上一章我们介绍了cordova android项目是如何运行的，这一章我们介绍cordova的核心内容，插件的调用。演示一个例子，通过cordova插件，去调用摄像头。

一、插件的安装以及基本信息：

　　我们先在项目中安装调用摄像头的插件cordova-plugin-camera，cd到hello下，执行：

cordova plugin add cordova-plugin-camera

　　(1)然后它会为我们的hello/plugins文件夹下添加cordova-plugin-camera插件（这个是整个这个插件模块，可用于安卓和ios。卸载插件用cordova plugin remove cordova-plugin-camera）；

　　(2)同时也会在我们的platforms/android/platform\_www/plugins下面添加对应的安卓的插件cordova-plugin-camera（这个是根据对应的平台生成的对应的插件内容，只可用于本安卓平台）；

　　(3)也会在我们的资源文件夹assets/www/plugins下添加cordova-plugin-camera插件（这个资源文件夹下的所有的文件，才是真正的app运行的时候调用的，加载的文件）；

　　(4)还会为我们的src下添加*org.apache.cordova.camera包以及对应的java文件。*

　　为什么要在项目中加这么多，这只是cordova的机制，我们不作重点研究。这三个目录是有不同的作用的，但我们只重点关注assets/www目录下。

　　除了上面这些，加完这个插件也会改变一些其他文件的内容，首先看res/xml/config.xml，最后多了一项配置信息：

<feature name="Camera">

<param name="android-package" value="org.apache.cordova.camera.CameraLauncher" />

</feature>

　　feature · name："js中间件通过它调用java方法"。  
　 （中间件就是我们的插件对应的一些js文件。我们写的js程序通过这些插件（中间件）去调用Jjava原生内容）  
　　param · name："android-package"     value：原生插件类的包类路径。

　　打开assets/www下cordova\_plugins.js，发现其中也增加了很多内容：

[复制代码](javascript:void(0);)

cordova.define('cordova/plugin\_list', function(require, exports, module) {

module.exports = [

{

"id": "cordova-plugin-camera.Camera",

"file": "plugins/cordova-plugin-camera/www/CameraConstants.js",

"pluginId": "cordova-plugin-camera",

"clobbers": [

"Camera"

]

},

{

"id": "cordova-plugin-camera.CameraPopoverOptions",

"file": "plugins/cordova-plugin-camera/www/CameraPopoverOptions.js",

"pluginId": "cordova-plugin-camera",

"clobbers": [

"CameraPopoverOptions"

]

},

{

"id": "cordova-plugin-camera.camera",

"file": "plugins/cordova-plugin-camera/www/Camera.js",

"pluginId": "cordova-plugin-camera",

"clobbers": [

"navigator.camera"

]

},

{

"id": "cordova-plugin-camera.CameraPopoverHandle",

"file": "plugins/cordova-plugin-camera/www/CameraPopoverHandle.js",

"pluginId": "cordova-plugin-camera",

"clobbers": [

"CameraPopoverHandle"

]

}

];

module.exports.metadata =

// TOP OF METADATA

{

"cordova-plugin-whitelist": "1.3.0",

"cordova-plugin-compat": "1.0.0",

"cordova-plugin-camera": "2.3.0"

};

// BOTTOM OF METADATA

});

[复制代码](javascript:void(0);)

　　id:插件中某个模块或具体功能的id

　　file:这个id对应的js文件（实现这个模块或功能的文件）

　　pluginId:插件的ID，通过这个ID可以进行插件的安装和卸载

　　clobbers:H5或js通过它去调用js中间件（插件）中定义的方法

二、在代码中如何调用摄像头插件

　　上面我们说的都是在执行了cordova plugin add cordova-plugin-camera，安装好插件之后所产生的内容以及插件的一些基本信息，配置信息。接下来我们说一下如何简单使用这个插件。

　　首先我们修改index.html（指的assets/www下的）

[复制代码](javascript:void(0);)

<!DOCTYPE html>

<html>

<head>

<meta name="format-detection" content="telephone=no">

<meta name="msapplication-tap-highlight" content="no">

<meta name="viewport" content="user-scalable=no, initial-scale=1, maximum-scale=1, minimum-scale=1, width=device-width">

<link rel="stylesheet" type="text/css" href="css/index.css">

<title>Hello World</title>

</head>

<body>

<div class="app">

<!-- 这里添加一个button去调用自己写的拍照函数 -->

<button onClick="takePhoto()">拍照</button>

</div>

<script type="text/javascript" src="cordova.js"></script>

<script type="text/javascript" src="js/index.js"></script>

</body>

</html>

[复制代码](javascript:void(0);)

　　接下来再修改index.js

按 Ctrl+C 复制代码



按 Ctrl+C 复制代码

　　至此，我们的Cordova摄像头插件已经在代码中调用好了，运行我们的项目，就可以拍照了。（当然，模拟器上是不行的，得用真机）

三、getPicture()方法详解：

　　camera.getPicture(cameraSuccess, cameraError, cameraOptions)  
　　选择使用摄像头拍照，或从设备相册中获取一张照片。图片以base64编码的字符串或图片URI形式返回。返回值会按照用户通过cameraOptions参数所设定的下列格式之一发送给cameraSuccess回调函数：

　　1.一个字符串，包含Base64编码的照片图像（默认情况）。  
　　2.一个字符串，表示在本地存储的图像文件位置。

　　cameraSuccess：

　　提供图像数据的onSuccess回调函数。

function(imageData) {

// 对图像进行处理

}

　　参数imageData：根据cameraOptions的设定值，为Base64编码的图像数据或图像文件的URI。（字符串类型）

　　errorCallback:

　　提供错误信息的onError回调函数。

function(message) {

// 显示有用信息

}

　　参数：message：设备本地代码提供的错误信息。（字符串类型）

　　cameraOptions：  
　　定制摄像头设置的可选参数。

[复制代码](javascript:void(0);)

{

quality : 75,

destinationType : Camera.DestinationType.DATA\_URL,

sourceType : Camera.PictureSourceType.CAMERA,

allowEdit : true,

encodingType : Camera.EncodingType.JPEG,

targetWidth : 100,

targetHeight : 100

};

[复制代码](javascript:void(0);)

　　quality：存储图像的质量，范围是[0,100]。（数字类型）  
　　destinationType：选择返回数据的格式。通过navigator.camera.DestinationType进行定义。（数字类型）  
　　备注：强烈建议将“Camera.destinationType”设为FILE\_URI。  
　　sourceType：设定图片来源。通过nagivator.camera.PictureSourceType进行定义。（数字类型）  
　　allowEdit：在选择图片进行操作之前允许对其进行简单编辑。（布尔类型）  
　　EncodingType：选择返回图像文件的编码方式，通过navigator.camera.EncodingType进行定义。（数字类型）  
　　targetWidth：以像素为单位的图像缩放宽度，必须和targetHeight同时使用。相应的宽高比保持不变。（数字类型）  
　　targetHeight：以像素为单位的图像缩放高度，必须和targetWidth同时使用。相应的宽高比保持不变。（数字类型）  
　　MediaType：设置选择图片的类型，只有当PictureSourceType is PHOTOLIBRARY or SAVEDPHOTOALBUM时才会生效，该参数由nagivator.camera.MediaType (数字类型)定义。