***Giáo trình***

**Lập trình Ứng dụng Di động**

**GS. Phan Viết Hoàng**

**Bài học 8 (Buổi 8, Buổi 9)**

**Kết nối công cụ native**

**Truy cập thông tin Internet**

# References

# ‘Building Mobile App with Ionic 2’*, Josh Morony, 2016*

**Chức năng của công cụ**

**(Lesson 12: Native Functionality)**

The problem with web-based mobile applications is that while we can run them on iOS and Android through a web browser, users can’t install them natively on their device and the app can not access native API’s like Contacts, Bluetooth and so on. This is why we use **Cordova** in conjunction with Ionic, Cordova allows us to wrap our applications in a native wrapper which allows for submission to app stores as well as communication with native API’s through plugins. When using Cordova, a HTML5 mobile application can do just about anything a normal native application can do.

As I just mentioned, to access native functionality we need to use plugins. Cordova provides a bunch of default **plugins** which include:

* [Device](https://github.com/apache/cordova-plugin-device)
* [Network Information](https://github.com/apache/cordova-plugin-network-information)
* [Camera](https://github.com/apache/cordova-plugin-camera)
* [Geolocation](https://github.com/apache/cordova-plugin-geolocation)
* [File](https://github.com/apache/cordova-plugin-file)
* [In App Browser](https://github.com/apache/cordova-plugin-inappbrowser)
* [Media](https://github.com/apache/cordova-plugin-media)
* [Splash Screen](https://github.com/apache/cordova-plugin-splashscreen)

But there are hundreds of open sourced plugins developed by the community to do just about everything, a few popular examples being:

* [Local Notifications](https://github.com/katzer/cordova-plugin-local-notifications)
* [Facebook Connect](https://github.com/jeduan/cordova-plugin-facebook4/blob/master/plugin.xml)
* [SQLite](https://github.com/litehelpers/Cordova-sqlite-storage)
* [Social Sharing](https://github.com/EddyVerbruggen/SocialSharing-PhoneGap-Plugin)

Basically what a plugin does is create an interface where JavaScript code can trigger native calls. So if you ever run into a situation where you need a Cordova plugin that doesn’t exist yet (which is pretty rare), you can even write it yourself (but it does involve writing native code).

**IMPORTANT:** Most plugins only work when running on a real device or emulator, so if you are trying to testa Cordova plugin through ionic serve you will likely receive errors.

**Using Cordova Plugins in Ionic 2**

There’s two ways to implement native functionality in Ionic 2. You can just use any Cordova plugin directly by installing the plugin in your project:

ionic plugin add plugin-name

and then accessing the functionality that the plugin provides, which is usually available on a global object like this:

window.plugins.somePlugin.someMethod();

Nothing needs to be imported, required, called from a specific section of code or anything else - once you have installed the plugin through the command line you will be able to access it from anywhere. Not all plugins will be accessible exactly like this, but it’s how most plugins work.

This is not specific to Ionic 2, you can use Cordova plugins in this manner in *any* Cordova project (the only diﬀerence being that you would use cordova plugin add instead of ionic plugin add). When using the normal Cordova syntax, using a plugin in Ionic 2 is no diﬀerent than using it in **Ionic 1, Sencha Touch, jQuery Mobile** or a normal web page built with Cordova.

Keep in mind that if you use Cordova plugins in this way, your application may fail to compile due to TypeScript warnings. This is because TypeScript does not know what it is, and you may need to install typings for it. To brute force your way past this, you can simply add:

**declare var** variableCausingProblems;

above the decorator in the class that you are using the plugin in.

Alternatively, you can use **Ionic Native** to make use of Cordova plugins, which is specific to Ionic 2. If you’re familiar with **ngCordova** from Ionic 1 then this is basically the same thing, just for Ionic 2. If you’re not familiar with ngCordova, Ionic Native basically just makes Cordova plugins play a little bit more nicely with Angular 2, by adding support for Promises and Observables.

Ionic Native is installed by default in all Ionic 2 applications, so all you need to do is install the plugin you want to use, just like you would normally, for example:

ionic plugin add cordova-plugin-geolocation

Next you will need to import the plugin from Ionic Native into the class you want to use it in:

**import** { Geolocation } from'ionic-native';

and then you can use it in your code:

Geolocation.getCurrentPosition().then((resp) => { **console**.**log**("Latitude: ", resp.coords.latitude); **console**.**log**("Longitude: ", resp.coords.longitude);

});

Notice that in the code above a promise is returned and we set up a handler using .then(), if we were just using the standard Cordova syntax this wouldn’t be possible - we would instead have to use callback functions, which are a bit messier.

It’s also important to note that not all Cordova plugins are available in Ionic Native. For a list of all of the available plugins, and how to use them, you should check the [Ionic Native documentation](http://ionicframework.com/docs/v2/native/). If a plugin you want to use is not available in Ionic Native, then you can just go back to using the standard Cordova syntax (or you can [add it to Ionic Native yourself](https://github.com/driftyco/ionic-native/)).

Although it is not required, **you should use Ionic Native wherever possible**. It’ll make your code cleaner, and it makes much more sense in the Angular 2 ecosystem (and typings will be handled automatically this way, so TypeScript won’t complain). Using plain old Cordova is not a crime though, so don’t feel too bad about it.