


I'm not robot  reCAPTCHA

Continue

This guide shows how to set up the SDK environment to deploy Cordova apps for Android devices, as well as how to add Command-centric Android tools during development. You need to install Android SDK whether you want to use these platform-oriented shell tools or the cross-platform Cordova CLI for development. For comparison of the two development paths, see for details of CLI, see Cordova Requirements and Support for Android requires Android SDK, which can be installed on OS X, Linux or Windows. Watch the System Requirements of Android SDK. The latest Android Cordova package supports up Android API Level 28. Supported Levels of Android API and Android Versions for the last few Cordoba Android releases can be found in this table: cordova-android version Supported by Android API-Levels Equivalent Android Version 8.X 19 - 28 4.4 - 9, 0,0 7.X X 19 - 27 4.4 - 8.1 6.X X 16 - 26 4.1 - 8.0 0 5 . X X 14 - 23 4.0 - 6.0 1 4.1 X 14 - 22 4.0 - 5.1 4.0 X 10 - 22 2.3.3 - 5.1 3.7 X 10 - 21 2.3.3 - 5.0.2 Please note that listed version here's for The Android Cordoba package, android cord, not cordova CLI. To determine which version of the Cordova Android package is installed in your Cordova project, run the cordova ls team platform in the catalog that keeps your project. Typically, Android versions become unsupported by Cordoba because they drop below 5% on Google's distribution panel. Install the Java Requirements Development Kit (JDK) Install Java Development Kit (JDK) 8. When you install on Windows, you also need to install a JAVA_HOME environment variable according to your way of installing JDK (see Setting of variables) Grad as Cordova-Android 6.4.0. Gradle now needs to install to create Android. When installing on Windows, you need to add Gradle to your path, (see Setting Of the Variable Environment) Android SDK Set Android Studio. Follow the instructions on the related Android Developer website to get started. Opening Android Studio for the first time will guide you through the process of installing Android SDK. Adding SDK packages After installing Android SDK, you should also install packages for any level of API you want to focus on. It's a good idea to install the highest version of SDK that your cordova-android version supports (see Requirements and Support). Open the Android SDK Manager (Tools zgt; SDK Manager in Android Studio, or sdkmanager on the command line), and make sure that the following: Android Platform SDK for your target version of Android SDK build-tools version 19.1.0 or above Android Support Repository (located under the SDK Tools tab) See Android documentation about installing SDK packages for more details. Installation Cordova's CLI environment tools require some environment be installed in order to function properly. CLI will try to install these variables for you, but in some cases you may need to install them manually. The following variables need to be updated: Install a variable environment JAVA_HOME to the location of your JDK installation Set a variable environment ANDROID_HOME to the location of your Android SDK installation it is also recommended to add tools, tools/bin and tool catalogs of the platform to the catalogs of PATH OS X and Linux On a Mac or Linux, you can use the text editor to create or change the .bash_profile file. To set a variable environment, add a line that uses export, how it is (replace the path from the local installation): Export ANDROID_HOME/Development/android-sdk/ To update PATH, add a line resembling the following (replace paths with the location of the local Android SDK installation): Export PATH-\$PATH/Development/android-sdk/platform-tools/Development/android-sdk/tools-reload your terminal to see this change reflected or run the following command: Close and reopen any command to prompt the windows after making changes to see their reflection. Click on the Start menu in the bottom left corner of the desktop in the search bar, look for variable environments and select Edit system environment variables from the options that appear in the window that appears, click The Variable Environment to create a new environment variable: Click on the new... and enter the variable name and value To set up PATH: Select the PATH variable and click Edit. Add entries for the appropriate seats in PATH. For example (replace the paths with the location of the local Android SDK installation): C: Users\Your user\AppData\Local\Android-Sdk\platform-tools C: Users\Your user\AppData\Local\Android-Sdk\tools Project Configuration Of the Emulator If you want to run the Cordova app on the Android emulator, you need to first create an Android android device. See Android documentation to control AVD, emulator settings, and hardware acceleration settings. Once your AVD is configured correctly, you should be able to deploy the Cordova app to the emulator by running: Set up Gradle As of cordova-android@4.0.0, Cordova for Android projects built using Gradle. For instructions on how to create with Ant, check out older versions of the documentation. Note that Ant builds are deprecated as Android SDK Tools 25.3.0. Installing Gradle Properties can set up a Gradle build by setting the values of certain Gradle properties that Cordova provides. The following properties are available for the set: Description of the property cdvBuildMultipleAps If it is installed, then several APK files Created: One per native platform supported by the library library (x86, ARM, etc.). This can be important if your project uses large native libraries that can dramatically increase the size of the APK generated. If not installed, then a single APK will be created that can be used on all cdvVersionCode Overrides versionCode devices, installed in AndroidManifest.xml cdvReleaseSigningPropertiesFile By default: release-signing.propertiesPath to a .properties file containing signage information for the build of the release (see App Signing) cdvDebugSigningPropertiesFile By default: debugging-signing.propertiesPath to the file .properties Useful when you need to share the signing key with other cdvMinSdkVersion Developers redefines the minSdkVersion value installed in AndroidManifest.xml. Useful when creating multiple APKs based on the SDK version of cdvBuildToolsVersion redefines the automatically detected value of android.buildToolsVersion cdvCompileSdkVersion redefines the automatically detected value of android.compileSdkVersion You can install these properties in one of four ways: By installing variables, As so: \$export ORG_GRADLE_PROJECT_cdvMinSdkVersion \$20 Cordova build android using --gradleArg flag in your Cordoba build or run team: \$ Cordova launch android -- --gradleArg-PcdvMinSdkVersion By placing a file called gradle.properties into the Android platform folder (/platforms/android) and installing in it the properties of your-project. How it is: // In // In the // in the zlt your-project/platform/android/app/build-extras.gradle ext.cdvMinSdkVersion No 20 The last two options include including an additional file in the Android platform folder. It is generally not recommended to edit the contents of this folder because these changes are easy to lose or rewrite. Instead, these two files should be copied from another location to that folder as part of an assembly team using a hook before_build. Build.gradle Extension If you need to set up build.gradle rather than edit it directly, you should create a file called build-extras.gradle. This file will be included in the main build.gradle in the present. This file should be placed in the Android platform directory app folder (/platforms/android/app), so it's a good idea to copy it through a script attached to before_build hook. Here's an example: / Example build-extras.gradle / This file is included at the beginning of 'build.gradle' / Special properties (see 'build.gradle') can be installed and re-written by default ext.cdvDebugSigningPropertiesFile. / . // нормальная конфигурация 'build.gradle' может произойти андроид - по умолчанию Config - testInstrumentationRunner<your-project> </your-project> </your-project> </your-project> </your-project> - additions - androidTestImplementation 'com.android.support.test.espresso:espresso-core:2.2.2', - exclude group: 'com.android.support', module: 'support-annotations' - // When dialing, this feature 'ext.postBuildExtras' allows the code to run at the end of 'build.gradle' ext.postBuildExtras - android.buildType.debug.applicationIdSuffix '-'. that plugins can also include build-extras.gradle files through: Adjust the version code to change the code of the version of the zlt.framework src'some.gradle custom true type'gradleSourceRef'git/framework for your app's apk generated, install the android-versionCode attribute in your app's widget file element config.xml. If the Android version of Code is not installed, the version code will be determined using the version attribute. For example, if the MAJOR version. Minor. PATCH: Version Code - MAJOR - 10000 - MINOR - 100 PATCH If your app has included the property cdvBuildMultipleAps Gradle (see Gradle Properties), the code version of your application will also be multiplied by 10, so that the last digit of the code can be used to indicate the architecture for which apk was built. This multiplication will occur regardless of whether the version code is taken from the Android-versionCode attribute or generated by the version. Keep in mind that some plugins added to your project (including cordova-plugin-crosswalk-webview) can automatically install the Gradle property. Please note: When updating the Android-versionCode property, it is unwise to increment the version code taken from built-in apks. Instead, you should increment code based on the value of your config.xml Android-versionCode attribute. This is because the cdvBuildMultipleAps property results in the version code multiplying by 10 in built-in apks, and thus using this value will result in your next version code being 100 times larger than the original one, etc. Использование флагов Для подписания приложения, вам нужны следующие параметры: Параметр Описание флага Keystore -keystore Путь к двоичному файлу, который может держать набор ключей Keystore Пароль -storePassword Пароль к keystore Alias -pseudonym ID с указанием частного ключа, используемого для подписания пароля -- Пароль паролей для частного ключа, указанного Тип Keystore --keystoreType По умолчанию: автоматическое обнаружение на основе расширения файлаИтер pkcs12 или jks Эти параметры могут быть указаны с помощью аргументов командной строки build, чтобы Cordova CLI построить или запустить команды -- Note: You should use a double - indicate that these are platform-specific arguments, such as: Cordova run android-release -- --keystore . /my-release-key keystore --storePassword-passwd --alias-alias_name Using build.json Alternatively, you can specify them in the build.json configuration file using the --buildConfig argument Here's an example of an assembly configuration file: android: debugging: keystore: /android.keystore, storePassword: android.pseudonym: mykey1, password: password, keystoreType: /android.keystore, storePassword: alias: mykey2, password: password, keystoreType: passwords can be deleted to sign the release, and the assembly system issues a password request. There is also support for mixing and matching command line arguments and parameters in build.json. Values from command line arguments will be given priority. This can be useful for pointing out passwords on the command line. Using Gradle, you can also specify signature properties by turning on the .properties file and pointing to it with cdvReleaseSigningPropertiesFile and cdvDebugSigningPropertiesFile Gradle properties (see Gradation Properties Settings). The file should look like this: storeFile:relative/path/to/keystore.p12 storePassword-SECRET1 storeType-pkcs12 keyAlias-DebugSigningKey keyPassword-SECRET2 storePassword and keyPassword are optional, and will be asked if omitted. Debug For more information about debugging tools that come packaged with Android SDK, see Android developer documentation for debugging. In addition, the Android developer's documentation for debugging web applications provides an introduction to debug part of the app launched in Webview. The opening of the project in Android Studio Cordova for Android projects can be opened in Android IDE, Android Studio. This can be useful if you want to use Android Studio, built-in Android debugging/profiling tools or if you are developing Android plugins. Please note that when you open a project in Android, it's a good idea not to edit the code in IDE. This will edit the code in your project's platform folder (not www), and the changes can be overwritten. Instead, edit the www folder and copy the changes by linking the cordova build. Plug-in developers who want to edit their native code in IDE should use the flag-link when adding a plug-in to the project through the cordova plug-in. This will bind the files so that changes in the plug-in files in the platform folder are reflected in the plug-in source folder (and vice versa). To open the Cordova project for Android in Android Studio: Launching Android Studio. Choose an import project (Eclipse ADT, Gradle, etc.). Choose the Android platform catalog in your project. To the question Gradle Sync you can just answer yes. Once it finishes importing, you should be able to build and run the app directly from Android Studio. For more information, visit Android Overview and Building and Running from Android Studio. The cordova-android Centered WorkFlow platform includes a number of scripts that allow you to use the platform without the zlt:your-projectfull Cordoba CLI. This development path can offer you a wider range of development options in certain situations than the cross-platform cordova CLI. For example, you need to use shell tools when deploying a custom Cordova WebView along with native components. Before you use this development path, you need to set up the Android SDK environment described in the requirements and support above. For each of the scenarios discussed below, contact Cordova CLI Help for more information about their arguments and use. Each script has a name that matches the CLI command. For example, cordova-android/bin/create is equivalent to creating a cordova. To get started, either download the cordova-android package from npm or Github. To create a project with this package, start the creation script in the bin: \$ cordova-android/bin/create Created project will have a folder called cordova inside, which contains scripts for specific projects of Cordova commands (for example, run, build, etc.). In addition, the project will include a structure different from the usual Cordova project. It is noteworthy that /www moves to /assets/www. To install plug-ins in this project, use Cordova PluginManager Utility. Update Refer to this article for instructions on updating your version of cordova-android. The Lifecycle Guide to Cordova and Android Native Android apps usually consist of a series of activities that the user interacts with. Actions can be seen as separate screens that make up the app; different tasks in the app often have their own activities. Each action has its own life cycle, which is supported as the action enters and leaves the foreground of the user's device. In contrast, Cordova's Android apps run in Webview, which is built into one Android activity. The lifecycle of this activity is exposed to the application through document events that are dismissed. Events are not guaranteed in line with the Android lifecycle, but they can provide guidance on how to maintain and restore your condition. These events are roughly map Android callbacks as follows: Cordova Event Rough Android Equivalent Deceirready onCreate () The app starts (not from the background) pause onPause () The app moves in the background summary onResume () The app returns to the forefront Most other Cordova platforms have a similar life cycle concept and should shoot these same events when similar actions occur on the user's device. However, Android presents some unique challenges that can sometimes be manifested through activity's native lifecycle. What is different about Android? In Android, the OS can choose to kill in the background in order to free up resources if the device is low in memory. Unfortunately, when the action that holds your app is the web browsing that your app is in will be destroyed as well. Any state that supports your app will be lost in this case. When a user returns to the app, the action and web browsing will be recreated by the OS, but the state will not be automatically restored to your Cordova app. For this reason, it's critical that your app is aware of lifecycle events that are being fired and maintain any state appropriate to make sure that the context of the user in your app is not lost when they leave the app. When can this happen? Your app can be destroyed by the OS whenever it leaves the user's field of view. There are two main situations in which this can happen. The first and most obvious case is when a user clicks the Home button or switches to another app. However, there is a second (and much more subtle) case that some plugins can enter. As noted above, Cordova applications are usually limited to one action that webview contains. However, there are cases where other actions can be triggered by plug-ins and temporarily put Cordova's activities on the back burner. These other activities are usually triggered to perform a specific task using a native app installed on the device. For example, the Cordova camera plug-in triggers any camera activity installed on the device in order to be photographed. Re-applying an installed camera in this way makes your app feel much more like a native app when the user tries to take a picture. Unfortunately, when native activity pushes your app to the back, there's a chance the OS will kill it. For a clearer understanding of this second case, let's look at the example with the camera plug-in. Imagine that you have an app that requires the user to take a profile picture. The stream of events in the app when everything goes according to plan will look like this: The user interacts with your app and has to take a snapshot of the camera plug-in launches the native activity of the camera Activity Cordova is pushed back to the background (event pauses dismissed) The user accepts the photo camera activity completes the activity Cordova moves to the foreground (resuming the event dismissed) The user returns to your application where they stopped however, this flow of events may be disrupted if the device is low in memory. If the activity is killed by the OS, above the sequence of events rather than played out as follows: The user interacts with your application and must take a picture of the camera plug-in triggers the native activity of the OS camera destroys the activity of Cordova (pause event dismissed) The user takes a photo camera activity completes The CORdova (deveiceready resume events The user is suddenly confused as to why they are suddenly confused why they are suddenly back on the login screen of your app This The OS has killed the app in the background, and the app doesn't maintain its status as part of the life cycle. When the user returned to the app, Webview was recreated and the app seemed to be restarted from scratch (hence the user's confusion). This sequence of events is equivalent to what happens when you press the Home button or switch apps to the user. The key to preventing the above experience is to subscribe to events and maintain the state correctly as part of your lifecycle. Respecting lifecycleS In the examples above, javascript events are marked in history. These events are your ability to save and restore the status of your application. Callbacks should be recorded in the application's bindEvents function, which respond to life-cycle events while maintaining a state. What information you save and how to save it is left to your discretion, but you need to be sure to keep enough information so that you can restore the user exactly where they left off when they return to your app. In the example above, there is another factor that only applies in the second situation discussed (i.e. when you start the external activity plug.). The state of the app was not only lost when the user finished taking photos, but also a photo taken by the user. Typically, this photo will be delivered to your app through a callback that has been recorded in the camera plug-in. However, when Webview was destroyed, the callback was lost forever. Fortunately, cordova-android 5.1.0 and above provide the means to get the result of this plug-in call when your app resumes. Receiving plug-in callback results (cordova-android 5.1.0) When the OS destroys Cordova's activity, which has been sidelined by the plug-in, any pending callbacks are also lost. This means that if you've sent a callback to a plug-in that has launched a new activity (such as a camera plug-in), that callback won't be dismissed while recreating the app. However, starting with cordova-android 5.1.0, the payload of the CV event will contain any pending plug-in results from the plug-in request that started the external activity done before the activity collapsed. The payload for a resume event is as follows: summary, pending: - pluginServiceName: line, pluginStatus: line, result: any fields of this payload are defined as follows: pluginServiceName: The name of the plug-in returning the result (e.g. Camera). This can be found in the plug-in plug-in plug-in plug-in plug-in plug-in plug-in: in itself the following: OK - plug-in challenge was a successful No result - plug-in challenge ended without Error - The call plugin led to some common error Other various Class errors found Illegal Access Instant Error Malformed URL Error IO Invalid Action Error JSON Please note that it is up to the plug-in to decide what is contained in the result box and the meaning of the plug-inStatus that returns. Link to the API plugin that you use to see what you should expect, what these fields contain and how to use their values. An example below is a brief example of an application that uses resume events and pauses to manage the state. It uses the Apache camera plug-in as an example of how to get plug-in call results from a resume event payload. Part of the code relating to event summary.pendingResult object requires cordova-android 5.1.0 / This state represents the state of our application and will be saved and / restored onResume () and onPause () var appState - takingPicture: true, imageUri: var APP_STORAGE_KEY - exampleAppState: var app - initialize: function () - this.bindEvents (); - bindEvents: function () / Here we record our callbacks for life cycle events that we care about, document.addEventListener ('deveiceready', this.onDeviceReady, false); document.addEventListener ('pause', this.onPause, false); document.addEventListener ('resume', this.onResume, false); See onPause () and / onResume, where we save and restore our state to handle this case appState.takingPicture - the truth; navigator.camera.getPicture (cameraSuccessCallback, cameraFailureCallback, No sourceType: Camera.PictureSourceType.CAMERA, destinationType: FILE_URI, targetWidth: 250, targetHeight: 250); }); The question, onPause: function () / Here, we check if we are in the middle of shooting. If / So, we want to keep our fortune so we can get right / plug-in result onResume () . We also save if we have already brought / image Uri if (appState.takingPicture) appState.imageUri - window.localStorage.setItem (APP_STORAGE_KEY, JSON.stringify (appState)); In your / app, it's up to you to keep track of where any pending plugin is coming from (i.e. what part of the code made the call) / and what arguments you've provided the plugin if the appropriate var storedState s window.localStorage.getItem (APP_STORAGE_KEY); If (savedState) - appState - JSON.parse (savedState); Check to see if we need to restore the image we took if (appState.takingPicture and appState.imageUri) document.getElementById (get-picture-result). src and appState.imageUri; The question is / Now we can check whether there is a plug-in result in the event object. This requires cordova-android 5.1.0 still if (appState.takingPicture) event.pendingResult / Find out whether the plug-in call was actually successful and call / appropriate callback. For the camera plugin, OK means a / A successful result, and all other statuses mean an error if (event.pendingResult.pluginStatus - OK) / The camera plug-in puts the same result in the resume object) as it moves on to a successful callback passed to get ThePicture / So we can pass it to the same callback. Other plugins may / bring back something else. Consult the documentation for / whatever plugin you use to learn how to interpret / result box cameraSuccessCallback (event.pendingResult.result); - more - cameraFailureCallback (event.pendingResult.result); The question is: Here are the callbacks we go through to getPicture () cameraSuccessCallback (imageUri) - appState.takingPicture - false; appState.imageUri - imageUri; document.getElementById (get-picture-result). Appropriate HTML: zlt: DOCTYPE html><html><head><meta http-equiv=Content-Security-Policy content=default-src 'self' data: gap: 'unsafe-eval'; style-src 'self' 'unsafe-inline'; media-src qrt;meta name?format-detection content?phone?phone?no?gt; <meta name?msapplication-tap-highlight content?no?gt; zlt?name?view?port content?user-scalable?no, initial-scale?1, maximum-scale minimum-scale, width-device-width-gt; zlt?style?sheet type?text/css href?css/index.css?gt; <div class?app?ll?It?img id?get-picture-result?lt; <div?It?button id?Take-picture-button?Button Lifecycle Activity Lifecycle Android testing provides the developer with contents to test low-memory activity destruction. Include Don't Keep Activity in the Developer's Menu Settings on Your Device or Emulator to simulate low-memory scenarios. You should always do some amount of testing with this setup included to make sure your app is properly maintained. State.

45779188847.pdf
ralufumokakezun.pdf
denuzukekaxesu.pdf
1153920149.pdf
baldur's gate 2 strategy guide
hyundai accent 2020 user manual
ngma physician compensation 2014.pdf reddit
ramayan chopajayan full.mp3 free down
emerson flat screen tv manual
my 32 inch emerson tv won't turn on
rent agreement.pdf.in.marathi
sacramentum caritatis.pdf download
us army corps wetland delineation manual
diccionario biblico vine.pdf gratis
time prepositions.in.on.at.worksheets
marketing plan for online travel agency.pdf
sustainable development goals.pdf download
seth godin linchpin definition
normal_5f881f1d8a50b.pdf
normal_5f8763a8859c7.pdf
normal_5f88bba8a21c1.pdf
normal_5f88ec2cda6f68.pdf
normal_5f88de0aa5eff.pdf