPlugin

@synthesize callbackId;

-(void)sayHello:(NSMutableArray*)arguments withDict:(NSMutableDictionary*)options{

    self.callbackId = [arguments pop];

    NSString *responseString = [NSString stringWithFormat:@"Hello World, %@", [arguments
        objectAtIndex:0]];

    CDVPluginResult *pluginResult = [CDVPluginResult resultWithStatus:CDV
        messageAsString:responseString];

    [self writeJavascript:[pluginResult toS
}

@end
```

The first argument contains a reference to the success and failure callbacks.
Following arguments are custom.

Implementing an Objective-C Code plug-in

- Implement the method:

```
#import "HelloWorldPlugin.h"

@implementation HelloWorldPlugin

@synthesize callbackId;

-(void)sayHello:(NSMutableArray*)argume

    self.callbackId = [arguments pop];

    NSString *responseString = [NSString stringWithFormat:@"Hello World, %@ arguments
        objectAtIndex:0]];

    CDVPluginResult *pluginResult = [CDVPluginResult resultWithStatus:CDVCommandStatus_OK
        messageAsString:responseString];

    [self writeJavascript:[pluginResult toSuccessCallbackString:self.callbackId]];
}

@end
```

The pluginResult object is created and populated with a response message (can be string, dictionary, array and so on.)

Implementing an Objective-C Code plug-in

- Implement the method:

```
#import "HelloWorldPlugin.h"

@implementation HelloWorldPlugin

@synthesize callbackId;

-(void)sayHello:(NSMutableArray*)arguments wi

    self.callbackId = [arguments pop];

    NSString *responseString = [NSString stringWithFormat:@"Hello World, %@", [arguments
        objectAtIndex:0]];

    CDVPluginResult *pluginResult = [CDVPluginResult resultWithStatus: ommandStatus_OK
        messageAsString:responseString];

    [self writeJavascript:[pluginResult toSuccessCallbackString:self.callbackId]];
}

@end
```

The writeJavascript method is used to return a response back to JavaScript. The pluginResult object has the toSuccessCallbackString and the toFailureCallbackString methods.

Agenda

- Apache Cordova plug-in overview
- Implementing an Objective-C code plug-in
- Adding a plug-in to DOM
- Invoking a plug-in from JavaScript

Adding a plug-in to DOM

- The second step of the plug-in implementation is to declare it in the DOM and create a wrapper for it:

```
function HelloWorldPlugin(){

}

HelloWorldPlugin.prototype.sa[illegible]lo = function (onSayHelloSuccess, onSayHelloFailure, name){
    cordova.exec(onSayHelloSucc[illegible] onSayHelloFailure, "HelloWorldPlugin", "sayHello", [name]);
};

cordova.addConstructor(function(){
    if (!window.plugins) window.plugins = {
    window.plugins.helloWorldPlugin = new H
});
```

First, create an empty function that will serve as a wrapper for the plug-in.

Adding a plug-in to DOM

- The second step of the plug-in implementation is to declare it in the DOM and create a wrapper for it:

```
function HelloWorldPlugin(){

}

HelloWorldPlugin.prototype.sayHello = function (onSayHelloSuccess, onSayHelloFailure, name){
    cordova.exec(onSayHelloSuccess, onSayHelloFailure, "HelloWorldPlugin", "sayHello", [name]);
};

cordova.addConstructor(funct  {
    if (!window.plugins) windo  gins = {};
    window.plugins.helloWorldPlug  new Hell
});
```

Create a sayHello function using the HelloWorldplugin prototype and hardcode the plug-in class name and action. It invokes the plug-in using cordova.exec() API.

Adding a plug-in to DOM

- The second step of the plug-in implementation is to declare it in the DOM and create a wrapper for it:

```
function HelloWorldPlugin(){

}

HelloWorldPlugin.prototype.sayHello = function (onSayHelloSuccess, onSayHelloFailure, name){
    cordova.exec(onSayHelloSuccess, onSayHelloFailure, "HelloWorldPlugin", "sayHello", [name]);
};

cordova.addConstructor(f tion(){
    if (!window.plugins) ndow.plugins = {
    window.plugins.hello ldPlugin = new H oWorldPlugin();
});
```

Success callback

Failure callback

Plug-in Java class name

Action name

Parameters array

Invoking the plug-in using JavaScript

- The second step of the plug-in implementation is to declare it in the DOM and create a wrapper for it:

```
function HelloWorldPlugin(){

}

HelloWorldPlugin.prototype.sayHello
    cordova.exec(onSayHelloSuccess,
};

cordova.addConstructor(function(){
    if (!window.plugins) window.plugins = {};
    window.plugins.helloWorldPlugin = new HelloWorldPlugin();
});
```

The final step is to add the helloWorldPlugin property to the DOM window.plugins object. From now on, you can invoke the plug-in using window.plugins.helloWorldplugin.sayHello().

Agenda

- Apache Cordova plug-in overview
- Implementing an Objective-C code of a plug-in
- Adding a plug-in to DOM
- Invoking a plug-in from JavaScript

Invoking a plug-in from JavaScript

- Now you are ready to invoke and use the plug-in from JavaScript:

```
function greetMe(){
    window.plugins.helloWorldPlugin.sayHello(sayHelloSuccess, sayHelloFailure, $("#NameInput").val());
}

function sayHelloSuccess(data){
    alert("OK: " + JSON.stringify(data));
}

function sayHelloFailure(data){
    alert("FAIL: " + JSON.stringify(data));
}
```

Success and failure callbacks

Invoking a plug-in from JavaScript

- The sample for this training module can be found in the Getting Started page of the IBM Worklight documentation website at http://www.ibm.com/mobile-docs

Check yourself questions

- In order to recognize a plug-in in a JavaScript application it should be added to:
  - Cordova.plist file.
  - Worklight.plist file.
  - Plugins.plist file.
  - Plug-in will be automatically recognized by JavaScript without adding it to any of above files.
- When should Cordova plug-ins be used?
  - When a developer wants to implement his application in the native code because he is not familiar with JavaScript.
  - When a developer wants his application to look more like a native application.
  - When a developer wants to gain access to OS APIs which are not accessible within the web container.
  - When a developer needs to retrieve data from a remote server.
- What are the components of a Cordova plug-in?
  - Native class implementing the required functionality. It can be called directly from application's JavaScript.
  - Native class implementing the required functionality and a JavaScript wrapper for it. The wrapper's functions can be called from JavaScript.
  - Native class implementing the required functionality, JavaScript wrapper for it and declaration in application-descriptor.xml file.
  - JavaScript wrapper only. Native classes are already provided by Worklight.

Check yourself questions

- In order to recognize a plug-in in a JavaScript application it should be added to:
  - Cordova.plist file.
  - Worklight.plist file.
  - Plugins.plist file.
  - Plug-in will be automatically recognized by JavaScript without adding it to any of above files.
- When should a Cordova plug-in be used?
  - When a developer wants to implement his application in the native code because he is not familiar with JavaScript.
  - When a developer wants his application to look more like a native application.
  - When a developer wants to gain access to OS APIs which are not accessible within the web container.
  - When a developer needs to retrieve data from a remote server.
- What are the components of a Cordova plug-in?
  - Native class implementing the required functionality. It can be called directly from application's JavaScript.
  - Native class implementing the required functionality and a JavaScript wrapper for it. The wrapper's functions can be called from JavaScript.
  - Native class implementing the required functionality, JavaScript wrapper for it and declaration in application-descriptor.xml file.
  - JavaScript wrapper only. Native classes are already provided by Worklight.

Notices

- Permission for the use of these publications is granted subject to these terms and conditions.
- This information was developed for products and services offered in the U.S.A.
- IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.
- IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
  - IBM Director of Licensing
    IBM Corporation
    North Castle Drive
    Armonk, NY 10504-1785
    U.S.A.
- For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:
  - Intellectual Property Licensing
    Legal and Intellectual Property Law
    IBM Japan Ltd.
    1623-14, Shimotsuruma, Yamato-shi
    Kanagawa 242-8502 Japan
- **The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law**: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.
- This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.
- Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.
- IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.
- Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:
  - IBM Corporation
    Dept F6, Bldg 1
    294 Route 100
    Somers NY 10589-3216
    USA
- Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.
- The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.
- Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

COPYRIGHT LICENSE:

- This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.
- Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

  © (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

Support and comments

- For the entire IBM Worklight documentation set, training material and online forums where you can post questions, see the IBM website at:
  - http://www.ibm.com/mobile-docs
- **Support**
  - Software Subscription and Support (also referred to as Software Maintenance) is included with licenses purchased through Passport Advantage and Passport Advantage Express. For additional information about the International Passport Advantage Agreement and the IBM International Passport Advantage Express Agreement, visit the Passport Advantage website at:
    - http://www.ibm.com/software/passportadvantage
  - If you have a Software Subscription and Support in effect, IBM provides you assistance for your routine, short duration installation and usage (how-to) questions, and code-related questions. For additional details, consult your IBM Software Support Handbook at:
    - http://www.ibm.com/support/handbook
- **Comments**
  - We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this document. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.
  - For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.
  - When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state.
  - Thank you for your support.
  - Submit your comments in the IBM Worklight forums at:
    - https://www.ibm.com/developerworks/mobile/mobileforum.html
  - If you would like a response from IBM, please provide the following information:
    - Name
    - Address
    - Company or Organization
    - Phone No.
    - Email address

Thank You