LEARNING Cordova

#cordova

Table of Contents

About

Chapter 1: Getting started with Cordova

Remarks

Apache Cordova is used to create Mobile apps with HTML, CSS & JS.

Apache Cordova targets multiple platforms with one code base.

Apache Cordova is Free and open source.

Cordova wraps your HTML/JavaScript app into a native container which can access the device functions of several platforms. These functions are exposed via a unified JavaScript API, allowing you to easily write one set of code to target nearly every phone or tablet on the market today and publish to their app stores.

Who could use Apache Cordova

1. a mobile developer and want to extend an application across more than one platform, without having to re-implement it with each platform's language and tool set.

2. a web developer and want to deploy a web app that's packaged for distribution in various app store portals.

3. a mobile developer interested in mixing native application components with a WebView (special browser window) that can access device-level APIs, or if you want to develop a plugin interface between native and WebView components.

Introduction to Cordova: https://cordova.apache.org/docs/en/latest/

Examples

Installation or Setup

To install the **cordova** command-line tool, follow these steps:

1. Download and install Node.js. On installation you should be able to invoke **node** and **npm** on your command line.

   - To see if Node is installed, open your CLI (command line interface). For Windows it's the Windows Command Prompt, for MAC it's the Terminal. Type:

     $ node -v

     This should print a version number, so you'll see something like this v0.10.35. If Node is not installed find your OS and follow the instructions here: https://nodejs.org/en/download/package-manager/

2. (Optional) Download and install a git client, if you don't already have one. Following installation, you should be able to invoke **git** on your command line. The CLI uses it to download assets when they are referenced using a url to a git repo.

3. Install the **cordova** module using **npm** utility of Node.js. The **cordova** module will automatically be downloaded by the **npm** utility.

on OS X and Linux:

```
$ sudo npm install -g cordova
```

on Windows:

```
C:\>npm install -g cordova
```

The **-g** flag above tells **npm** to install **cordova** globally. Otherwise it will be installed in the **node_modules** subdirectory of the current working directory.

Following installation, you should be able to run **cordova** on the command line with no arguments and it should print help text.

Creating an application

Preliminary

Install the Cordova cli tools, if you haven't already.
```
$ npm install -g cordova
```

Navigate to your desired working folder.
```
$ cd /path/to/coding/folder
```

Creating the application

Create a new application
```
$ cordova create <appProjectName> <appNameSpace> <appName>
```

For this example, we'll create a 'HelloWorld' application:
```
$ cordova create helloWorld com.example.helloworld HelloWorld
```

Adding platforms and plugins

Platforms

First off, navigate to the application's folder.
```
$ cd <appName>
```

Add the platforms you wish to build for. The list of supported platforms can be found here.
```
$ cordova platform add <platformList>
```

We'll be adding the Android, iOS and browser platform. Use space separation to add multiple platforms at once. The `browser` platform will come in handy for in-browser testing.
Using the `--save` argument will save platform list to Cordova's `config.xml` file.

```
$ cordova platform add android ios browser --save
```

An extensive list of options regarding the `platform` command can be found in the cordova documentation.

Plugins

Cordova plugins can give you access to device hardware, OS specific functions and lots more features.
The structure of the `plugin` command is the same as the one for platforms

```
$ cordova plugin add <plugins.value>
```

We'll be adding the cordova file plugin (for easy device storage access) and the camera plugin, which gives access to the device's camera to make photos and videos.

```
$ cordova plugin add cordova-plugin-file cordova-plugin-camera --save
```

> **Remember**: using the `--save` argument writes your settings to the `config.xml` file. Very useful to easily recreate the project on another machine.

Cordova has an excellent plugin search page set-up for your convenience. You can find it here.

Running your application

Running the application is pretty straightforward. Simply use the following command.
```
$ cordova run <platform name>
```

For our example, we'll be running our test app in the browser.
```
$ cordova run browser
```

This open your default browser with your application ready for testing.

Install Cordova on Windows

First, Install Java SE Development Kit

This can be as simple as downloading, double-clicking on the downloaded file, and following the installation instructions. For install Java SE Development Kit download it from official web site.
Java SE Development Kit. Downloads

After JDK install is complete you need to add new `JAVA_HOME` system variable with path to your JDK

Next to the `PATH` system variable add path to bin dirrectory of JDK

Now you can test the install. Open Command Prompt and use command

```
javac -version
```

If you see a version number you did everything right!

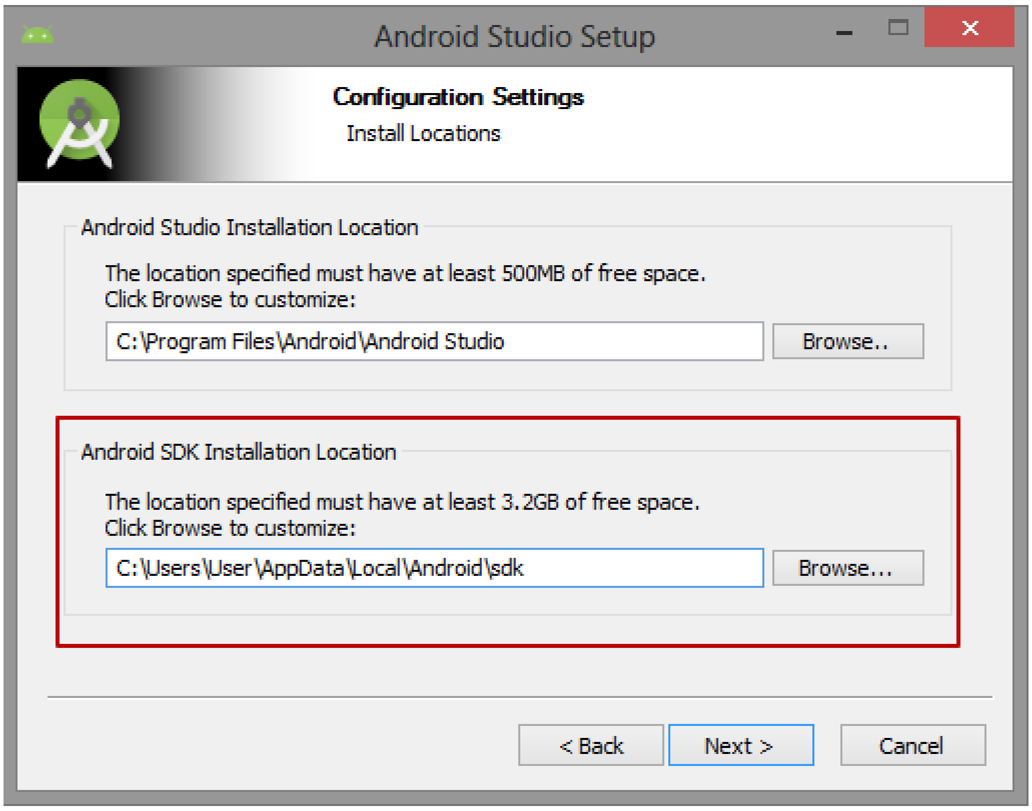
Now Install Android SDK Tools with Android Studio

I recomended install the Android Studio because at the moment it is the best way to quickly and easily install all the most necessary things for Android Development. The list of things includes:

- Android Development Kit (Android SDK, Android SDK Manager, Android SDK Platform-tools, Android SDK Build-tools)
- Android Emulator with a large number of Android configurations
- IDE (for Android Development on Java)
- Gradle
- It would be very helpful if you are learning Java, and in the future want to start developing for Android on Java

So, download Android Studio from official web site developer.android.com

Android Studio Installation is very simple and you just need to follow the instructions. But you should take note on Android SDK Installation Location


Android Studio Setup

Configuration Settings

Install Locations

Android Studio Installation Location

The location specified must have at least 500MB of free space.
Click Browse to customize:

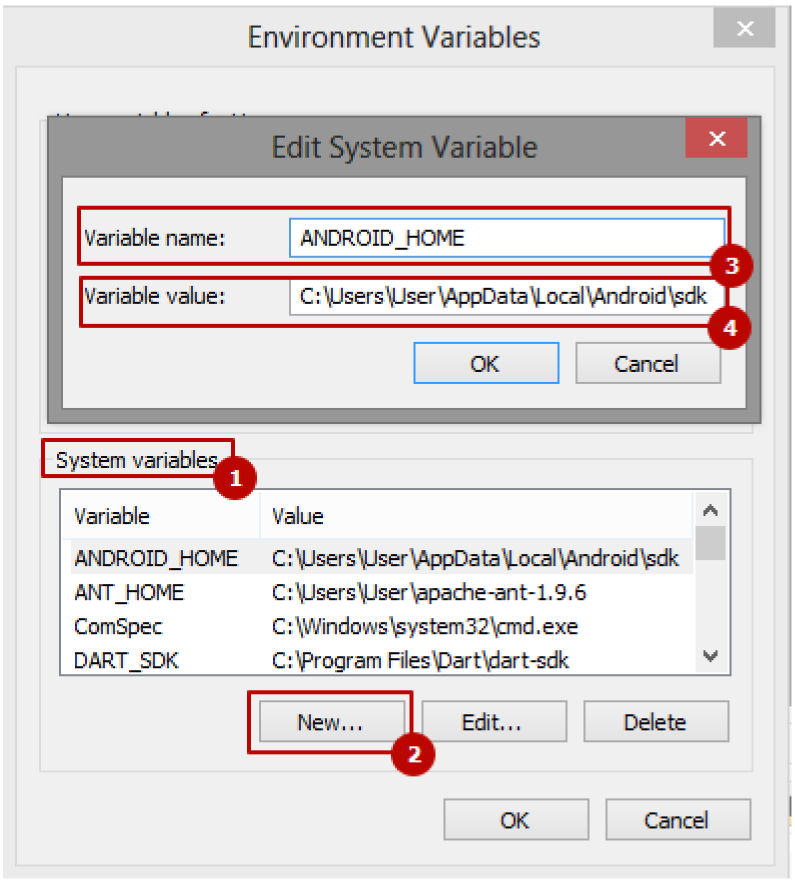
C:\Program Files\Android\Android Studio

Android SDK Installation Location

The location specified must have at least 3.2GB of free space.
Click Browse to customize:

C:\Users\User\AppData\Local\Android\sdk

After Android Studio installation is complete you need to add new `ANDROID_HOME` system variable with path to your `Android SDK`



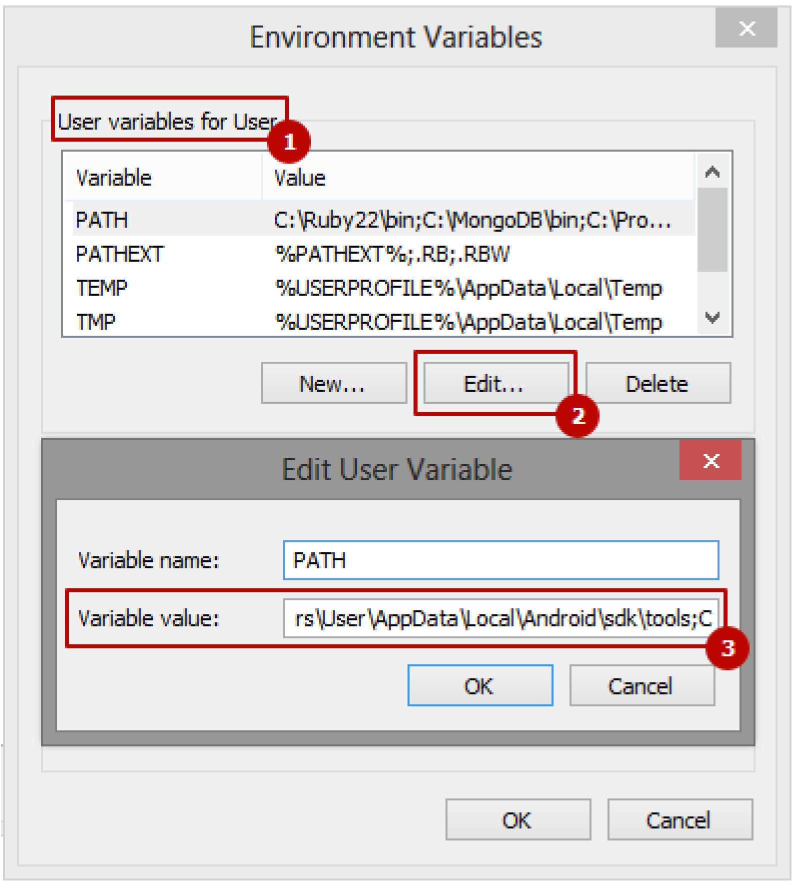
Variable name: ANDROID_HOME

Variable value: C:\Users\User\AppData\Local\Android\sdk

Variable	Value
ANDROID_HOME	C:\Users\User\AppData\Local\Android\sdk
ANT_HOME	C:\Users\User\apache-ant-1.9.6
ComSpec	C:\Windows\system32\cmd.exe
DART_SDK	C:\Program Files\Dart\dart-sdk

Now you need to add Android SDK and Android SDK Tools to PATH System Variable. In the list User variables select PATH and click the Edit button. At the end of the field Variable value, add a semicolon and follow paths:

```
C:\Users\User\AppData\Local\Android\sdk;C:\Users\User\AppData\Local\Android\sdk\tools;C:\Users\User\App
tools;
```


Environment Variables

User variables for User

Variable	Value
PATH	C:\Ruby22\bin;C:\MongoDB\bin;C:\Pro...
PATHEXT	%PATHEXT%;.RB;.RBW
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

Edit User Variable

Variable name: PATH

Variable value: rs\User\AppData\Local\Android\sdk\tools;C

Now you can test the install. Open Command Prompt and use command

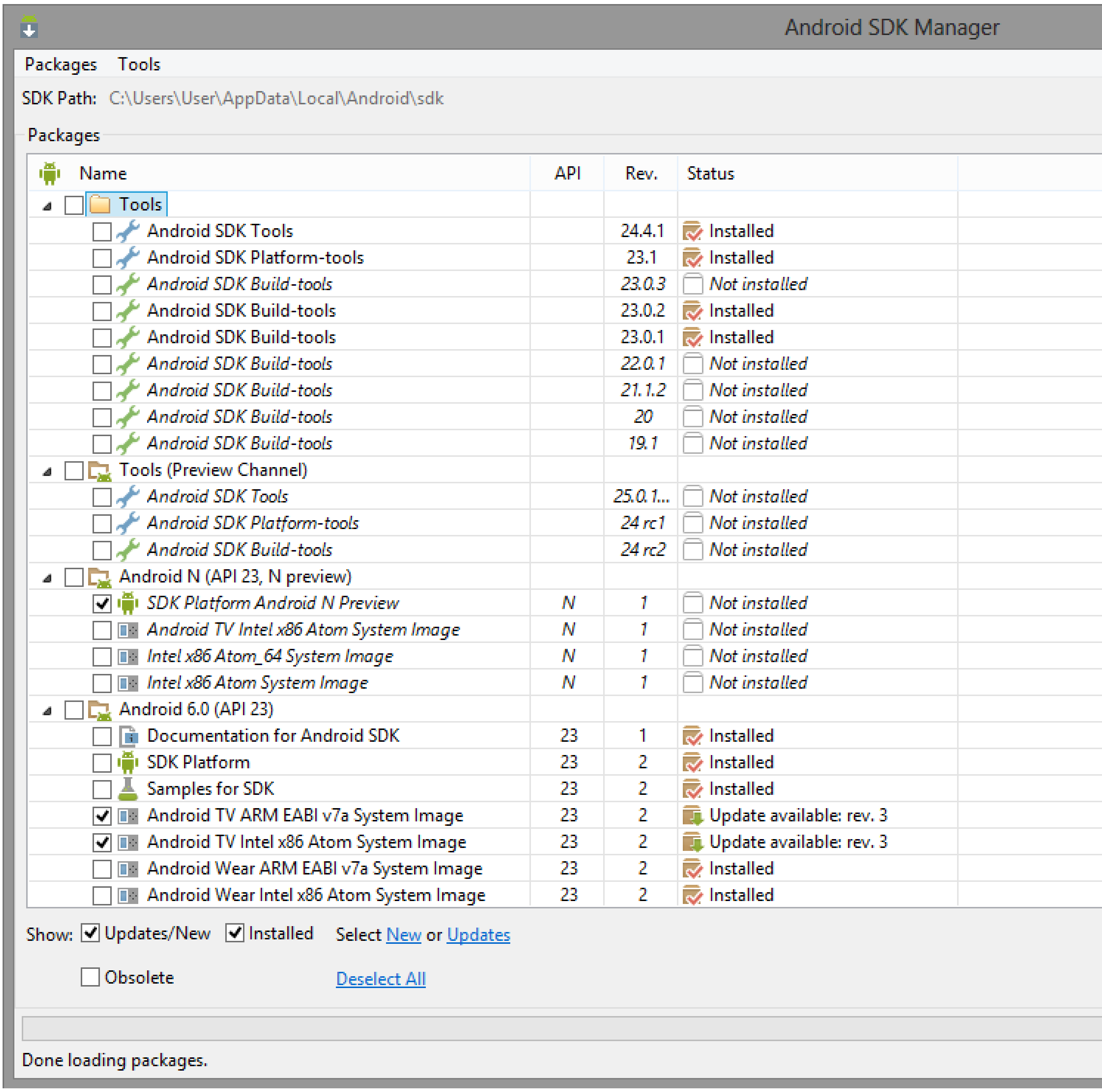
```
adb version
```

This should display the version of the Android Debug Bridge. If you see a version number you did everything right!

Now again open Command Prompt and use command

```
android
```

for open Android SDK Manager


Android SDK Manager

Packages Tools

SDK Path: C:\Users\User\AppData\Local\Android\sdk

Packages

	Name	API	Rev.	Status
☐	Tools			
☐	Android SDK Tools		24.4.1	Installed
☐	Android SDK Platform-tools		23.1	Installed
☐	*Android SDK Build-tools*		*23.0.3*	*Not installed*
☐	Android SDK Build-tools		23.0.2	Installed
☐	Android SDK Build-tools		23.0.1	Installed
☐	*Android SDK Build-tools*		*22.0.1*	*Not installed*
☐	*Android SDK Build-tools*		*21.1.2*	*Not installed*
☐	*Android SDK Build-tools*		*20*	*Not installed*
☐	*Android SDK Build-tools*		*19.1*	*Not installed*
☐	Tools (Preview Channel)			
☐	*Android SDK Tools*		*25.0.1...*	*Not installed*
☐	*Android SDK Platform-tools*		*24 rc1*	*Not installed*
☐	*Android SDK Build-tools*		*24 rc2*	*Not installed*
☐	Android N (API 23, N preview)			
✓	*SDK Platform Android N Preview*	*N*	*1*	*Not installed*
☐	*Android TV Intel x86 Atom System Image*	*N*	*1*	*Not installed*
☐	*Intel x86 Atom_64 System Image*	*N*	*1*	*Not installed*
☐	*Intel x86 Atom System Image*	*N*	*1*	*Not installed*
☐	Android 6.0 (API 23)			
☐	Documentation for Android SDK	23	1	Installed
☐	SDK Platform	23	2	Installed
☐	Samples for SDK	23	2	Installed
✓	Android TV ARM EABI v7a System Image	23	2	Update available: rev. 3
✓	Android TV Intel x86 Atom System Image	23	2	Update available: rev. 3
☐	Android Wear ARM EABI v7a System Image	23	2	Installed
☐	Android Wear Intel x86 Atom System Image	23	2	Installed

Show: ✓ Updates/New ✓ Installed Select New or Updates

☐ Obsolete Deselect All

Done loading packages.

In the Android SDK Manager select to install

- Android SDK Tools
- Android SDK Platform-tools
- Android SDK Build-tools
- Android SDK Build-tools
- Android 6.0 (API 23)
- Android 5.1.1 (API 22)
- Android 5.0.1 (API 21)
- Android 4.2.2 (API 17)
- GPU Debugging tools

- Android Support Repository
- Android Support Library
- Google Play services
- Google Repository
- Google USB Driver
- Intel x86 Emulator Accelerator (HAXM installer)

and click Install button.

Note:

Cordova Android Supported API Levels

Understanding Android API Levels

Android Platform/API Version Distribution

Install Cordova

Open Command Prompt and install Cordova using command

```
npm install -g cordova
```

Versions

Latest Cordova version:

Cordova 6.1.0 - https://cordova.apache.org/news/2016/03/23/tools-release.html Cordova 6.0.0 - https://cordova.apache.org/news/2016/01/28/tools-release.html

Latest Android platform and iOS plaatform

Cordova Android 5.2.2 - https://cordova.apache.org/announcements/2016/07/02/android-5.2.0.html Cordova iOS 4.2.1 - https://cordova.apache.org/announcements/2016/07/11/cordova-android-5.2.1.html

Read Getting started with Cordova online: https://riptutorial.com/cordova/topic/884/getting-started-with-cordova

Chapter 2: Cordova Crop Image Plugin

Examples

Crop Image after clicking using camera or selecting image.

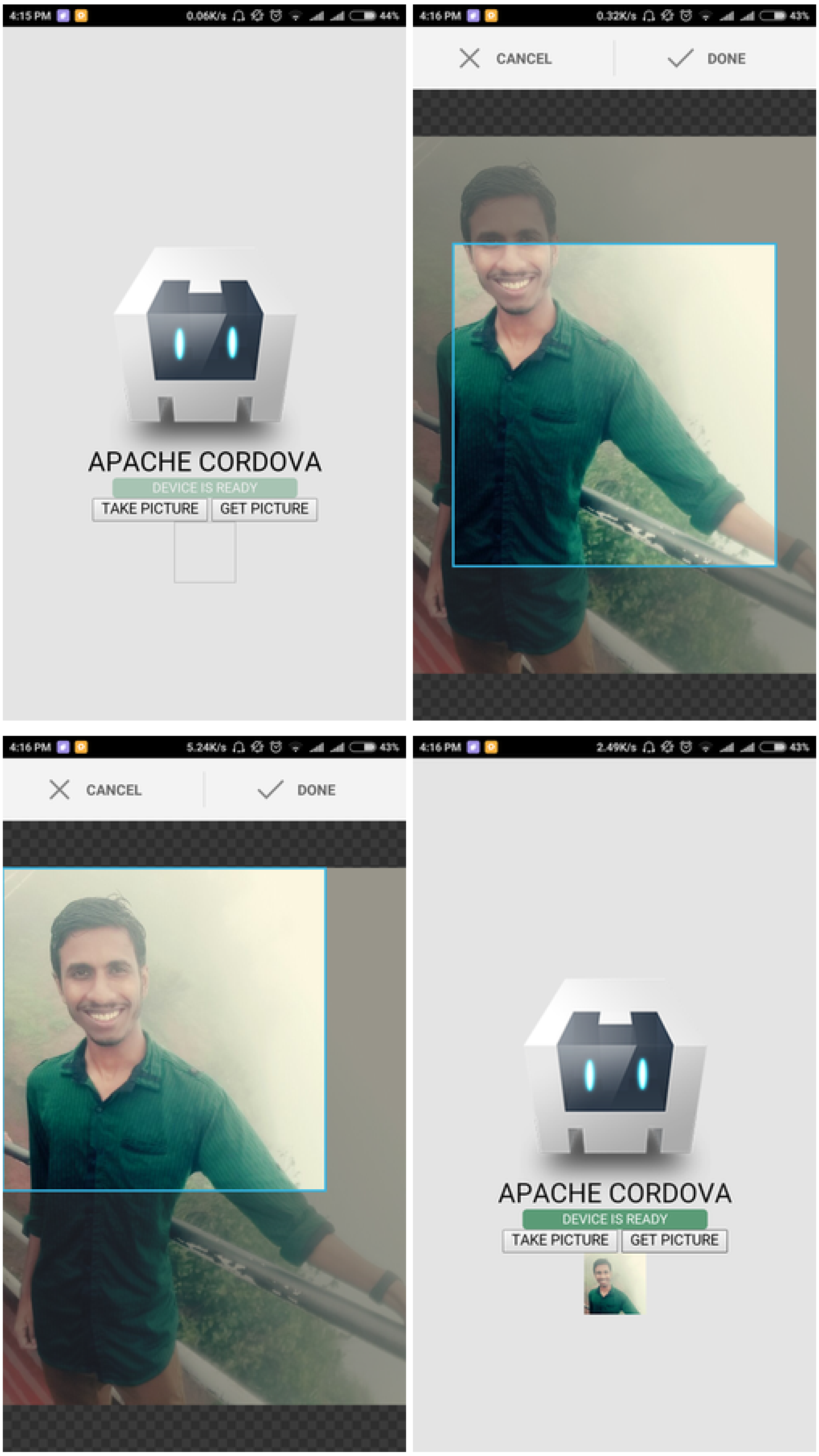
It crops the images in square shape.

This cordova project uses two plugins:

1. Cordova Camera Plugin -- https://github.com/apache/cordova-plugin-camera
2. Cordova Crop Image Plugin -- https://github.com/jeduan/cordova-plugin-crop

The Camera plugin is combined with the Crop Image Plugin by putting the Cop Image Plugin Code within the success callback of Camera Plugin Code.

```
  /*Camera Plugin Code*/
  navigator.camera.getPicture(onSuccess, onFail, {
      quality: 50,
      destinationType: Camera.DestinationType.FILE_URI
  });

  function onSuccess(imageData) {
       console.log(imageData);

       /*Crop Image Plugin Code*/
       plugins.crop(function success (data) {
          console.log(data);
          var image = document.getElementById('myImage');
          image.src = data;
       },
       function fail () {

       }, imageData, {quality:100});
  }

 function onFail(message) {
    alert('Failed because: ' + message);
 }
```



Read Cordova Crop Image Plugin online: https://riptutorial.com/cordova/topic/7078/cordova-crop-image-plugin

Chapter 3: Cordova ios build

Introduction

Hope you are familiar with the basics of cordova. Let's try to build a cordova ios build, ios build is bit different as that of android build, we need mac machine to perform this task. No can also prepare the ios build online, but to test and debug your application on a mac simulator you have to have the mac machine with you.

let's start with an example.

Remarks

you face any problem during preparing the ios build please reach to me, will try to assist you.

Examples

cordova HelloWorld Project

To prepare the ios build first we need to create the cordova project. lets create the project by the command line tool.

```
  cordova create hello com.example.hello "HelloWorld"
```

Go to the project dir by `cd hello`.

we are now in the project directory, lets check which platforms are available to us

```
cordova platform ls
```

As, we need to prepare the ios build, we will add ios platform to the project.

```
cordova platform add ios@version    => put the desired version you want to add.
cordova platform add ios@latest.    => add the latest version available.
```

Now we need **Xcode** on mac machine to prepare the ios build.

Download the latest Xcode available to us via app store. Please ignore the beta versions of xcode as they have compatibility issue sometime.

After Xcode installation install the Xcode cli using terminal by the command below.

```
xcode-select --install
```

It will popup the window, please follow the instructions and install the cli properly. Now we just need one more tool to deploy the ios build, install it as well.

```
npm install -g ios-deploy
```

during the installation of deployment tool if you get the permission denied error then please try with the sudo command.

```
sudo npm install -g ios-deploy
```

lets prepare the ios build using the commands

```
cordova platform add ios
```

if you didnt add the platform before, add it.

```
cordova prepare => it will move the all required files to platform folder and create a
.xcworkspace file in the platform.ios folder.

cordova build ios => to build the ios application.
```

But wait, we haven't run the build on simulator, lets do it also.

you have two ways to do it

1. using cli
2. using Xcode

first lets do it with CLI.

```
corodva run ios => it will run the application on the default simulator available.
```

if you want to play with the simulators then please explore the `cordova emulator` command

using Xcode

jump the project folder and move into `/platform/ios folder` of your project. open the file .xcworkspce using command `open <ProjectName>.xcworkspcae` eg `open MyApp.xcworkspcace.`

it will redirect you to the Xcode window, there on the window top left you can see your project, click on the project, and on top header you can see the run button, click it and play with your app.

Thank you.

Read Cordova ios build online: https://riptutorial.com/cordova/topic/10666/cordova-ios-build

Chapter 4: Cordova plugins: how to install, how they work, examples

Examples

What are cordova plugins?

Cordova plugins are in simple words a layer on top of the respective native platform.

Plugins provide an interface between to access the native platform.

How can Cordova plugins be useful?

Cordova Plugins provide a common interface to interact with the native code.

Each plugin has an intermediary JavaScript file that provides access to platform specific features.

Installing Cordova Plugin

```
cordova plugin add <plugin-name>
```

Example: cordova plugin add cordova-plugin-camera

The plugin should be installed in the root directory of the project.

Note:

> Before adding the plugin replace the platform specific www folder content with the outer www folder in the root directory. This is because on adding a plugin the contents of the outer www folder are replaced with the platform specific www folder.

Note:

> When you add a new platform and if you have any installed plugins on your project is not necessary to install the existing plugins. Cordova automatically will add the installed plugins for the new platform.

Most Popular Plugins

1. **cordova-plugin-battery-status** - used for monitoring device's battery status.

2. **cordova-plugin-camera** - provides an API for taking pictures and for choosing images from the system's image library.

3. **cordova-plugin-contacts** - provides access to the device contacts database.

4. **cordova-plugin-device** - describes the device's hardware and software.
5. **cordova-plugin-device-motion** - access to the device's accelerometer.
6. **cordova-plugin-file** - implements a File API allowing read/write access to files residing on the device.
7. **cordova-plugin-geolocation** - provides information about the device's location, such as latitude and longitude.
8. **cordova-plugin-globalization** - obtains information and performs operations specific to the user's locale, language, and timezone.
9. **cordova-plugin-inappbrowser** - show helpful articles, videos, and web resources inside of your app. Users can view web pages without leaving your app.
10. **cordova-plugin-network-information** - provides information about the device's cellular and wifi connection, and whether the device has an internet connection.
11. **cordova-plugin-vibration** - provides a way to vibrate the device.
12. **cordova-plugin-statusbar** - provides some functions to customize the iOS and Android StatusBar.
13. **cordova-plugin-whitelist** - implements a whitelist policy for navigating the application webview on Cordova 4.0. **Recommended plugin!**

For cordova specific plugins follow below link https://cordova.apache.org/plugins/

Read Cordova plugins: how to install, how they work, examples online: https://riptutorial.com/cordova/topic/7077/cordova-plugins--how-to-install--how-they-work--examples

Chapter 5: Creating Your First Application With Cordova

Remarks

In case running `cordova run android` fails. Make sure that your Android device is connected to your computer and run `adb devices` to make sure the Android Development Tools (ADT) can detect your device.

Examples

Using the command-line tool

First you create a new Cordova project:

```
cordova create HelloWorld my.application.identifier AppName
```

This will create a blank Cordova project

- in the *HelloWorld* folder
- with identifier *my.application.identifier* (which should be unique for each application)
- with name *AppName*.

Next you add the desired platforms:

```
cordova platform add android
// and/or
cordova platform add browser
// and/or
cordova platform add ios    // On macOS only
// etc…
```

Build your application to generate executable file:

```
cordova build                   // Build project for all platforms
cordova build ios               // Build project only for iOS platform
cordova build android           // Build project only for Android platform
```

Once built, you can run the app on one of the platforms you added:

```
cordova run android --emulator  // Run Android app in emulator
cordova run android --device    // Run Android app on physical connected device
cordova run browser             // Will run the app in the browser
```

If you want to build the application for Eclipse, Xcode, Visual Strudio, etc:

```
cordova prepare [platform_name] // Prepare copies of www folder and any plugins into the
appropriate platform folder
```

Read Creating Your First Application With Cordova online:
https://riptutorial.com/cordova/topic/2396/creating-your-first-application-with-cordova

Chapter 6: Debugging the application

Remarks

Important thing to remember when debugging cordova apps, if you have an OnDeviceReady event and code that executes there, by the time the app launches, your debugger will still not be attached(unlike say Visual Studio C# debugging where application waits for the debug process to attach before continuing with launching the program).

This means that any initial set up console messages or breakpoints will not be captured.

Solution for this can be a delayed set up or delayed console logging with setTimeout when DeviceReady event is fired.

Examples

Debug on Android Device using USB

A Cordova application runs as a website on a WebView component within the native mobile platform. Debugging a cordova application can therefore be done by utilizing your favourite browsers development tools. The following steps are needed to hook the application, running on the device, to the Chrome browser on a development machine:

1. Enable USB Debugging on your Device (you can follow this guide)
2. Install the Android Debug Bridge `adb` (not required on recent versions of Chrome) (guide for OSX)
3. Connect your phone and execute `adb devices` in your terminal (not required on recent versions of Chrome), and select `ok` in the popup on your phone `Allow USB debugging?`.
4. Open Chrome
5. Browse to `chrome://inspect`, or choose More tools => Inspect Devices...
6. Select your device and debug using Chrome's developer tools

Depending on your device you may need to download USB drivers first.

You also need to enable 'unknown sources' under Security in Settings if you want to load the app on to your phone.

Debug Cordova apps using GapDebug

https://www.genuitec.com/products/gapdebug/

GapDebug is a comprehensive mobile debugging tool that bridges the gap left by other debugging options. Operating on both the Windows and Mac platforms, GapDebug allows debugging of hybrid mobile apps, such as PhoneGap and Cordova, on modern iOS and Android devices. And, GapDebug is always free for local debugging.

Step for first time configuration are given in this following link:

https://www.genuitec.com/products/gapdebug/learning-center/configuration/

Debug on iOS device using USB

1. **Disable Private Browsing**

Open your device's Safari settings and ensure that **Private Browsing is turned off**. Remote debugging will not work if *Private Browsing* is enabled.

2. **Enable Web Inspector**

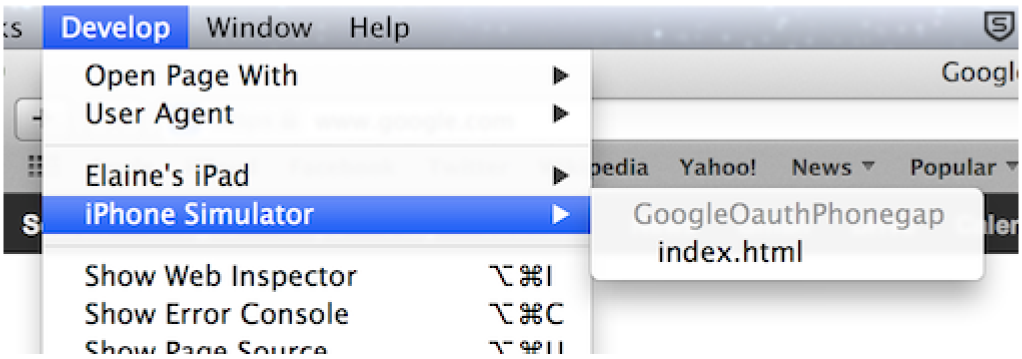
Tap the *Advanced* tab on your device's Safari settings and ensure that **Web Inspector is turned on**.

3. **Enable Safari's Develop Menu**

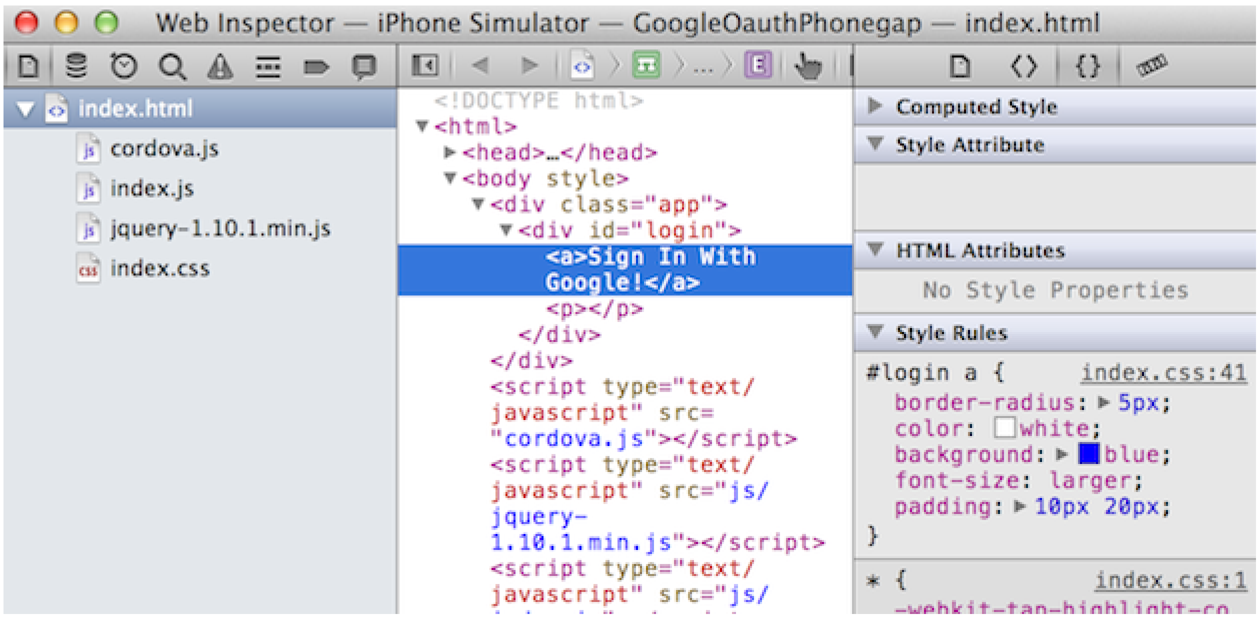
On your desktop or laptop, open Safari's Preferences and click on the Advanced tab. Check the box to **Show Develop menu in menu bar**.

4. **Start Web Inspector**

Launch your app either in the iOS simulator or on a physical device. If you are using a physical device you'll need to connect it to your desktop or laptop with the standard USB cable. Once the app has launched, switch to Safari, select the **Develop** menu item, then find the entry corresponding to the web page you want to debug.



Now you can use web inspector just like you would to debug a web page.



```
<!DOCTYPE html>
▼<html>
  ▶<head>…</head>
  ▼<body style>
    ▼<div class="app">
      ▼<div id="login">
          <a>Sign In With
          Google!</a>
          <p></p>
        </div>
      </div>
      <script type="text/
      javascript" src=
      "cordova.js"></script>
      <script type="text/
      javascript" src="js/
      jquery-
      1.10.1.min.js"></script>
      <script type="text/
      javascript" src="js/
```

```
#login a {          index.css:41
  border-radius: ▶ 5px;
  color: □white;
  background: ▶ ■blue;
  font-size: larger;
  padding: ▶ 10px 20px;
}
* {                 index.css:1
  -webkit-tap-highlight-co
```

Read Debugging the application online: https://riptutorial.com/cordova/topic/4004/debugging-the-application

Chapter 7: Firebase Push Notification Cordova

Examples

Firebase Push Notification in Cordova Android

Add Firebase to Your Android Project

Add Firebase to your app

To add Firebase to your app you'll need a Firebase project and a Firebase configuration file for your app.

1. Create a Firebase project in the Firebase console, if you don't already have one. If you already have an existing Google project associated with your mobile app, click Import Google Project. Otherwise, click Create New Project.
2. Click Add Firebase to your Android app . If you're importing an existing Google project, this may happen automatically and you can just download the config file.
3. When prompted, enter your app's package name. It's important to enter the package name your app is using; this can only be set
   when you add an app to your Firebase project.
4. At the end, you'll download a google-services.json file. You can download this file again at any time. If you haven't done so already, copy this into your project's module folder, typically app/.

Cordova Firebase Push Notification Plugin

https://www.npmjs.com/package/cordova-plugin-fcm

For obtaining the access token:

```
FCMPlugin.getToken(
  function(token){
    alert(token);
  },
  function(err){
    console.log('error retrieving token: ' + err);
  }
);
```

Callback for receiving push notification:

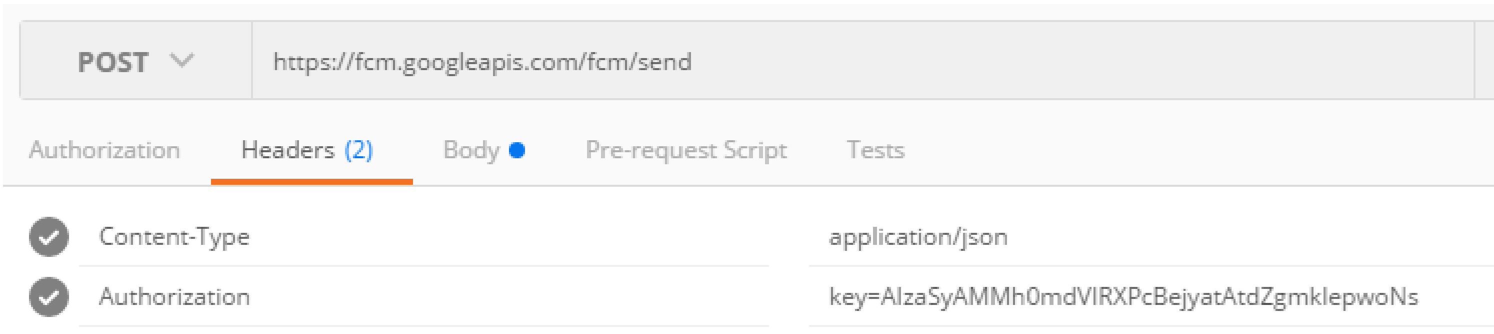
```
FCMPlugin.onNotification(
  function(data){
    if(data.wasTapped){
      //Notification was received on device tray and tapped by the user.
      alert( JSON.stringify(data) );
    }else{
      //Notification was received in foreground. Maybe the user needs to be notified.
      alert( JSON.stringify(data) );
    }
  },
  function(msg){
    console.log('onNotification callback successfully registered: ' + msg);
  },
  function(err){
    console.log('Error registering onNotification callback: ' + err);
  }
);
```

Place the get access token and callback for receiving push notification inside index.js file within receivedEvent function

Sending Push Notification via REST API

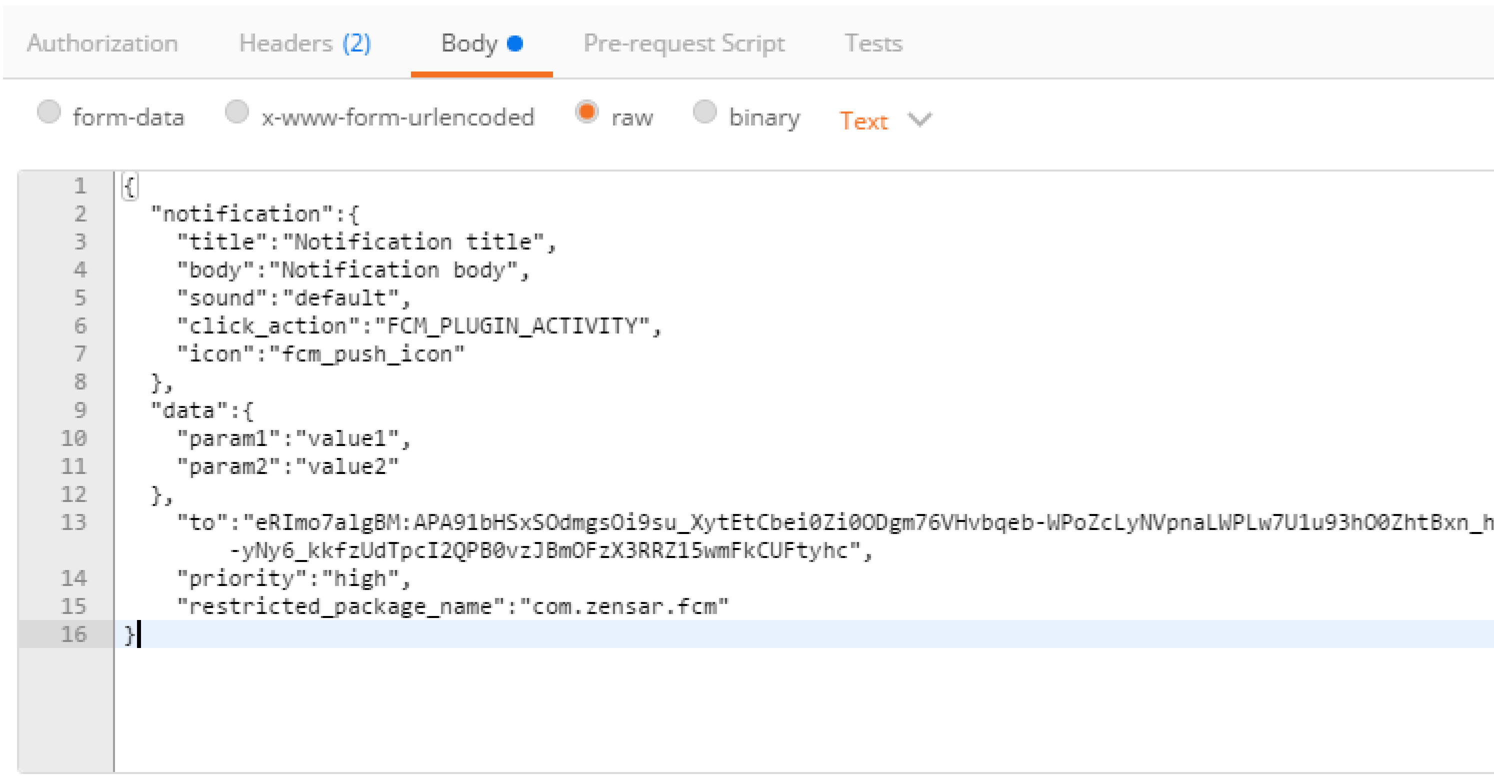
```
    //POST: https://fcm.googleapis.com/fcm/send
    //HEADER: Content-Type: application/json
    //HEADER: Authorization: key=AIzaSyAMMh0mdVIRXPcBejyatAtdZgmklepwoNs //key is server-key
    {
      "notification":{
        "title":"Notification title",  //Any value
        "body":"Notification body",  //Any value
        "sound":"default", //If you want notification sound
        "click_action":"FCM_PLUGIN_ACTIVITY",  //Must be present for Android
        "icon":"fcm_push_icon"  //White icon Android resource
      },
      "data":{
        "param1":"value1", /Any data to be retrieved in the notification callback
        "param2":"value2"
      },
        "to":"eRImo7algBM:APA91bHSxSOdmgsOi9su_XytEtCbei0Zi0ODgm76VHvbqeb-
WPoZcLyNVpnaLWPLw7U1u93hO0ZhtBxn_hVGxPAwxXXfc-yNy6_kkfzUdTpcI2QPB0vzJBmOFzX3RRZ15wmFkCUFtyhc",
//Topic or single device
        "priority":"high", //If not set, notification won't be delivered on completely closed
iOS app
        "restricted_package_name":"com.zensar.fcm" //Optional. Set for application filtering
    }
```

Configure the above REST API using Postman rest client.


POST https://fcm.googleapis.com/fcm/send

Authorization | Headers (2) | Body | Pre-request Script | Tests

Content-Type	application/json
Authorization	key=AIzaSyAMMh0mdVIRXPcBejyatAtdZgmklepwoNs



```
{
  "notification":{
    "title":"Notification title",
    "body":"Notification body",
    "sound":"default",
    "click_action":"FCM_PLUGIN_ACTIVITY",
    "icon":"fcm_push_icon"
  },
  "data":{
    "param1":"value1",
    "param2":"value2"
  },
    "to":"eRImo7algBM:APA91bHSxSOdmgsOi9su_XytEtCbei0Zi0ODgm76VHvbqeb-WPoZcLyNVpnaLWPLw7U1u93hO0ZhtBxn_h
        -yNy6_kkfzUdTpcI2QPB0vzJBmOFzX3RRZ15wmFkCUFtyhc",
    "priority":"high",
    "restricted_package_name":"com.zensar.fcm"
}
```

How it works *Send a push notification to a single device or topic.*

1.a Application is in foreground: The user receives the notification message in its device notification bar. The user taps the notification and the application is opened. The user receives the notification data in the JavaScript callback'.

1.b Application is in background: The user receives the notification message in its device notification bar. The user taps the notification and the application is opened. The user receives the notification data in the JavaScript callback'.

Read Firebase Push Notification Cordova online: https://riptutorial.com/cordova/topic/6892/firebase-push-notification-cordova

Chapter 8: Getting started with Cordova

Introduction

Mobile App Development using Cordova

Apache Cordova is an open-source mobile development framework. It allows you to use standard web technologies - HTML5, CSS3, and JavaScript for cross-platform development.

Examples

Creating Android build (.apk)

Install cordova using the following command **npm install -g cordova**.

Use cordova -version to check the **cordova version**.

Set path variables ANDROID_HOME and JAVA_HOME.

Example:

```
export ANDROID_HOME = /home/geethu/android-sdk-linux

export PATH = $PATH:$ANDROID_HOME/tools:$ANDROID_HOME/platform-tools

export JAVA_HOME = /usr/lib/jvm/java-7-openjdk-amd64

export PATH = $PATH:$JAVA_HOME/bin
```

Create a blank Cordova project using the command-line tool. Navigate to the directory where you wish to create your project and type.

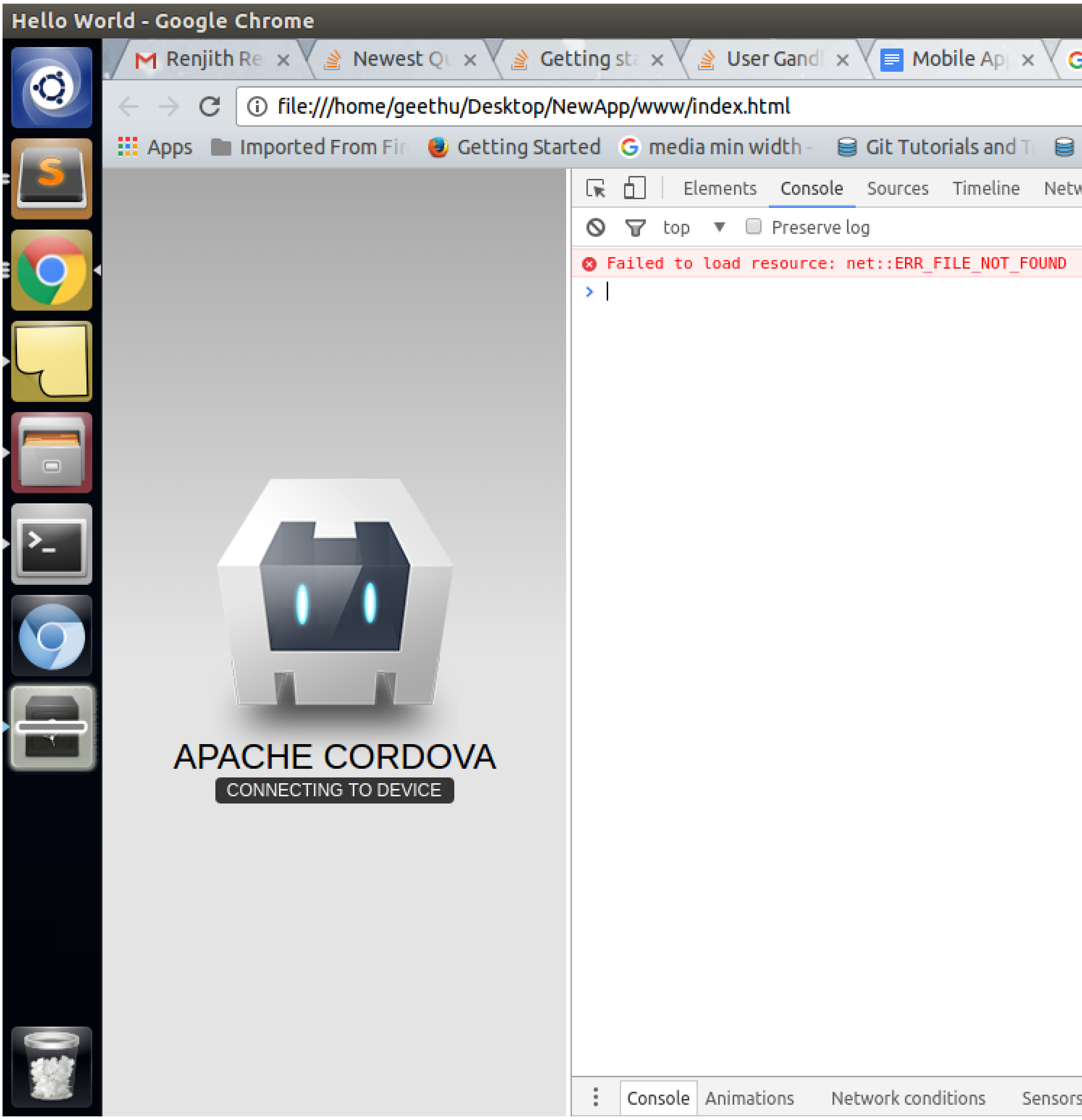
cordova create workshop com.yourname.workshop Workshop

Cordova CLI, create a Cordova project named Workshop in a directory named workshop.

Navigate to the project directory cd workshop.

Add Android platform as **cordova platform add android**.

[



```
Failed to load resource: net::ERR_FILE_NOT_FOUND
```

]1

Add these script files to your index.html in the following order.

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

To change the app name and icon edit the config.xml file in www folder.

To build the project in the workshop/platforms/android folder

cordova build android

Run it on an Android device connected to your computer using a USB cable, type:

cordova run android

---

1. **> Cordova .apk installation on device**

In addition, I would like to add some more detail about How you can install .apk file in your android mobile, while deploying your application on the server.

```
cordova plugin add cordova-plugin-mfp
```

run the above command, then try to add the server on which you want to deploy the application.

```
mfpdev server add
```

then follow the instructions by CLI. run the command below to change your configuration settings.

```
mfpdev app config server <profile Name of server>

cordova build
```

it will create .apk file in the

/platforms/android folder

download the apk in your mobile, install it and play with your app.

Read Getting started with Cordova online: https://riptutorial.com/cordova/topic/9287/getting-started-with-cordova

Chapter 9: Google Analytics in Cordova

Examples

Google Analytics in cordova without any plugin.

Insert the analytics function within index.js

```
    function analytics(){

        (function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
        (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
        m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
        })(window,document,'script','lib/analytics.js','ga');

        if(window.localStorage) {

            ga('create', 'UA-XXXXXXX-1', {
              'storage': 'none'
              , 'clientId': window.localStorage.getItem('ga_clientId')  /*The tracker id
obtained from local storage*/
            });
            ga(function(tracker) {
              window.localStorage.setItem('ga_clientId', tracker.get('clientId'));
             /*The tracker id for each device is different and stored in local storage*/
            });
        }
        else {

            ga('create', 'UA-XXXXXXX-1', 'auto');
        }

    }
```

Insert each below script tags in each html page and modify the page name

```
    <script>
        (function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
        (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
        m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
        })(window,document,'script','lib/analytics.js','ga');

        ga('set','checkProtocolTask',null);
        /*checkProtocal Task is set to null so that GA allows tracking other than http/https
*/

        ga('set', 'page', "Page Name");
        /*Page Name is name of each html page*/

        ga('send', 'pageview');
    </script>
```

Read Google Analytics in Cordova online: https://riptutorial.com/cordova/topic/6909/google-analytics-in-cordova

Chapter 10: How to customize platform specific www folder in cordova

Examples

create css/js specific to a platform (android/ios)

Let say you want to create css/js file specific to a platform. For that you have to create a **merges** folder in root folder of you cordova porject. In merges folder create directory for each platform (android/ios..). then in specific platform folder create a css/js folder and put your css/js file specific to platform folder. That's it, once you run **cordova build** command, all js/css files corresponding to each platform would be placed in respected platform folder

Note: Make sure your root www/index.html will have same css/js defined. For that make sure you have same filename corresponding to each platform in merges folder.

```
//let say you are in CordovaMergesExample folder
cd CordovaMergesExample

//create test folder with com.test id and TestApp as name
cordova create test com.test TestApp

//add platform android and ios
cordova platform add android ios


----------


//create merges/android/css/override.css and merges/ios/css/override.css


----------

//In root www/index.html add this stylesheet
<link rel="stylesheet" type="text/css" href="css/override.css" />


----------


cordova build

---> cordova build engine automatically identify the platform in merges folder and add files
in respective folders. Check platforms/android/assets/www/css and platforms/ios/www/css
```

Read How to customize platform specific www folder in cordova online: https://riptutorial.com/cordova/topic/7498/how-to-customize-platform-specific-www-folder-in-cordova

Chapter 11: How to detect the state of the network connection

Examples

Using the cordova-plugin-network-information plugin

Detecting the current state of the network connection and responding to any changes that might occur, can be done by using one of several plugins. This example is about the cordova-plugin-network-information plugin.

Add the plugin to the project:

```
cordova plugin add cordova-plugin-network-information
```

After the Cordova deviceready event a connection object is available through `navigator.connection`. The `type` property contains the current network state:

```
document.addEventListener("deviceready", function() {
    var networkState = navigator.connection.type;
}, false);
```

`networkState` now contains one of the following constants:

```
Connection.UNKNOWN  //  Unknown connection
Connection.ETHERNET //  Ethernet connection
Connection.WIFI     //  WiFi connection
Connection.CELL_2G  //  Cell 2G connection
Connection.CELL_3G  //  Cell 3G connection
Connection.CELL_4G  //  Cell 4G connection
Connection.CELL     //  Cell generic connection
Connection.NONE     //  No network connection
```

Detecting a change in network connection can be done by hooking a function to either the `online` or the `offline` event:

```
document.addEventListener("online", function() {
    // device went online
    var networkState = navigator.connection.type; // Get new network state
    ...
}, false);

document.addEventListener("offline", function() {
    // device went offline
    var networkState = navigator.connection.type; // Get new network state
    ...
}, false);
```

Read How to detect the state of the network connection online:
https://riptutorial.com/cordova/topic/3615/how-to-detect-the-state-of-the-network-connection

Chapter 12: How to install/uninstall custom cordova plugin

Examples

Use plugman command to install/uninstall cordova plugin

You can use plugman command to install/uninstall custom cordova plugins.

To install plugman

```
npm install -g plugman
```

Install plugin command syntax:

```
plugman <install|uninstall> --platform <ios|android|blackberry10|wp8> --project <directory> --
plugin <name|url|path>
```

Example:

```
plugman install --platform ios --project platforms/ios/ --plugin plugins/cordova-plugin-test/
```

For more plugman options check this link

Read How to install/uninstall custom cordova plugin online:
https://riptutorial.com/cordova/topic/7506/how-to-install-uninstall-custom-cordova-plugin

Chapter 13: Make application released from Cordova CLI

Examples

Android

Step 1: Go to root directory of project and open command line prompt

```
 cordova build --release android
```

This generates an unsigned apk under \platforms\android\build\outputs\apk with the name

android-release-unsigned.apk

Step 2: Key generation for obtaining signed apk

Syntax:

```
 keytool -genkey -v -keystore <keystoreName>.keystore -alias <Keystore AliasName> -keyalg <Key
algorithm> -keysize <Key size> -validity <Key Validity in Days>
```

Example:

```
keytool -genkey -v -keystore ExampleApp.keystore -alias TestExampleApp -keyalg RSA -keysize
2048 -validity 10000


keystore password? : xxxxxxx
What is your first and last name? :  xxxxxx
What is the name of your organizational unit? :  xxxxxxxx
What is the name of your organization? :  xxxxxxxxx
What is the name of your City or Locality? :  xxxxxxx
What is the name of your State or Province? :  xxxxx
What is the two-letter country code for this unit? :  xxx
```

The keystore is generated in the same folder with the name ExampleApp.keystore

Step 3: Move the generated keystore to ***\platforms\android\build\outputs\apk***

Run the jarsigner tool in the command prompt under ***\platforms\android\build\outputs\apk***

Syntax:

```
jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore <keystorename <Unsigned APK
file> <Keystore Alias name>
```

Example:

```
jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore ExampleApp.keystore android-
release-unsigned.apk TestExampleApp
```

This generates the signed apk with the same name.

Step 4: zip align tool to optimize the APK

```
 zipalign -v 4 android-release-unsigned.apk android.apk
```

The zipalign is located under \Android\sdk\build-tools\23.0.3\zipalign

This generates a signed apk with the name android.apk which can now be uploaded to app store

iOS

Step 1: Create a build.json file in the root directory of the project.

Sample of build.json

```
{
  "ios": {
    "debug": {
      "codeSignIdentity": "iPhone Developer",
      "provisioningProfile": "your-developer-provisioning-profile-UUID-here"
    },
    "release": {
      "codeSignIdentity": "iPhone Distribution",
      "provisioningProfile": "your-distribution-provisioning-profile-UUID-here"
    }
  }
}
```

> Note: The UUID can be obtained by opening the .mobileprovision file on a text editor and search for 'UUID'.

Step 2: Run the following command from the root folder of the project on the terminal

```
cordova build ios --device --release
```

Read Make application released from Cordova CLI online:
https://riptutorial.com/cordova/topic/7092/make-application-released-from-cordova-cli

Chapter 14: Push Notification in Android and iOS

Examples

Using the new phonegap-plugin-push

For the purpose of sending push notifications to cordova apps. The first step is to obtain a device token. A "device token" is specific to each device and each project.

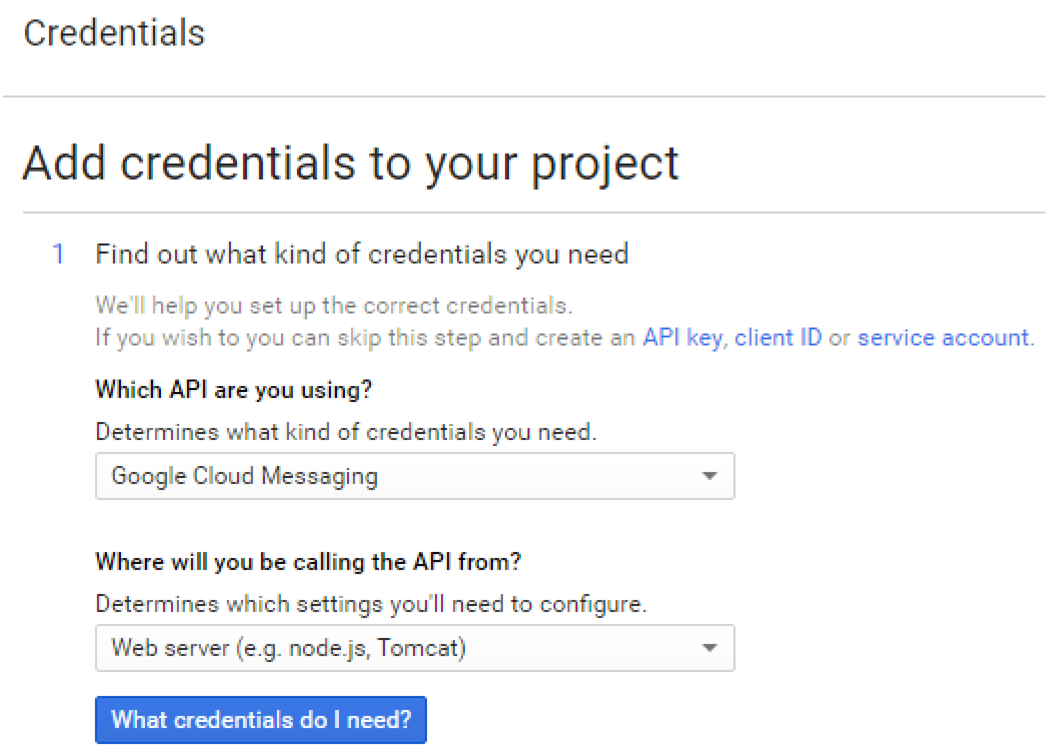
Pre-requisite:

1. Google Cloud Messaging Project Number

For this go to Google Developer Console and create a new project.
Under Project Information is the Project Number

2. Google Cloud Messaging API Key for above Project (needed for server)

Go to Library -> Google Cloud Messaging -> Enable. Go to Credentials to create an API key of Type server.



Credentials

Add credentials to your project

1 Find out what kind of credentials you need

We'll help you set up the correct credentials.
If you wish to you can skip this step and create an API key, client ID or service account.

Which API are you using?

Determines what kind of credentials you need.

Google Cloud Messaging

Where will you be calling the API from?

Determines which settings you'll need to configure.

Web server (e.g. node.js, Tomcat)

Adding the push-plugin to project:

```
cordova plugin add https://github.com/phonegap/phonegap-plugin-push --variable
SENDER_ID="XXXXXXX"
```

SENDER_ID represents the Project Id

Place the following code inside receivedEvent function within index.js

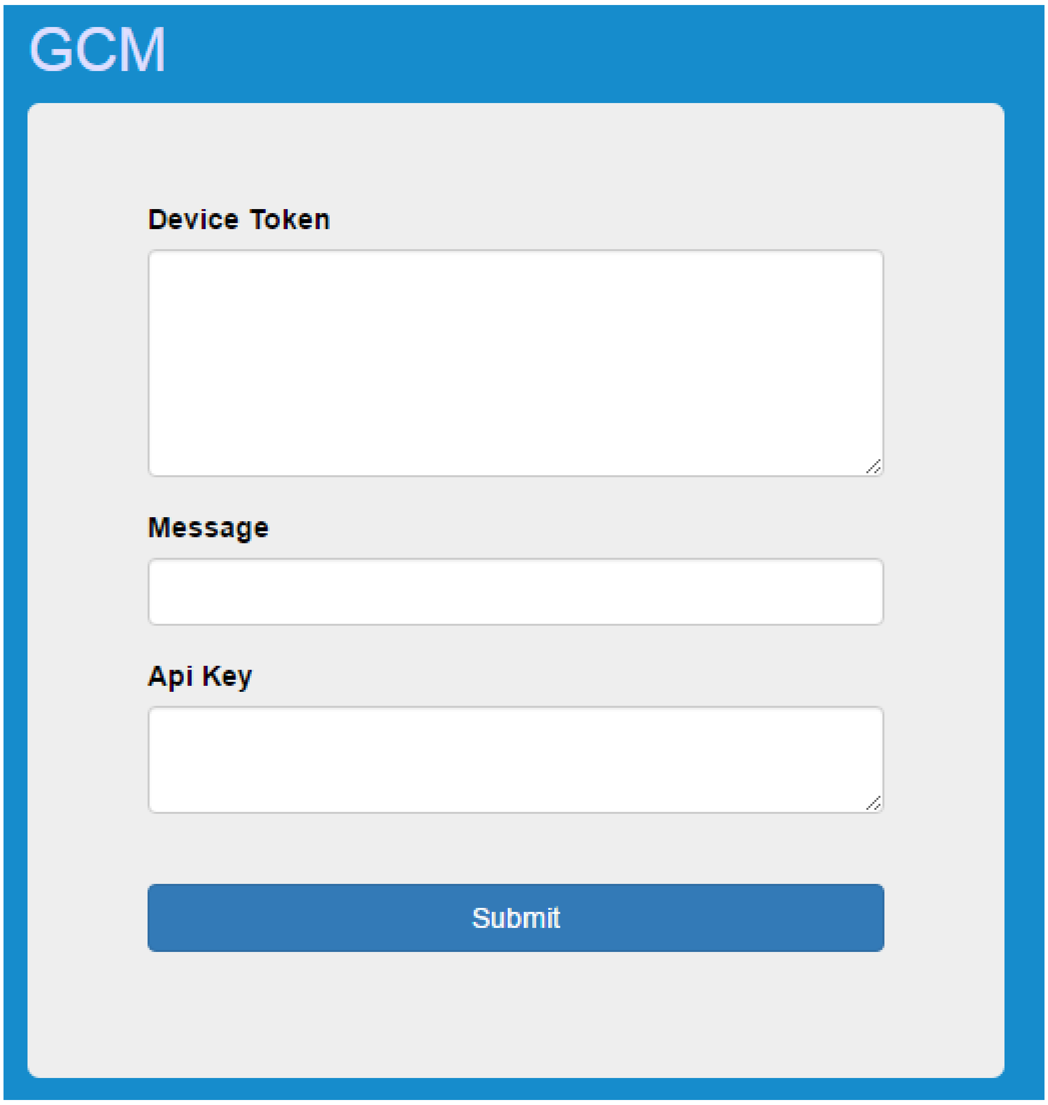
```
var push = PushNotification.init({
    android: {
        senderID: "XXXXXX"
    },
    ios: {
        alert: "true",
        badge: "true",
        sound: "true"
    },
    windows: {}
});

push.on('registration', function(data) {
    console.log("device token: " + data.registrationId);
});

push.on('notification', function(data) {
       console.log(data.message);
       console.log(data.title);
       console.log(data.count);
       console.log(data.sound);
       console.log(data.image);
       console.log(data.additionalData);
});

push.on('error', function(e) {
       console.log(e.message)
});
```

On running the above code from an Android or iOS device gives a device token.

NOTE: Device token shall be generated only on a real device not a virtual device.

For testing push notification go to this link Online Push Notification Test

For Android: Enter the Device token, Message and API key



Read Push Notification in Android and iOS online: https://riptutorial.com/cordova/topic/6181/push-notification-in-android-and-ios

Chapter 15: Sign Android build with Cordova 5

Examples

Add the build configuration to sign the .apk file

1. Add a keystore using:

```
keytool -genkey -v -keystore example.keystore -alias example -keyalg RSA -keysize 2048 -
validity 10000
```

Note: This should be at root of project. Though not a hard requirement, it eases the file referencing

2. Add a build.json with release/dev configuration for keystore, at the root of project:

```
{
  "android": {
    "debug": {
      "keystore": "..\android.keystore",
      "storePassword": "android",
      "alias": "mykey1",
      "password" : "password",
      "keystoreType": ""
    },
    "release": {
      "keystore": "..\android.keystore",
      "storePassword": "",
      "alias": "mykey2",
      "password" : "password",
      "keystoreType": ""
    }
  }
}
```

3. Add the --buildConfig switch to Cordova/ Ionic build command:

```
cordova build android --release --buildConfig=build.json
```

or with Ionic as

```
   ionic build android --release --buildConfig=build.json
```

The signed file will be generated under the new folder structure at

```
/platforms/android/build/outputs/apk/android-release.apk
```

Read Sign Android build with Cordova 5 online: https://riptutorial.com/cordova/topic/3061/sign-android-build-with-cordova-5

Chapter 16: Visual Studio Tools for Apache Cordova

Examples

Get Apache Cordova Tools in Visual Studio

1. Open Control Panel -> Programs and Features, choose the Visual Studio 2015 item, and then choose the Change button.

2. In the setup wizard for Visual Studio, choose the Modify button.

3. In the list of optional features to install, select the HTML/JavaScript (Apache Cordova) checkbox, choose the Next button, and then choose the Update button.

Update Apache Cordova Tools in Visual Studio

1. In Visual Studio, choose Tools->Extensions and Updates.
2. In the Updates tab of the Extensions and Updates dialog box, choose Product Updates. If an update for Visual Studio Tools for Apache appears, select it, and then choose the Update button.

Read Visual Studio Tools for Apache Cordova online:
https://riptutorial.com/cordova/topic/2393/visual-studio-tools-for-apache-cordova

Credits

S. No	Chapters	Contributors
1	Getting started with Cordova	Akash Pal, Community, Hitesh Riziya, Michiel, Mikhail, ModusPwnens
2	Cordova Crop Image Plugin	Akash Pal
3	Cordova ios build	sparrowTrajon
4	Cordova plugins: how to install, how they work, examples	Akash Pal, Hristo Eftimov
5	Creating Your First Application With Cordova	Devid Farinelli, grgarside, Hristo Eftimov, James Wong, Philip Bijker, ProllyGeek
6	Debugging the application	Akash Pal, Alex Filatov, Alexus, Devid Farinelli, mike nelson, Philip Bijker
7	Firebase Push Notification Cordova	Akash Pal
8	Google Analytics in Cordova	Akash Pal
9	How to customize platform specific www folder in cordova	Rahul Raghuvanshi
10	How to detect the state of the network connection	Philip Bijker
11	How to install/uninstall custom cordova plugin	Rahul Raghuvanshi
12	Make application released from Cordova CLI	Akash Pal

13	Push Notification in Android and iOS	Akash Pal
14	Sign Android build with Cordova 5	Aditya Singh
15	Visual Studio Tools for Apache Cordova	Charitha Goonewardena