Cordova tutorial pdf tutorialspoint

This plug-in is used to obtain information about the user's device. Step 1 - Installing a plug-in device To install this plug-in, we need to run the next snippet in a command hint. C: UsersUsernameDesktop-CordovaProject'gt; cordova-plugin add cordova-plugin device Step 2 - Adding a button We will use this plugin just like we have used other Cordova plugins. Add a button in index.html. This button will be used to obtain information about the device. (button id)cordovaDevice/zLADOVA)lt;/button/gt; Step 3 - Adding Events Listener Cordova plugins are available after the devicereadyready event, so we'll validate the listener event inside onDeviceReady features in index.js. document.getElementById (cordovaDevice).addEventListener (click, cordovaDevice); Step 4 - Creating the Next Feature feature will show how to use all the features the plugin provides. We're going to smear it in index.js. cordovaDevice function () - Alert (Cordova version: - device.cordova - Device model: - device.model - Device platform: - device.platform - UUID device: At the touch of the CORDOVA DEVICE button the alert will display the cordova version, device model, platform, UUID and version of the device. S.No software and description 1 NodeJS and NPM NodeJS is the platform needed to develop Cordova. Check out our NodeJS settings for more information. 2 Android SDK for the Android platform, you must have an Android SDK installed on your car. For more information, check out the Android environment setting up. 3 XCode for the iOS platform, you must have xCode installed on your computer. Check out the iOS setting environment for more information about installing Cordova Before we start, you should know that we will use the Windows tip command in our tutorial. Step 1 - Install git Even if you don't use git, it should be installed, since Cordova uses it for some background processes. You can download git here. Once you've installed git, open the environment variable. Right-Click On Settings Advanced Computer System Settings Variable Variable System Edit Copy next at the end of the variable value field. This is the default way to install git. If you set it on a different path, you should use it instead of our example code below. ; C: Program Files (x86)Git'bin; C: Program files (x86)Git'cmd You can now enter git in the command hint to check if the installation is successful. Step 2 - Installation of Cordova This step will load and install the Cordova module throughout Open the team's hint and run the following: User name qgt;npm set -g cordova You can check check The C-run version: Users username'gt;cordova-v is all you need to start developing Cordova apps in the Windows operating system. In our next tutorial, we'll show you how to create the first app. We figured out how to install Cordova and create an environment for that. Once everything is ready, we can create our first cordova hybrid app. Step 1 - Create an App Open directory where you want the app to be installed in a command hint. We're going to create it on the desktop. C: Users Username'gt;cordova create CordovaProject io.cordova.hellocordova CordovaApp CordovaProject - a catalog where the app is created. io.cordova.hellocordova is the default domain value. You should use your own domain value if possible. CordovaApp is the name of your app. Step 2 - Adding platforms that need to open a project directory in a team request. In our example, this is CordovaProject. You only have to choose the platforms you need. To be able to use this platform, you need to install a specific SDK platform. As we evolve on windows, we can use the following platforms. We have already installed Android SDK, so we will install only android platforms for this tutorial. C: Users of the username DesktopCordovaProject's corda platform add android There are other platforms that can be used on Windows. C: User'username Platform (User'username) Desktop-CordovaProject'gt'gt;cordova, add wp8 C: Users username Desktop-CordovaProject'gt; cord platform add amazon-fireos C: Users username Desktop CordovaProject Platform qgt;corda add windows C: Users username DesktopCordovaProject'gt;cord platform add blackberry10 C: UsersUsername'Desktop-CordovaProject's corda platform add firefoxos If you develop on Mac, you can use - $ cordova platform add iOS $ cordova platform to add Amazon-fireos $ cordova platform add Platform from your project with the help of : C: Users username Desktop-CordovaProject'gt;cordova build android Now we can launch our application. If you're using the default emulator, you should use: Users username DesktopCordovaProject'gt;Cordova emulate Android If you want to use an external emulator or a real device that you should use : Users username DesktopCordProjectova'gt;launch android NOTE We will use the Genymotion android emulator, as it is faster and responsive. You find the emulator here. You can also use a real-world device for testing by turning usb debugging out of the options and connecting it to your computer via a USB cable. For some devices, you will also need to install USB USB Once we run the app, it will install it on the platform we've pointed out. If it's all over without errors, the output should show the app's start screen by default. In our next tutorial, we'll show you how to set up the Cordova app. Cordova is a platform for creating hybrid mobile applications using HTML, CSS and JavaScript. Official documentation gives us the definition of Cordova: Apache Cordova is an open source mobile development platform. This allows standard web technologies such as HTML5, CSS3 and JavaScript to be used for cross-platform development, avoiding every language in mobile platform development. Applications run in platform-oriented wrappers and rely on standard-matching APIs to access sensors, data, and network status of each device. Cordoba Features Let's Now understand the peculiarities of Cordoba briefly. Command Line Interface (Cordova CLI) This tool can be used to run projects, build processes for different platforms, install plug-ins and many other useful things that facilitate the development process. You'll learn how to use the command line interface in future chapters. Cordova Core Components Cordova offers a set of basic components that every mobile application needs. These components will be used to build an application base so we can spend more time implementing our own logic. Cordova Plugins Cordova offers an API that will be used to implement native mobile features in our javaScript app. The Cordova license is licensed under the Apache license, Version 2.0. Apache and Apache pen logos are trademarks of the Apache Software Foundation. The benefits of Cordova Now we discuss the benefits of Cordoba. Cordova offers one platform for creating hybrid mobile apps so we can develop one app that will be used on various mobile platforms - iOS, Android, Windows Phone, Amazon-fireos, BlackBerry, Firefox OS, Ubuntu and tizien. It's faster to develop a hybrid app, then a native app so Cordova can save on development time. Because we use JavaScript when working with Cordova, we don't need to learn specific platform programming languages. There are many community add-ons that can be used with Cordova, they have multiple libraries and frameworks that are optimized to work with it. Cordoba's limitations are following Cordoba's limitations. Hybrid applications are slower than domestic applications, so it's not optimal to use Cordova for larger applications that require a lot of data and functionality. Cross-browser compatibility can create a lot of problems. Most of the time we build applications for platforms, so testing and optimization can be time-consuming, as we need to cover a large number of devices and operating systems. Some plug-ins have compatibility issues with different devices and platforms. There are also some native APIs that have not yet been Cordoba. Cordova is a platform for creating hybrid mobile applications using HTML, CSS and JavaScript. Official documentation gives us the definition of Cordova: Apache Cordova is an open source mobile development platform. This allows standard web technologies such as HTML5, CSS3 and JavaScript to be used for cross-platform development, avoiding every language in mobile platform development. Applications run in platform-oriented wrappers and rely on standard-matching APIs to access sensors, data, and network status of each device. Cordoba Features Let's Now understand the peculiarities of Cordoba briefly. Command Line Interface (Cordova CLI) This tool can be used to run projects, build processes for different platforms, install plug-ins and many other useful things that facilitate the development process. You'll learn how to use the command line interface in future chapters. Cordova Core Components Cordova offers a set of basic components that every mobile application needs. These components will be used to build an application base so we can spend more time implementing our own logic. Cordova Plugins Cordova offers an API that will be used to implement native mobile features in our javaScript app. The Cordova license is licensed under the Apache license, Version 2.0. Apache and Apache pen logos are trademarks of the Apache Software Foundation. The benefits of Cordova Now we discuss the benefits of Cordoba. Cordova offers one platform for creating hybrid mobile apps so we can develop one app that will be used on various mobile platforms - iOS, Android, Windows Phone, Amazon-fireos, BlackBerry, Firefox OS, Ubuntu and tizien. It's faster to develop a hybrid app, then a native app so Cordova can save on development time. Because we use JavaScript when working with Cordova, we don't need to learn specific platform programming languages. There are many community add-ons that can be used with Cordova, they have multiple libraries and frameworks that are optimized to work with it. Cordoba's limitations are following Cordoba's limitations. Hybrid applications are slower than domestic applications, so it's not optimal to use Cordova for larger applications that require a lot of data and functionality. Cross-browser compatibility can create a lot of problems. Most of the time we build applications for different platforms, so testing and optimization can be time consuming as we need to cover a large number of devices and operating systems. Some plug-ins have compatibility issues with different devices and platforms. There are also local APIs that are not yet supported by Cordoba. Cordoba - Environment Setting In this chapter, we will see the environment installation of Cordova. To start the installation, you must first install several components. The components are listed in the following table. Table. NodeJS and NPM NodeJS software and description is the platform needed to develop Cordova. Check out our NodeJS settings for more information. 2 Android SDK for the Android platform, you must have an Android SDK installed on your car. For more information, check out the Android environment setting up. 3 XCode for the iOS platform, you must have xCode installed on your computer. Check out the iOS setting environment for more information about installing Cordova Before we start, you should know that we will use the Windows tip command in our tutorial. Step 1 - Install git Even if you don't use git, it should be installed, since Cordova uses it for some background processes. You can download git here. Once you've installed git, open the environment variable. Right-Click On Settings Advanced Computer System Settings Variable Variable System Edit Copy next at the end of the variable value field. This is the default way to install git. If you set it on a different path, you should use it instead of our example code below. ; C: Program Files (x86)Git'bin; C: Program files (x86)Git'cmd You can now enter git in the command hint to check if the installation is successful. Step 2 - Installation of Cordova This step will load and install the Cordova module worldwide. Open the command hint and run the following: Userusername'gt;npm install-g cordova You can check the installed version by leaking to C: Users username'gt;cordova-v Is all you need to start developing Cordova applications on the Windows operating system. In our next tutorial, we'll show you how to create the first app. Cordova - The first application We figured out how to install Cordova and create the conditions for it. Once everything is ready, we can create our first cordova hybrid app. Step 1 - Create an App Open directory where you want the app to be installed in a command hint. We're going to create it on the desktop. C: Users Username'gt;cordova create CordovaProject io.cordova.hellocordova CordovaApp CordovaProject - a catalog where the app is created. io.cordova.hellocordova is the default domain value. You should use your own domain value if possible. CordovaApp is the name of your app. Step 2 - Adding platforms that need to open a project directory in a team request. In our example, this is CordovaProject. You only have to choose the platforms you need. To be able to use this platform, you need to install a specific SDK platform. As we evolve on windows, we can use the following platforms. We have already installed Android SDK, so we will install only Android for this textbook. C: Users username DesktopCordovaProject'gt;Cord platform add android There are other platforms that can be used on Windows OS. C: Users username DesktopCordovaProject'gt;cordova platform add wp8 wp8 platform to add amazon-fireos C: Users username'Desktop'gt; CordovaProject'gt;cord platform add windows C: Users username'Desktop-CordovaProject's cordal platform add blackberry10 C: Users username Desktop CordovaProject'gt;cordova platform add firefox if you develop on Mac, You can use the - cordova platform to add iOS $ cordova platform to add Amazon-fireos $ cordova platform to add android $cordova platform to add blackberry10 $ cordova platform to add firefoxos You can also remove the platform from your project qgt; using : C: Users username Desktop-CordovaProject'gt;Cordova build android Now we can launch our app. If you're using the default emulator, you should use: Users username DesktopCordovaProject'gt;Cordova emulate Android If you want to use an external emulator or a real device that you should use : Users username DesktopCordProjectova'gt;launch android NOTE We will use the Genymotion android emulator, as it is faster and responsive. You can find the emulator here. You can also use a real-world device for testing by turning usb debugging out of the options and connecting it to your computer via a USB cable. For some devices, you will also need to install a USB driver. Once we run the app, it will install it on the platform we've pointed out. If it's all over without errors, the output should show the app's start screen by default. In our next tutorial, we'll show you how to set up the Cordova app. Cordova - config.xml File File config.xml is a place where we can change the configuration of the application. When we created our app in the last tutorial, we installed a reverse domain and a name. Values can be changed in the config.xml file. When we create the app, the default config file will also be created. The following table explains the configuration elements in config.xml. config.xml Configuration Table S.No Element - Details 1 widget App the reverse domain value that we indicated when creating the app. 2 name app name that we pointed out when creating the app. 3 Description description for the app. 4 author of the app. 5 app's homepage content. It is inside the www. 6 plug-in plugins that are currently installed. 7 Access is used to manage access to external domains. The default origin is set on th, which means that access to any domain is allowed. This value will not allow you to open some specific URLs to protect the information. 8 allow-intent allows certain URLs to ask for an app to open. For example, zlt;allow-intent Will allow the bodies: links to open dialer. 9 platform to create an app. Cordova - Storage We can use API storage available for customer application data. This will help you use the app when the user is offline, and it can also improve performance. Since this tutorial is for beginners, we'll show you how to use local storage. In one of our later tutorials we will show you other plugins that can be used. Step 1 - Adding buttons We will create four buttons in the index.html file. The buttons will be located inside the class div element and the app. SET LOCAL STORAGE It will produce the next screen - Step 2 - Adding Events to Cordova's Security Policy Listeners does not allow inline events, so we will add the event to listeners inside the index.js files. We will also assign a window.localStorage local variable, which we will use later. document.getElementById (setLocalStorage).addEventListener (click, setLocalStorage); document.getElementById (showLocalStorage).addEventListener (click, showLocalStorage); document.getElementById (deleteProjectFromLocalStorage).addEventListener (click, deleteProjectFromLocalStorage); document.getElementById (getLocalStorageByKey).addEventListener (click, getLocalStorageByKey); var localStorage - window.localStorage; Step 3 - Creating features Now we need to create features that will be called when you press the buttons. The first feature is used to add data to your local store. setLocalStorage - localStorage.setItem localStorage.setItem localStorage.setItem The next one will be to record the data we've added to the console. showLocalStorage () - console.log (localStorage.getItem); console.log (localStorage.getItem); console.log (localStorage.getItem)); If we put the SET LOCAL STORAGE button, we put three items in the local store. If we tap SHOW LOCAL STORAGE after that, the console will register the items we want. Now let's create a feature that removes the project from the local store. removeProjectFromLocalStorage If we press the SHOW LOCAL STORAGE button after the project is deleted, the result will show zero value for the project field. We can also obtain local storage items using the key method () which will take the index as an argument and return the item with the appropriate index value. getLocalStorageByKey () - console.log (localStorage.key (0)); Now that we're going to see the GET BY KEY button, the next exit will be displayed. NOTE When we use the key () The console will register the work instead of the name, even if we passed the 0 argument to get the first object. This is because the local store stores the data in alphabetical order. Teh Teh Teh table shows all available local storage methods. S.No methods and details 1 setItem (key, value) Is used to customize the item for local storage. 2 getItem (key) Is used to retrieve goods from local storage. 3 DeleteItem (key) Used to remove an item from the local store. 4 keys (index) used to obtain an item using an item index in a local store. This helps to sort the items in alphabetical order. 5 length () Is used to extract the number of items that exist in the local store. 6 Clear () Used to remove all pairs of keys/values from local storage. Cordoba - Events There are various activities that can be used in Cordoba projects. The following table shows the events available. S.No Events and Details 1 deviceReady This event is triggered after Cordova is fully loaded. This helps ensure that Cordova features are not triggered before everything is loaded. 2 pause This event is triggered when the app is in the background. 3 summary This event is triggered when the app returns from the background. 4 backbutton This event is triggered by the click of the Back button. 5 menubton This event is triggered by pressing the menu button. 6 searchbutton This event is triggered by the press of the Android search button. 7 startcallbutton This event is triggered by pressing the call start button. 8 endcallbutton This event is triggered by pressing the end button of the call. 9 volumedownbutton This event is triggered by pressing the volume button down. 10 volumeupbutton This event is triggered by pressing the volume button up. The use of events All events are used almost equally. We should always add event listeners to our js instead of calling an event, as Cordova's Content Security Policy does not allow Javascript to enter. If we try to call an inline event, the next error will appear. The right way to handle events with addEventListener. We'll see how to use the volumeupbutton event as an example. document.addEventListener (volume upbutton, callbackFunction, false); CallBack FeatureFunction () - Alert (volume button pressed!) As soon as we press the volume button up, the screen will display the following warning. Processing the Back button We have to use the Android Back button for the functionality of the app as a return to the previous screen. To implement our own functionality, we must first disable the Back button that is used to exit the App. document.addEventListener (backbutton, onBackKeyDown, false); onBackKeyDown (e) - e.preventDefault Alert (button back pressed!); Now that we're pressing the Back button on our native Android, an alert will appear on the screen instead of From the app. This is done with the e.preventDefault command. Cordova - Back Button You usually want to use the Android Back button for some of the app's features, such as returning to the previous screen. To be able to implement functionality, you first need to disable the exit from the app at the touch of the Back button. document.addEventListener (backbuy, onBackKeyDown, false); onBackKeyDown (e) - e.preventDefault Alert (button back pressed!); Now that we press the Back button on our native Android, an alert will appear on the screen instead of leaving the app. This is done with e.preventDefault. Cordova - Plugman Cordova Plugman is a useful command line tool for installing and managing plug-ins. You should use a plugin if your app has to run on one particular platform. If you want to create a cross-platform app, you have to use cordova-cli, which will modify plugins for different platforms. Step 1 - Install Plugman Open the team window and run the next piece of code to install the plug-in. C: Users username DesktopCordovaProject'gt;npm install -g plugin step 2 - Installing plugins To understand how to install the Cordova plugin with the plug-in, we will use the plug-in camera as an example. C: Users of username-Desktop-CordovaProject'gt;plugman install--platform android-project-platform-android-plugin-cordin-camera-remove--platform android--project platform---plugin-plugin-camera We should consider three parameters, as shown above. --platform - platform that we use (android, ios, amazon-fireos, wp8, blackberry10). --project - the way on which the project is built. In our case, it is a catalog of android platforms. --plugin is the plug-in that we want to install. When you set the actual parameters, the command window should show the next output. Additional methods can be used in the same way. C: Users username Desktop'gt; CordovaProject'gt;plugman remove --platform android-project platform --plugin-cordova-plugin-camera Team hint console will display the next output. Plugman offers some additional techniques that can be used. The methods are listed in the following table. S.No method and details of the one set used to install Cordova plugins. 2 remove used to remove Cordova plugins. 3 Bring Used to copy the Cordova plugin in a certain place. 4 Prepare to update the configuration file to help support the JS module. 5 adduser Is used to add a user account to the registry. 6 Publish Used to publish plugin in registry. 7 unpublished is used for non-publishing plug-in from the registry. 8 Search Used to search for plugins in the registry. 9 configuration configuration of the configuration of the registry parameters. 10 to create Used to create a custom plug-in. 11 platform used for or remove the platform from a custom created plug-in. Extra commands If you're stuck, you can always use the plug-in-aid team. The version can be checked using the plug-in -v. To find the plugin, you can use the search plugin and finally you can change the registry plugins using the help of config to establish a roster team. NOTE Since Cordova is used for cross-platform development, in our subsequent chapters we will use Cordova CLI instead of Plugman to install plug-ins. Cordova - Battery Status This Cordova plug-in is used to monitor the state of the device's battery life. The plug-in will monitor every change that occurs with the device's battery. Step 1 - Installing a Plugin battery to install this plug-in, we have to open the team's operational window and run the next code. C: Users of Username'Desktop-CordovaProject'gt; cordova-pluginbattery-status Step 2 - Add Event Listener When you open index.js file, you will find the feature onDeviceReady. A listener of the event should be added here. window.addEventListener (batterystatus, onBatteryStatus, false); Step 3 - Create a callback feature We will create the onBatteryStatus callback feature at the bottom of the index.js file. onBatteryStatus (info) - Alert (BATTERY STATUS: Level: - info.level isPlugged: - info.isPlugged); When we run the app, the alert is triggered. At the moment, the battery is charging 100%. If status changes, you'll see a new alert. Battery condition shows that the battery is now charging at 99%. If you connect your device to a charger, the new alert will show that isPlugged has been changed to true. Additional Events This plugin offers two additional events besides the batterystatus event. These events can be used in the same way as a battery event. S.No Event - Details 1 of the Battery Event is triggered when the battery percentage reaches low cost. This value varies depending on the different devices. 2 Battery Critical event is triggered when the battery percentage reaches a critical value. This value varies depending on the different devices. Cordova - Camera This plug-in is used to capture or use files from the image gallery. Step 1 - Install the plug-in camera Run the next code in the team window to install this plugin. C: Users username DesktopCordovaProject'gt;cordova-plugincamera Step 2 - Adding buttons and images now, we will create a button to call the camera and img, where the image will be displayed after being accepted. This will be added to index.html inside the div class and app element. Step 3 - Adding the Event Listener Event event is added inside the onDeviceReady function to make sure Cordova is loaded before we start using it. document.getElementById (cameraTakePicture).addEventListener (click, cameraTakePicture); Step 4 - We will create the cameraTakePicture feature, which is transmitted as a callback to our event listener. He will be fired at the touch of a button. Inside this feature, we'll call the global navigator.camera object provided by the plug-in plug-in If the shooting is successful, the data will be sent to the onSuccess callback feature, if not, an alert with an error message will be shown. We will now this code at the bottom of index.js. cameraTakePicture feature - navigator.camera.getPicture (onSuccess, onFail, quality: 50, destinationType: Camera.DestinationType.DATA_URL); onSuccess (imageData) - var image.getElementById ('myImage'); image.src - data:image/jpeg;base64 onFail When we start the app and press the button, the home camera will be triggered. When we take and save the image, it will appear on the screen. The same procedure can be used to obtain an image from a local file system. The only difference is in the function created at the last stage. You can see that the original optional option has been added. Step 1 B C: UsersUsernameDesktop-CordovaProject'gt;Cordova-plugin to add cordova-plugincamera Step 2 B zlt'button id'cameraGetPicture'gt;GET PICTURE Step'lt;button'gt; 3 B document.getElementById (cameraGetPicture). addEventListener (click, cameraGetPicture); Step 4 B cameraGetPicture () - navigator.camera.getPicture (onSuccess, onFail, quality: 50, destinationType: Camera.DestinationType.DATA_URL, sourceType: Camera.PictureSourceType.PHOTOLIBRARY); onSuccess (imageURL) - var image.getElementById ('myImage'); image.src - imageURL; When we press the second button, the file system will open instead of the camera so we can select the image that will be displayed. This plug-in offers a lot of extra options to customize. S.No Option - Details 1 image quality in the 0-100 range. The default is 50. 2 destinationType DATA_URL or 0 Returns base64 coded line. FILE_URI or 1 Returns URI Image File. NATIVE_URI or 2 Returns the image of the native URI. 3 sourceType PHOTOLIBRARI OR 0 Opens photo library. CAMERA or 1 Opens your home camera. SAVEDPHOTOALBUM or 2 Opens a saved photo album. 4 allowEdit to edit images. 5 codingType JPEG or 0 Returns JPEG encoded image. PNG or 1 Returns a coded PNG image. 6 Target Image Scaling Width in Pixels. 7 targetHeight Image scaling height in pixels. 8 mediaType PICTURE or 0 allows only the selection of images. VIDEO or 1 only allows the selection of videos. ALLMEDIA or 2 allows all types of multimedia of choice. 9 Adjustment orientation is used to correct the orientation of the image. 10 saveToPhotoAlbum Used to save images on the photo album. 11 popoverOptions used to install popover location on iOS. 12 cameraDirection FRONT or 0 Front camera. BACK or 1 Rear Camera. ALLMEDIA Cordova - Contacts This plug-in to access the device's contact database. In this tutorial, we'll show you how to create, request, and delete contacts. Step 1 - Contact Contacts C: Users of Username'Desktop-CordovaProject'gt;cordova-plugincontacts Step 2 - Adding buttons button will be used to call the createContact function. We'll break it up in the div class and the index.html app. ADD CONTACT'LT;button id'findContact'gt;find CONTACT/GTT)/button document.getElementById (createContact). addEventListener (click, createContact); document.getElementById (findContact). addEventListener (click, findContact); document.getElementById addEventListener (click, deleteContact); Step 3A - Callback (navigator.contacts.create) Now we don't have any contacts stored on the device. Our first callback feature will be called navigator.contacts.create method where we can specify new contact details. This will create a contact and assign it to the myContact variable, but it will not be stored on the device. To save it, we need to call the save method and create success and call back functions errors. createContact - var myContact - navigator.contacts.create (displayName: Test User); myContact.save (contactSuccess, contactError); contactSuccess feature - alert (Contact saved!); When we press the ADD CONTACT button, the new contact will be stored in the device's contact list. Step 3B - Callback (navigator.contacts.find) Our second callback feature will request all contacts. We will use navigator.contacts.find. The option object has a filter option that is used to treat the search filter. used somewhat true, as we want to return all contacts with the device. The field key is to find contacts on displayName, as we used it to keep contact. After a set of options, we use the find method to request contacts. The alert message will work for every contact you find. findContacts options.filter - ; options.multiple and the truth; Fields and displayName; navigator.contacts.find contactfindSuccess (contacts) - for (var i y 0; i qlt; contacts.length; i) q alert (Display Name) function contactfindError(message) { alert('Failed because: ' + message); } } When we press the FIND CONTACT button, one alert popup will be triggered since we have saved only one contact. Step 3C - Callback function (delete) In this step, we will use the find method again but this time we will set different options. The options.filter is set to search that Test User which has to be deleted. After the callback function has returned contacts.length;= i++)= {= alert(display= name=></ contacts.length; i++) { alert(Display Name = + contacts[i].displayName); } } function contactfindError(message) { alert('Failed because: ' + message); } } When we press the FIND CONTACT button, one alert popup will be triggered since we have saved only one contact. Step 3C - Callback function (delete) In this step, we will use the find method again but this time we will set different options. The options.filter is set to search that Test User which has to be deleted. After the contactfindSuccess callback function has returned > > Contact we want, we will remove it using a removal method that requires its own success and reverse errors. deleteContact - var options - new ContactFindOptions options.filter - Test User; options.multiple - false; Fields - displayName; navigator.contacts.find contactfindSuccess (contacts) contact.remove (contactRemoveSuccess, contactRemoveError); contactRemoveSuccess (contact) - alert (Contact removed); Now we have only one contact stored on the device. We'll manually add another one to show you the removal process. We will now press the DELETE CONTACT button to remove the user's test. If we check the contact list again, we'll see that the test user no longer exists. Cordova - This plug-in device is used to obtain information about the user's device. Step 1 - Installing a plug-in device To install this plug-in, we need to run the next snippet in a command hint. C: UsersUsernameDesktop-CordovaProject'gt; cordova-plugin add cordova-plugin device Step 2 - Adding a button We will use this plugin just like we have used other Cordova plugins. Add a button in index.html. This button will be used to obtain information about the device. (button id)cordovaDevice/zLADOVA)lt;/button/gt; Step 3 - Adding Events Listener Cordova plugins are available after the devicereadyready event, so we'll validate the listener event inside onDeviceReady features in index.js. document.getElementById (cordovaDevice).addEventListener (click, cordovaDevice); Step 4 - Creating the Next Feature feature will show how to use all the features the plugin provides. We're going to smear it in index.js. cordovaDevice function () - Alert (Cordova version: - device.cordova - Device model: - device.model - Device platform: - device.platform - UUID: At the touch of cordOVA DEVICE button the alert will display the cordova version, Model device, platform, UUID and version of the device. Cordova - Accelerometer Accelerometer plug-in is also called device-movement. C: UsersUsernameDesktop-CordovaProject'gt;cordova-plugin-movement-movement Step 2 - Add buttons At this stage, we'll add two buttons to the index.html file. One will be used to get the current acceleration, and the other will monitor the acceleration changes. Button id'getAccelerationStep 3 - Add Event Listeners Let's Now add event listeners to our onDeviceReady buttons features inside index.js. document.getElementById (getAcceleration). addEventListener (click, getAcceleration); document.getElementById (watchAcceleration).addEventListener (click, watchAcceleration); Step 4 - Creating features now, we'll create two features. The first feature will be used to generate the current acceleration, and the second function will monitor the acceleration, and the acceleration information will be triggered every three seconds. We'll also add a clearWatch wrapped with setTimeout to stop the acceleration viewing after the deadline. The frequency option is used to trigger the callback function every three seconds. getAcceleration () - navigator.accelerometer.getCurrentAcceleration Accelerometer functionSuccess (acceleration) - alert ('Acceleration X:' - acceleration.x'Acceleration Y:' - acceleration.y'acceleration:' 'Acceleration:' B; - watchacceleration functions () - var accelerometerOptions - frequency: 3000 - var watchID - navigator.accelerometer.watchAcceleration AccelerometerEror, AccelerometerOptions); accelerometer functionUss (acceleration) - alert ('Acceleration X: 'Acceleration.x' 'Acceleration Y': '- Acceleration':' 'Acceleration': 'z' ' 'Timestamp':' - acceleration.timestamp'; No, 10000); Now, if we press the GET ACCELERATION button, we get the current acceleration value. If we press the WATCH ACCELERATION button, the warning will be triggered every three seconds. After the third alert, the clearWatch feature will be called, and we won't receive any more alerts as we set time out up to 10,000 milliseconds. Cordova - The Compass Orientation Device is used to show direction relative to the cardinal's geographical northern point. Step 1 - Install the Device Orientation plug-in Open the operational team window and run the following. C: Users of Username'Desktop-CordovaProject'gt;corda-pluginevice-orientation Step 2 - Add buttons This plugin is similar to the acceleration plugin. Now let's create two buttons in index.html. THE APP STEP 3 - Add the Listeners Now event, we'll add an event of listeners inside onDeviceReady features in index.js. document.getElementById (getOrientation). addEventListener (click, getOrientation); watchOrientation); Step 4 - Creating features We will create two functions; The first feature will generate the current acceleration, and the other will check the orientation changes. You can see that we are using the frequency option again to monitor the changes that occur every three seconds. getOrientation function () - navigator.compass.getCurrentHeading compassSuccess (headline) - alert ('headline: '-heading.magneticHeading); compassError feature (error) - alert ('CompassError: ' - error.code); compassSuccess feature (headline) - alert ('Heading: ' - heading.magneticHeading); The Timeout set (function) - navigator.compass.clearWatch (watchID); Since the compass plugin is almost the same as the acceleration plugin, we'll show you the error code this time. Some devices don't have the magnetic sensor that you need to operate the compass. Cordova - Dialogues Cordova Dialogues plugin will call the platform a native dialogue user interface element. Step 1 - Install the Dialog Type next command in the team window to install this plugin. C: Users of Username-Desktop-CordovaProject'gt;cordova-plugin-dialogues Step 2 - Add buttons Let's now open index.html and add four buttons, one for each type of dialogue. <button id=dialogAlert>ALERT</button> <button id=dialogConfirm>CONFIRM</button> <button id=dialogPrompt>PROMPT</button> <button id=dialogBeep>BEEP</button> Шаг 3 - Добавить событие Слушатели Теперь мы добавим событие слушателей внутри функции onDeviceReady в index.js. Listeners will call back after clicking the appropriate button. document.getElementById (dialogAlert). addEventListener (click, dialogAlert); document.getElementById (dialogConfirm).addEventListener (click, dialogConfirm); document.getElementById (dialogPrompt). addEventListener (click, dialogPrompt); document.getElementById (dialogBeep). addEventListener (click, dialogBeep); Step 4A - Create an alert feature, as we've added four listener events, we'll now create callback features for all of them in index.js. First dialogueAlert. function dialogAlert () - var message - I am Alert Dialog!; var title - ALERT; var buttonName - Alert Button; navigator.notification.alert alertCallback feature - console.log If we press the ALERT button, we'll see a dialogue alert window. When you press the dialogue button, the next exit will appear on the console. Step 4B - Create The second function we have to create is the dialogConfirm function. function dialogConfirm () - var message - I confirm The Dialogue?; var title - CONFIRM; var buttonLabels - YES,NO; navigator.notification.confirm confirmCallback (buttonIndex) - console.log (you pressed the Index button) A new dialogue pops up when you press the CONFIRMATION button. We press the YES button to answer the question. The next exit will be displayed on the console. Step 4C - Create a quick function of the Third function of the dialogPrompt function. This allows users to type text into the dialogue input element. function dialogPrompt () - var message - Am I Prompt Dialog?; var title - PROMPT; var buttonLabels - YESNO; var defaultText - Default navigator.notification.prompt (message, promptCallback, name, buttonLabels, defaultText); promptCallback (result) - You entered - result.input1); The PROMPT button will run the dialog box like the next screenshot. In this dialogue field, we have the opportunity to introduce a text. We'll put this text in the console along with the button that's pressed. Step 4D - Create a beep function The latest of these is the dialogBeep feature. This is used to call a beep notification. The time setting will set the number of res repeated beeps. dialogBeep function - var times 2; navigator.notification.beep When we press the BEEP button, we will hear the notification sound twice, as the time value is set at 2. Cordova - File system This plug-in is used to manipulate the native file system on the user's device. Step 1 - Installing plug-in files We need to run the next code in a command hint to install this plugin. C: Users of Username'Desktop-CordovaProject'gt;cordova-plugin-file Step 2 - Add buttons In this example, we will show you how to create a file, write a file, read it and delete it. For this reason, we will create four buttons in

index.html. We will also add textarea in which, the contents of our file will be shown. CREATE FILE'lt;button (button id)readFile/FILE'lt;button/gt; button id'removeFile/file Step/button/textarea id'textarea'gt;gt;lt;textarea/textarea'gt; 3 - Add the Listener event We will add the event to the audience in index.js inside the function onDeviceReady document.getElementById (createFile). addEventListener (click, createFile); document.getElementById (writeFile). addEventListener (click, writeFile); document.getElementById (readFile). addEventListener (click, readFile); document.getElementById (removeFile). Step 4A Create a file function File will be created in the root folder of applications on the device. To have access to the root folder, you must give the superuser access to the folders. In our case, the way to the root folder is datadata-com.example.hello-cache. At the moment, this folder is empty. Now let's add a feature that will create a log.txt file. We will write this code to index.js and send the request to the file system. This method is used by WINDOW. TEMPORARY or WINDOW. Permanent. The amount required for storage is estimated by bytes (5MB in our case). createFile function () - var type and window. TEMPORARY; var size 5'1024-1024; window.requestFileSystem (type, size, successCallback, errorCallback) successCallback (fs) - fs.root.getFile ('log.txt', Create: True, exclusive: true, function (fileEntry) - Alert ('Making the file successfully!') Now we can check our root folder apps again and we can find our new file there. Step 4B - Write a file feature At this stage we will write the text to our file. to be able to write the Lorem Ipsum text, which we assigned to the variable drop. (fs) - fs.root.getFile ('log.txt', 'Create: true, fileEntry, fileEntry.createWriter (fileWriter) - fileWriter.onwriteend - feature (e) - alert ('Write completed.'); Type: text/simple); fileWriter.write (blob); » , errorCallback); » , errorCallback); Callback error, ERROR: After clicking the WRITE FILE button, the alert will tell us that the spelling is successful, as in the next screenshot. Now we can open log.txt and see that Lorem Ipsum is written inside. Step 4C - Read the file feature At this point we'll read log.txt and display it in the textarea element. We will send the request to the file system and receive the file object, and then create the reader. When the reader is loaded, we will assign the returned textarea value. readFile function -- var type and window. TEMPORARY; var size 5'1024-1024; window.requestFileSystem ErrorCallback Success (fs) - fs.root.getFile ('log.txt', function (fileEntry) - fileEntry.file (file) - var reader - new FileReader (); reader.onloadend - function (e) - var txtArea - document.getElementId. ErrorScall); B, errorSzing); errorCallback - alert (ERROR: - error.code) - When we press the READ FILE button, the text from the file will be written inside textarea. Step 4D - Delete the file function and finally we'll create a feature to delete the log.txt file. removeFile function (. . . . . TEMPORARY; var size 5'1024-1024; window.requestFileSystem (type, size, successssS errorCallback (fs) - fs.root.getFile ('log.txt', 'Create: false, feature (fileEntry) fileEntry.remove (function) alert ('file deleted.'); » » , errorCallback); errorCallback error warning ERROR: We can now click the DELETE FILE button to remove the file from the folder of the root folder of applications. The alert will notify us that the removal operation has been successful. If we check the root folder, we'll see that it's empty. Cordova - File transfer This plug-in is used to download and download files. Step 1 - Installing a file transfer plug-in We need to open a command request and run the next team to install the plugin. C: Users of Username'Desktop-CordovaProject'gt; cordova-plugin-file transfer Step 2 - Create buttons In this chapter, we'll show you how to download and download files. Let's create two buttons in index.html qlt;button id'uploadFile'gt;upload'lt;button'lt;button id'downloadFile'gt;DOWNLOAD'lt;/button'gt; Step 3 - Add Event Listeners event listeners will be created in index.js inside the function onDeviceReady. We add click events and callback features. document.getElementById (uploadFile). addEventListener (click, uploadFile); document.getElementById (downloadFile). addEventListener (click, downloadFile); Step 4A - Download feature This feature will be used to download files from server to device. We uploaded the file to the postimage.org to make things simpler. You'll probably want to use your own server. The function is placed in index.js and triggered when the appropriate button is pressed. uri is a link to a server load, and fileURI is the way to the DCIM folder on our device. downloadFile - var fileTransfer - new FileTransfer var uri - encodeURI (; var fileURL - //storage/emulated/0/DCIM/myFile; fileTransfer.download (uri, fileURL, function (entry) - console.log (download full: - entry.toURL();console.log (download error target - error.target); console.log (download error code - error.code); issue, false, headliners: Authorization: Basic dGVzdHVzXJuYW1lOnRlc3RwYXNzd29y-y); As soon as we press the DOWNLOAD button, the file will be downloaded from postimg.org to our mobile device. will look like this: Step 4B - Download feature Now let's create a feature that will take the file and upload it to the server. Again, we want to simplify this as much as possible, so we'll use posttestserver.com server for testing. The value of uri will be tied to be placed in a posttestserver. uploadFile feature -- var fileURL - //storage/emulated/0/DCIM/myFile var uri - encodeURI (; var options - new FileUploadOptions options options.fileKey - file options.fileName - fileURL.substr (fileURL.lastIndexOf,'/') 1); options.mimeType - text/simple; var headlines ft.upload (fileURL, uri, onSuccess, onError, options); onSuccess (r) - console.log console.log console.log (Sent - r.bytesSent); onError function (error) - alert (error: code and error.code); console.log (download error source - error.source); console.log (target of download error - error.target); Now we can press the UPLOAD button to trigger this feature. Cordova - Geolocation geolocation is used to obtain information about the latitude and longitude of the device. Step 1 - Installing plug-in We can install this plugin by entering the following code for the operational window command. C: UsersUsernameDesktop-CordovaProject'gt;cordova-plug-in step 2 - Add buttons In this tutorial we will show you how to get the current position and how to follow the changes. First, we need to create buttons that will call these features. (button id)getPosition'gt;CURRENT POSITION)lt;button We'll add a sample of the code below to onDeviceReady in index.js. document.getElementById (getPosition). addEventListener (click, getPosition); document.getElementById addEventListener (click, watchPosition); Step 3 - Create Features Two Features should be created for two event listeners. One will be used to obtain an ongoing position and the other to monitor the situation. getPosition () - var options - enableHighAccuracy: true, maximumAge: 360,000 - var watchID - navigator.geolocation.getCurrentPosition (onSuccess, onError, options); function onSuccess (position) - alert ('Latitude: ' - position.coords.latitude' 'Longitude: ' - position.coords.longitude' Height: 's position.coords Height - ''Precision:' - position.coords.accuracy':'Precision of Height:' s.coords.altitude's':'title: 's position.coords.heading' '" ' Timestamp: ' - position.timestamp ''); }; onError function - alert ('code: ' - error.code' '' 'message:' - error.'; watchPosition function () - var options - maximumAge: 360,000, time out: 3000, includeHighAccuracy: true, var watchID - navigator.geolocation.watchPositionPosition (onSuccess, onError, options); function onSuccess (position) - alert ('Latitude: ' - position.coords.latitude'Longitude: '- position.coords.longitude' Height: 'Headline: - Position.coords.heading'Speed: 's position.coords.speed' Timestamp:' For example, above we use two methods - getCurrentPosition and watchPosition. Both functions use three options. As soon as we press the CURRENT POSITION button, the alert will show the geolocation values. If we press the WATCH POSITION button, the same warning will be triggered every three seconds. In this way, we can track the movement changes in the user's device. NOTE This plug-in uses GPS. Sometimes it can't return the values on time, and the request returns the timeout error. That's why we've clarified: true and maximumAge: 3,600,000. This means that if the request is not completed on time, we will use the last known value instead. In our example, we set a maximum duration of up to 36,000,000 milliseconds. Cordova - Globalization This plugin is used to obtain information about the language language of the user's language, date and time zone, currency, etc. Step 1 - Installing the Globalization Plugin Open team hint and install the plugin by entering the following code C: Users username DesktopCordovaProject'gt;cordova plugin add cordova-plugin-globalization Step 2 - Add buttons We will add a few buttons for index.html THE BUTTON id'getLanguage DATE (button id)getDate'gt;/button/gt;button id'getCurrency/step/button/gt; 3 - Add event Listeners Event listeners will be added inside the getDeviceReady feature in the index.js file to make sure that our app and Cordova are downloaded before we start using it. document.getElementById (getLanguage).addEventListener (click, getLanguage); document.getElementById (getLocaleName). addEventListener (click, getLocaleName); document.getElementById (getDate). addEventListener (click, getDate); document.getElementById (getCurrency). addEventListener (click, getCurrency); Step 4A - Language function The first feature we use returns the BCP 47 language tag of the customer's device. We will use the getPreferredLanguage method. The feature has two options and Error. We add this feature to index.js. getLanguage () - navigator.globalization.getPreferredLanguage (onSuccess, onError); onSuccess - Alert ('language: 's language.value'); As soon as we press the LANGUAGE button, the alert will be shown on the screen. Step 4B - Locale Feature This feature returns the BCP 47 tag to local customer settings. This feature is similar to the one we've created before. The only difference is that this time we use the getLocaleName method. getLocaleName () - navigator.globalization.getLocaleName (onSuccess, onError); onSuccess (locale) - alert ('locale:' - locale.value); When we press the LOCALE button, the alert will show our lock tag. Step 4C - Date Feature This feature is used to return a date according to customer location and time zone settings. The date option is the current date, and the options are optional. getDate - var date - new date var options - formatLength:'short', selector: 'date and time' - navigator.globalization.dateToString (date, onSuccess, onError, options); onSuccess (date) - alert ('date: ' - date.value); Now we can run the app and press the DATE button to see the current date. The last feature we show is the return of currency values in accordance with the customer's device settings and the currency code ISO 4217. You can see that the concept is the same. getCurrency - var currencyCode - EUR; navigator.globalization.getCurrencyPattern (currencyCode, onSuccess, onError); onSuccess function (pattern) - alert ('pattern: 'pattern.pattern' '' 'code:' - pattern.code' faction: 'rounding' and 'decimal: ' pattern. onError(onError) warning (Pattern Receipt Error); The CURRENCY button will trigger an alert that will show users a currency pattern. This plugin offers other methods. You can see it all in the table below. Details of the parameters of the getPreferredLanguage onSuccess method, in the current language of the clientEror Returns. getLocaleName onSuccess, on the current customer localization settings. dateToString Date, onSuccess, onError, Options Returns Date according to Customer Localization and Time zone. stringToDate dateString, onSuccess, onError, Parses date options according to customer settings. getCurrencyPattern currencyCode, onSuccess, onError Returns of the customer's currency model. getDatePattern onSuccess, onError, options Returns customer date template. getDateNames onSuccess, onError, options returns an array of names of months, weeks or days in accordance with customer settings. isDayLightSavingsTime Date, successCallback, Error Challenge Used for If daylight saving time is active in accordance with the customer's time zone and calendar. getFirstDayOfWeek onSuccess, onError returns the first day of the week according to the customer's settings. ToString, onSuccess, onError, Options Returns number according to customer settings. toNumber, onSuccess, onError, Parses variants number according to customer settings. getNumberPattern onSuccess, onError, options returns the number pattern according to the customer's settings. Cordova - InAppBrowser This plugin is used to open a web browser inside the Cordova app. Step 1 - Installing plug-in We need to install this plugin in the operational team window before we can use it. C: Users of Username'Desktop-CordovaProject'gt;cordova-plugin-inappbrowser Step 2 - Add the button We will add one button to be used to open the InAppBrowser window in index.html. Step 3 - Add Event Listener Now let's add a listener event for our button in onDeviceReady features in index.js. document.getElementById (openBrowser). addEventListener (click, openBrowser); Step 4 - Create a feature At this stage we create a feature that will open the browser inside our app. We assign it to a variable ref, which we can use later to add listeners to events. openBrowser - var url - ' '; var target - '_blank'; var options - location - yes var ref - cordova. InAppBrowser.open (url, goal, options); ref.addEventListener ('loadstart', loadstartCallback); ref.addEventListener ('loadstop', loadstopCallback); ref.addEventListener ('loaderror', loaderrorCallback); ref.addEventListener loadstartCallback (event) - console.log ('Loading started: ' - event.url) - loadtopCallback feature (event) - console.log ('Loading finished: ' - event.url) - loaderrorCallback feature (error) - console.log ('download error: '- error.post) - exitCallback feature.) we'll see the next outing on the screen. The console will also listen to events. The start-up event will be prepared when the URL is downloaded and downloads when the URL is downloaded. We can see it in the console. As soon as we close the browser, the exit event will be prepared. There are other possible options for the InAppBrowser window. We'll explain this in the table below. S.No option is the details of one location used to turn the browser location bar on or off. Values yes or no. 2 Hidden Used to hide or show inAppBrowser. Values yes or no. 3 clearCache Is used to clear the browser cookie cache. Values yes or no. 4 clearsessioncache Used to clear the cache of session cookies. Values yes or no. 5 zoom Used to hide or show zoom control of the Android browser. Values yes or no. 6 Yes use the hardware back button to navigate back through the browser history. No to close the browser after pressing the Back button. We can use (link) variable for some other features. We'll show you just quick examples of this. We can use ref.removeEventListener (event name, callback) to remove listeners; We can use ref.close to close InAppBrowser. If we open a hidden window, we can show it - ref.show (); Even JavaScript can be entered into InAppBrowser - var parts - javascript/file/url ref.executeScript (details, callback); The same concept can be used for injections of CSS - var parts - css/file/url ref.inserCSS (details, callback); Cordova - Media Cordova media plugin is used to record and play sound sounds in Cordova applications. Step 1 - Installing the Media Plugin Media plug-in can be installed when the next code is launched in the team window. C: Users username DesktopCordovaProject'gt;corda plugin add cordova-plugin-media Step 2 - Add buttons In this tutorial, we will create a simple audio player. Let's create the buttons we need in index.html. Button id/playAudio/button BUTTON id'stopAudio/stop-lt;button/gt; VOLUME zlt;button id'volumeUp'gt;button'lt;button'lt;button id'volumeDown'gt;'3 - Add the Listeners event Now we need to add a listener event for our inside onDeviceReady features inside the index.js. document.getElementById (playAudio). addEventListener (click, playAudio); document.getElementById (pauseAudio).addEventListener (click, pauseAudio); document.getElementById (stopAudio). addEventListener (click, stopAudio); document.getElementById addEventListener (click, volumeUp); document.getElementById (volumeDown). addEventListener (click, volumeDown); Step 4A - The playback feature the first feature we're going to add is playAudio. We define myMedia outside of function because we want to use it in features that will be added later (pause, stop, volumeUp and volumeDown). This code is placed in the index.js file. var myMedia - zero; playAudio function -- var src - /android_asset/www/audio/piano.mp3; If (myMedia - null) - myMedia - new media (src, onSuccess, onError); onSuccess () - console.log (playAudio Success); myMedia.play(); We can press the PLAY button to start the piano music from the src path. Step 4B - Pause and stop the features next features that we need pauseAudio and stopAudio. pauseAradio - if (myMedia) - myMedia.pause (); Now we can pause or stop the sound of the piano by pressing pause or STOP buttons. Step 4C - Volume Features To set the volume we can use setVolume. This method takes a 0 to 1 option. We will set the starting value to 0.5. var volumeValue - 0,5; volumeUp () - если (myMedia) volumeValue < 1) { 1)= {=></ 1) { > > 0.1 euros); - volumeDown () - if (myMedia) volumeValue zgt; 0) - myMedia.setVolume (volumeValue - 0.1); Once we press VOLUME UP or VOLUME DOWN, we can change the volume by 0.1. The following table shows other methods that this plugin provides. S.No Method - Details 1 getCurrentPosition Returns the current position of the sound. 2 getDuration Returns the duration of sound. 3 playback Used to start or restart sound. 4 pauses used to pause sound. 5 release releases audio operating system resources. 6 seekTo Used to change the position of the sound. 7 setVolume Used to set volume for audio. 8 startRecord Start recording audio file. 9 stopRecord Stop recording audio file. 10 stop stopping the audio file from playing. Cordova - Media Capture This plug-in is used to access device capture options. Step 1 - Install media Capture Plugin To install this plugin, we will open the team's hint and run the following code: Users username DesktopCordovaProject'gt; cordova-plugin-media-capture Step 2 - Add buttons, as we want to show you how to capture audio, images and video, we will create three buttons in index.html. Button id'audioCapture/button/lt;button id'imageCapture'gt;imageCapture/button/button'lt'gt; button id'videoCapture/step 3 - Add the Listeners event The next step is to add listeners' events inside onDeviceReady in the index.js. document.getElementById (audioCapture). addEventListener (click, audioCapture); document.getElementById (ImageCapture). addEventListener (click, imageCapture); document.getElementById (videoCapture). addEventListener (click, videoCapture); Step 4A - Capture audio function The first callback feature in index.js is audioCapture. To start the sound recorder, we will use the captureAudio method. We use two options: the restriction will allow you to record only one audio clip per capture operation, and the duration - the number of seconds of the audio clip. audioCapture function - var options : 1, duration: 10 euros; navigator.device.captureAudio (onSuccess, onError, options); onSuccess (mediaFiles) - var i, path, len; for (i q 0, len - mediaFiles.length; i < len;= i= +=1) {= path=mediaFiles[i].fullPath; console.log(mediafiles);= }= }= function= onerror(error)= {= navigator.notification.alert('error= code:= '= += error.code,= null,= 'capture= error');= }= }= when= we= press= audio= button,= sound= recorder= will= open.= console= will= show= returned= array= of= objects= that= users= captured.= step= 4b= -= capture= image= function= the= function= for= capturing= image= will= be= the= same= as= the= last= one.= the= only= difference= is= that= we= are= captureimage= method= this= time.= function= imagecapture()= {= var= options={ limit:= 1= };= navigator.device.capture.captureimage(onsuccess,= onerror,= options);= function= onsuccess(mediafiles)= {= var= i,= path,= len;= for= (i=0, len=mediaFiles.length; i=>< len; i += 1) { path = len;= i= +=1) {= path=></ len; i += 1) { path = > > console.log (mediaFiles); Function onError (error) - navigator.notification.alert ('error code: ' - error.code, null, 'capture error'); Now we can press the IMAGE button to start the camera. When we shoot, the console will register an array with an object of the image. Step 4C - Capture video features Let's repeat the same concept for capturing videos. This time we will use the videoCapture method. videoCapture function - var options - limit: 1, duration: 10 euros; navigator.device.capture.captureVideo (onSuccess, onError, options); onSuccess (mediaFiles) - var i, path, len; for (i q 0, len - mediaFiles.length; i qlt; len; i q q q 1) q path'mediaFiles.fullPath; console.log (mediafiles);function) onerror (error)) The camera is open and we can't record the video.' once the video is saved, the console is saved, the console will be a return to the array once more.' object' inside.' cordova' -- network' information' this ' plugin' provides information' about' the device's network.' The following code, the following code, is to create one button in the index. Step 3 - Add The Listeners event We will add three listener events inside onDeviceReady features in index.js. One will listen to clicks on the button we created before, while the other two will listen to changes in connection status. document.getElementById addEventListener (click, networkInfo); document.addEventListener (offline, onOffline, false); document.addEventListener (online, onOnline, false); Step 4 - Creating a networkInfo feature feature will return information about your current connection to the network after clicking. We call the type method. Other features are onOffline and onOnline. These features are listened to by connection changes, and any change will trigger an alert message. NetworkInfo - var networkState - navigator.connection.type; var states -; Connection.UNKNOWN - Unknown connection Connection.ETHERNET - Ethernet connection Connection.CELL_3G; Connection.WIFIConnection.CELL_2G States; State (Connection.CELL_4G) - Link to 4G cell; It says: Connection. CELL - Total cell connection; Connection.NONE - No network connection Alert ('Connection type: ' When we launch an app connected to the network, onOnline will trigger an alert. If we press the INFO button, the alert will show the state of our network. If we disconnect from the network, OnOffline will be called. Cordova - Splash Screen This plug-in is used to display the screen splash when you launch an app. Step 1 - Installing the Splash Screen Plugin Splash plug-in can be installed in the team window, leaking the following code. C: Users of Username'Desktop-CordovaProject'gt;cordova-plugin-splashscreen Step 2 - Add Splash Screen Adding a Splash Screen is different from adding other Cordova plugins. We need to open config.xml and add the following snippets of code inside the widget element. The first clip of SplashScreen. It has a property value that is the image title in the platform/android/res/drawable-daddy. Cordova offers the default screen.png images we use in this example, but you'll probably want to add your own images. It is important to add images for the portrait and landscape view, as well as to cover different screen sizes. We have to add splashScreenDelay. We set the value to 3000 to hide the burst screen in three seconds. The last preference is optional. If the value is set to be true, the image will not stretch to match the screen. If it's false, it'll be stretched. Now that we're launching the app, we'll see a splash screen. Cordova - Vibration This plug-in is used to connect to the device's vibration function. Step 1 - Installing Vibration Plugin We can install this plug-in in the operational launch command window by turning off the following code: Users of Username'Desktop-CordovaProject'gt;cordov plugin add cordova-plugin-vibration Step 2 - Add buttons After installing the plug-in we can add buttons to index.html that will be used later to cause vibration. Step 3 - Add the Listeners event Now we're going to add an event of listeners inside onDeviceReady to index.js. document.getElementById (vibration). addEventListener (click, vibration); document.getElementById (vibrationPattern).addEventListener (click, vibrationPattern); Step 4 - Create features This plugin is very easy to use. We'll create two features. Vibration function - var time - 3000; navigator.vibrate - VibrationPattern function - var pattern (1000, 1000, 1000, 1000); navigator.vibrate The first function takes the time setting. This option is used to establish Vibration. The device will vibrate for three seconds as soon as we press the VIBRATION button. The second function uses a template setting. This array will ask the device to vibrate for one second and then wait one second then process again. Cordova - Whitelist This plug-in allows us to implement a white list policy to navigate the app. When we create a new Cordova project, the whitelist plug-in is installed and implemented by default. You can open the config.xml file to see the default settings provided by Cordova. Navigating the White List In the simple example below we allow links to some external URLs. This code is placed in config.xml. Navigation on file:// URLs is allowed by default. The Asterix sign is used to provide navigation with multiple values. In the example above, we allow navigation on all dungeons example.com. The same can be applied to the protocol or set-top box to the host. There is also a valid element of intent that is used to indicate which URLs are allowed to open the system. You can see in config.xml that Cordova has already allowed most of the necessary links for us. Network Request White List When you look inside the config.xml file, there is a zlt'access origin.'lt/access'gt; access'gt; element. This item allows all network requests into our app through Cordova hooks. If you want to only allow specific requests, you can remove it from config.xml and install it yourself. The same principle is used as in previous examples. This will allow you to make all network requests from . You can see the current security policy for your app inside the head element in index.html. <meta http-equiv=Content-Security-Policy content=default-src 'self' data: gap: 'unsafe-eval'; style-src 'self' 'unsafe-inline'; This is the default configuration. If you want to allow everything from the same origin and example.com, then you can use - zlt;meta http-equiv'Content-Security-Policy content'default-src 'self' foo.com'gt; you can also allow everything but limit CSS and JavaScript to the same origin. <meta http-equiv=Content-Security-Policy content=default-src *; style-src 'self' 'unsafe-inline'; Since this is a beginner's tutorial, we recommend Cordova's default options. Once you are familiar with Cordova, you can try some different values. Cordova - Best Practices Cordova is used to create hybrid mobile apps, so you need to consider this before choosing it for your project. Here are the best practices for developing Cordova apps. One-page apps are a recommended design for all Cordova applications. SPA uses router and navigation downloaded on one page (usually index.html). Routing is processed through AJAX. If you've followed our tutorials, you've probably noticed that almost every Cordova plugin has to wait until the device is ready before it can be used. Spa Spa will increase download speed and overall performance. Touch Events Since Cordova is used for the mobile world, it is natural to use touchstart and touchend events instead of clicks of events. Click events have 300ms delays, so clicks don't feel native. On the other hand, sensory events are not supported on every platform. You have to take this into account before deciding what to use. Animations You should always use hardware accelerated CSS transitions instead of JavaScript animations, as they will work better on mobile devices. Storage Use storage caching as much as possible. Mobile network connections are usually bad, so you should minimize network calls inside the app. You also have to handle the offline status of the app, as there will be times when the user's devices are offline. Scroll most of the time the first slow part inside the app will scroll through the lists. There are several ways to improve the performance of the app's scrolling. Our recommendation is to use native scrolling. When there are many items on the list, they should be downloaded in part. Use loaders if necessary. Images can also slow down the mobile app. You should use CSS image SPRITES whenever possible. Try to perfectly match the images rather than zooming in. CsS styles should avoid shadows and gradients as they slow down page rendering time. Simplifying the DOM browser is slow, so you should try to minimize dom manipulation and the number of DOM items. Testing ensures that the app is tested on as many devices and versions of the operating system as possible. If an app works flawlessly on one device, it doesn't necessarily mean it's working on some other device or platform. Platform. apache cordova tutorial tutorialspoint