How to Convert a Simple Web Site into a Mobile App

Using PhoneGap Build, PhoneGap Desktop, PhoneGap CLI or Android Studio

Instead of learning how to program in a native mobile app language (e.g., Java/Kotlin for Android or Objective-C/Swift for iOS), you can create simple apps using standard web technologies (e.g., HTML, CSS, and JavaScript). In 2015, designer Frances Berriman and Google Chrome engineer Alex Russell coined the phrase Progressive Web Apps (PWAs) to describe this new trend.

These apps look like traditional apps that attempts to combine features offered by most modern browsers with the benefits of a mobile experience. These hybrid apps (web pages within a mobile app) are made possible because of the following:

1. Enhanced web technologies (e.g., HTML5, CSS3, and JavaScript)
2. Responsive Design (e.g., Flexbox, Grid)
3. More capable web browser standards (e.g., Local Storage, geolocation, service workers, and web app manifests)
4. Faster processors (e.g., A10 and Snapdragon 821)
5. Mobile friendly JavaScript frameworks (e.g., jQuery Mobile, Bootstrap, or AngularJS)
6. Ease-to-use app tools (e.g., PhoneGap Build, PhoneGap Desktop, PhoneGap CLI, Android Studio, XCode, etc.) to create assets (key stores, graphics, code, etc.)

The biggest advantage of PWAs is the cost saving due to lower app development and maintenance cost which may be up to 10 times smaller than native apps [14]

EXCEPTION: The major reason that you may want to create a native app using Java/Kotlin or Objective-C/Swift is if you are creating a game or graphic intensity app that you need the performance.

We will examine four (4) options to convert a standard web site to a mobile app:

OPTION	LEVEL
Use PhoneGap Build	Beginner
Use PhoneGap Desktop	Intermediate
Use PhoneGap CLI	Advanced
Use Android Studio	Expert

NOTE: There are dozens of other ways and tools available to create mobile apps:

- Ionic, React Native, Como, etc.
- Progressive Web Apps (PWA)

- Android Studio with HTML, CSS, and JS
- Android Studio with Java or Kotlin
- xCode with Objective-C or Swift

OPTION 1: Use PhoneGap Build

Before creating an app, start by creating a responsive web site or use an existing web site with standard HTML5, CSS3, and optionally any JavaScript framework (Bootstrap, jQuery Mobile, and AngularJS). Then, use PhoneGap Build to create the app.

PHASE 1: Create Web Site

First, create a web site and add a few tags that are used by mobile devices.

1. **Create a responsive web site or use an existing web site**.
2. **In the index.html file, add the following line inside of the head element**:

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

3. **Add the following line above the closing body tag (</body>) inside of the index.html file:**

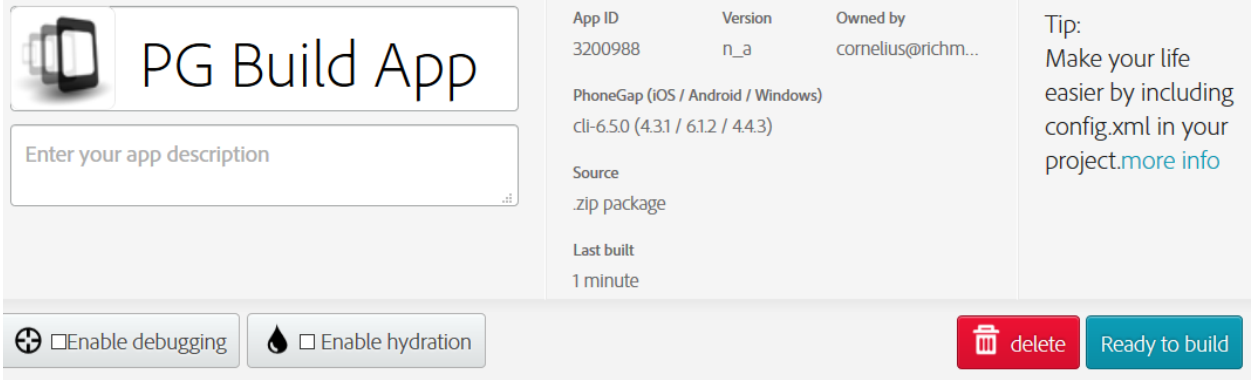
   <script type="text/javascript" src="cordova.js"></script>

NOTE: These two lines are needed to make your web site mobile ready and mobile friendly. You **DO NOT** need to include an actual **cordova.js** file as it will be created by the PhoneGap Build framework.

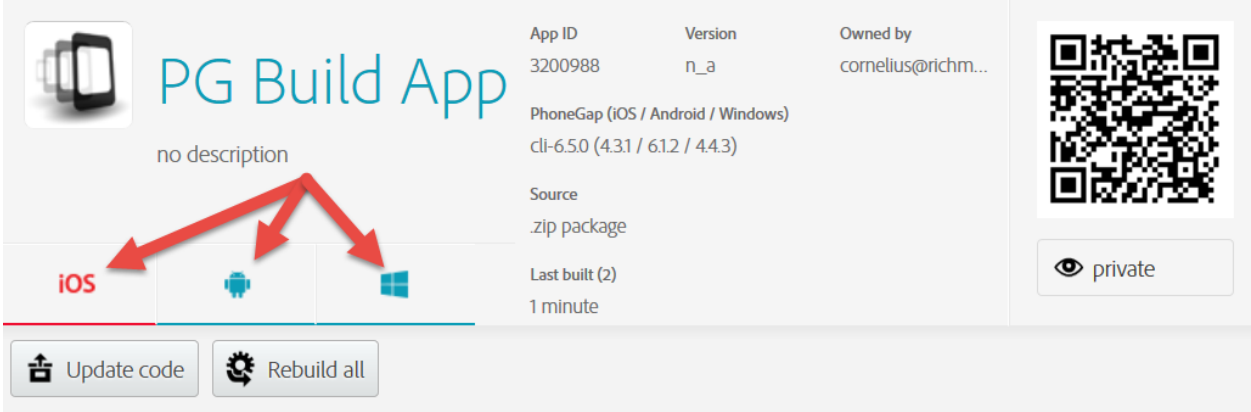
PHASE 2: Create Debug Version

After you create a web site, you may want to convert it into a debug version of the app to see how things work. You could skip this step once you become more efficient in how to create an app.

1. **Zip your web content** (HTML, CSS, JS, Images) and **give it a descriptive name** (e.g., HelloWorldApp.zip).
2. **Go to the PhoneGap Build Cloud Service** (https://build.phonegap.com/). You may want to take some time to preview this home page to see what is available.
3. If you have not already done so, sign up for the PhoneGap Build service by **clicking the Sign up button** at the top of the page and **provide all the necessary information needed**.
4. **Return to the PhoneGap Build home page** and **click the Sign In button**.
5. **Click the + new app button, click the Upload a .zip file button, navigate the zip** file you created earlier, and then **click the Open button**.
6. **Click the Ready to Build button**.



7. **CHECK POINT:** You should see the platform icons (IOS, Android, and Windows) get highlighted. The IOS icon is highlighted in red because it needs additional information.



8. If necessary, **download a QR Reader** on your mobile device.
9. Using the QR Reader, take a snapshot of the QR code (See screenshot above).
10. On your mobile device, **click the following prompts**:
    a. **Click OK** on the **Open URL prompt**
    b. **Click SAVE** on the **Download file? prompt**
    c. **Click Open** on the **1 file download prompt**
    d. **Click Install** on the **Do you want to install this application?...** prompt
    e. **Click Open** again
11. **CHECK POINT:** The app will be downloaded to your mobile device.
12. Test the app on your mobile device.

PHASE 3: Add Assets and Config File

If you don't add graphic assets and a config.xml file to the zip file, the PhoneGap Build Cloud Service will use default values and images. This is great for initial test, but you will want to include these items before you create a release version of the app.

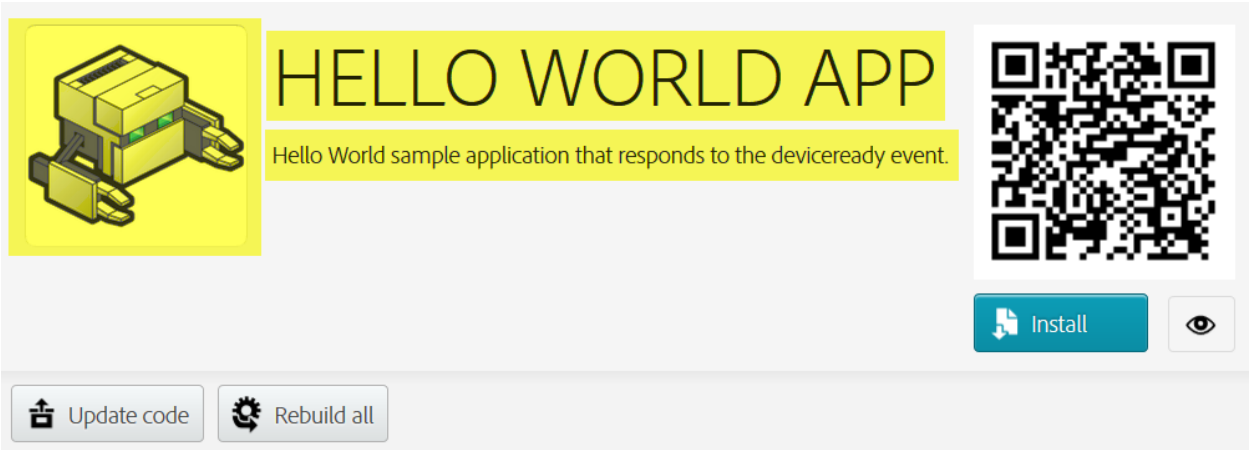
1. **Create Assets** (icon, splash screens) **in a folder called res** (short for resources).

   NOTE: See list of Splash Screen Generators under resources for ways to create these resources automatically.

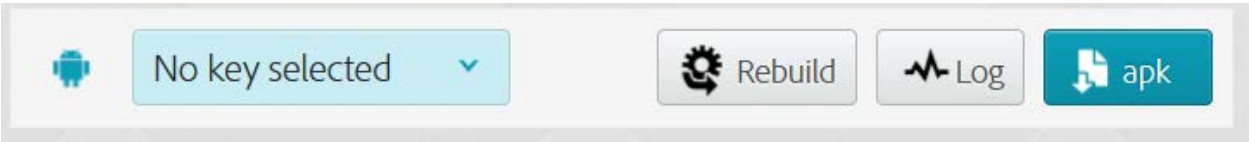
2. **Create a config.xml file or use an existing config.xml file and update it.**

   NOTE: This file is used to "configure" your app with specific settings (e.g., app name, app version, author, platforms, plugins, internet access, preferences, etc.) For details on what to add to the config.xml file, click here.

3. **Delete initial zip file.**
4. **Rezip files to include asset folder and the config.xml with the same name** (HelloWorldApp).
5. **Go back to PhoneGap Build, click the Update Code button**, **click the Browse… Button**, **navigate to the newly created zip file**, and then **click the Upload button**.
6. **CHECK POINT:** Because a config.xml file was included in the zip file this time, you should see the **app image, app name,** and a **short description** that was gleaned from the config.xml file



7. **Click the Rebuild button** to update the app with the new content.



8. Using the QR Reader, **take a snapshot of the QR code** (See screenshot above).
9. On your mobile device, **click the following prompts**:
   a. **Click OK** on the **Open URL prompt**

b. **Click SAVE** on the **Download file? prompt**
c. **Click Open** on the **1 file download prompt**
d. **Click Install** on the **Do you want to install this application?... prompt**
e. **Click Open** again

10. **CHECK POINT:** The app will be downloaded to your mobile device.
11. Test app on your mobile device.

PHASE 4: Digitally Sign App

Once an app has been created, it must be **DIGITALLY** signed to create a **RELEASE** version instead of a **DEBUG** version of the app before it can be submitted to an app store (e.g., Google Play, Apple App Store).

CAUTION: You need to have JDK installed to create a signing key.

1. If you have not already done so**, download the** Java Development Kit (JDK), **double-click the file and follow the instructions on the JDK web site on how to install it**.
2. Open the **command prompt** from within Windows by typing **command** in the search field at the bottom of the screen: **(VERIFY BULLETS BELOW)**
   - Type cd... and press the ENTER key.
   - Type cd Program files and press the ENTER key.
   - Type cd Java and press the ENTER key.
   - **Type cd jdk1.8.0_25** (or whatever is the later version) and p**ress the ENTER key**.
   - **Type cd bin** and **press the ENTER key**.

     TIP: You can also set up a system variable to avoid having to do these steps.

3. **Type the following syntax** and **change what is highlighted to your information**:

   keytool -genkey -v -keystore **Your_personal_file_title**.keystore -alias **Your_personal_alias** -keyalg RSA -keysize 2048 -validity 10000

CAUTION: Ensure you write the password down that you will enter in the next step and do not lose it. Note also that when you are TYPING your password, the large block insertion will not move pass the first letter in your password.

4. Enter your **profile** when prompted:
   i. key store password: **mykeystore$500**
   ii. first and last name: **Cornelius Chopin**
   iii. organizational unit: **skip**
   iv. name of your organization: **Rich Media Creative Services**
   v. city: **Round Rock**

vi. state: **Texas**
vii. two letter country code: **US**
viii. is correct? **yes**
ix. key password: Press the **Enter key** to add the **SAME** password as key store password **mykeystore$500**
x. **CHECK POINT:** This will create an Android Keystore file in the **CURRENT** directory: (e.g., C:\Users\Cornelius) with the name you gave it. You can use it with whatever app you generate (e.g., https://build.phonegap.com).

TIP: You may want to take a screen shot or a text file of the command prompt with all its settings and save an electronic copy in a save place along with a printed copy. So, you may want to save it in multiple places and even in the cloud (e.g., drop box or google drive). If you lose SIGNING KEY, you won't be able to update your app.

PHASE 5: Create Release Version

Now that you have created a signing key, you can use it to digitally sign the app to create a release version instead of a debug version that can be sent to an app store.

1. **Click the down arrow next to No Key selected text, select the newly created key, enter the certificate and key store passwords, and then click the submit key button.**
2. **CHECK POINT:** This now shows the lock icon as being unlock.
3. **Click the Rebuild button again.**
4. This time, **click the apk button** to download the APK to your computer.
5. **CHECK POINT:** Notice the file name has the word "release" included in it. This is the file that can be used to submit your app to an app store.

PHASE 6: Upload App

Once the release version of the app is created, you can **UPLOAD** it to Google Play or Apple App Store.

NOTE: You will need **to pay $25 one-time fee to Google** to create as many apps as you want. However, you will need **to pay $99 every year to Apple** to create as many apps as you want.

1. **Go to the** Google Play Console and sign up if you have not already done so.
2. **Click the Launch Play Console button**
3. **Click the CREATE APPLICATION and add the necessary information.**

OPTION 2: Use PhoneGap Desktop

As an alternative to using **PhoneGap Build** or PhoneGap CLI (next option), you can use the **PhoneGap Desktop** app to create a mobile app. This is for users who prefer to use a visual interface instead a Command Line Interface (CLI). Another advantage of using this option is that it will automatically create a default config.xml file and assets that you can updated with your own data.

PHASE 1: Create Web Site

Create or use an existing **responsive** web site using standard HTML5, CSS3, and any JavaScript framework (Bootstrap, jQuery Mobile, and AngularJS)

PHASE 2: Install PhoneGap Desktop

First, choose one of two OS installation options below:

Window Installation

1. **Download the latest version of the PhoneGap Desktop Installer**.

   NOTE: If you cannot run the Installer, download the zip file for the latest release and follow the instructions in the README and INSTALL files within the downloaded zip file.

2. **Double-click the installer icon and follow the prompts and buttons** to install the PhoneGap Desktop app.

Mac Installation

1. **Download the latest version of the Mac OS X Installer**.
2. **Double-click the installer icon to run the installer and accept the license agreement.**
3. **Drag and drop the application into the Applications folder** on your Mac to install the PhoneGap Desktop app.

PHASE 3: Install PhoneGap Developer App (Optional)

The PhoneGap Developer App can be used to preview and test PhoneGap mobile apps you create across multiple platforms without additional SDK setup. It provides access to PhoneGap core API without having to install any plugins or compile anything locally.

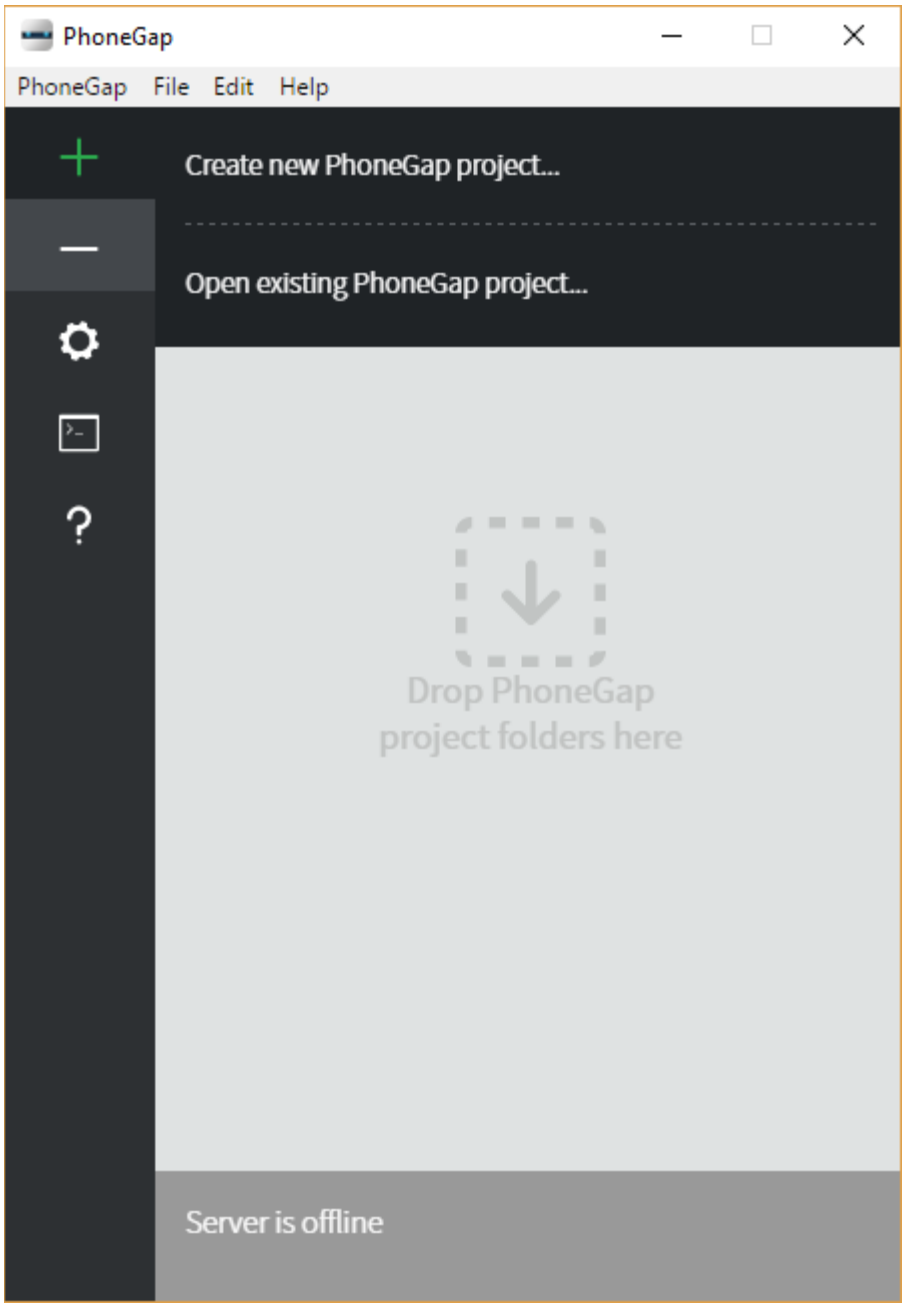
NOTE: Due to Apple guidelines, the PhoneGap Developer App has been removed from the iOS App Store. New users will be unable to download it.

1. Download PhoneGap Developer App for Android on your mobile device.
2. It will be used later once an app is created.

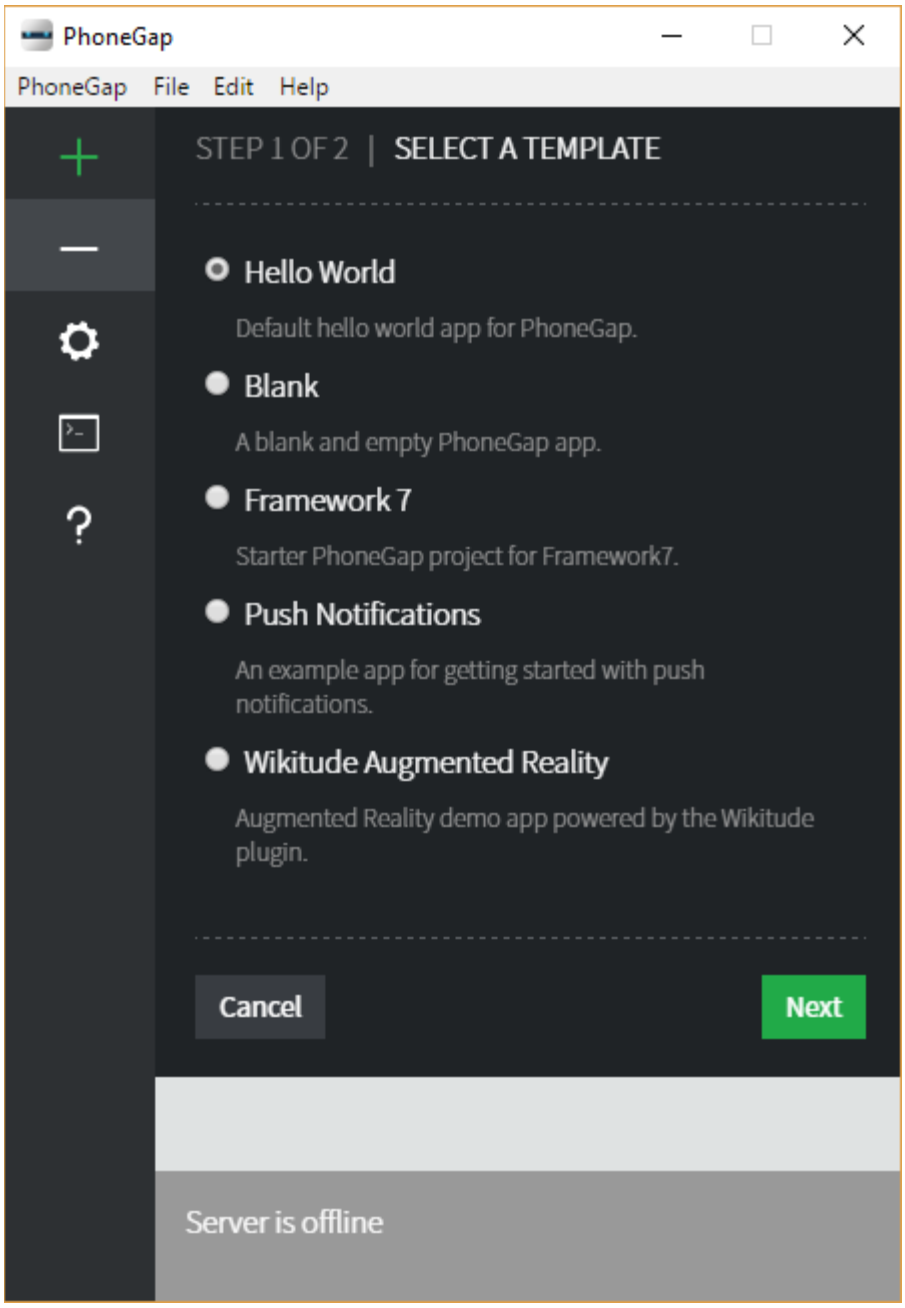
PHASE 4: Create Stub App

Now that we have the PhoneGap Desktop app installed, let's create a stub app.

1. **Create a folder where you want your mobile app to be created (**e.g., Desktop/MyApps)
2. **Open the PhoneGap Desktop app, click the plus sign (+), and then select the Create new PhoneGap project…** from the menu.



3. You are prompted in STEP 1 of 2 to select a template. For this demo, **select the Hello World radio button** to select that template and then **click the Next button**.

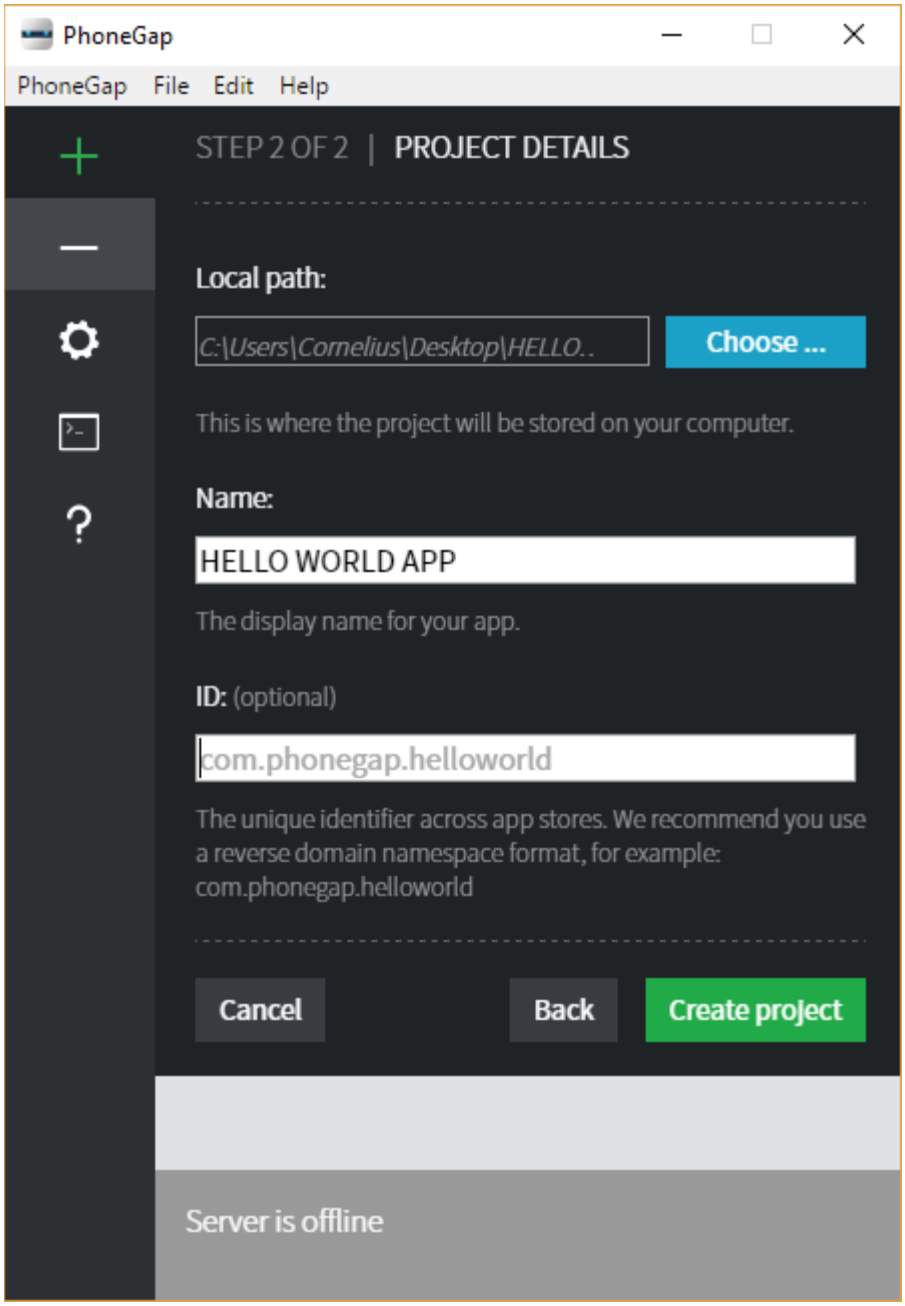


4. In STEP 2 OF 2 of the template:

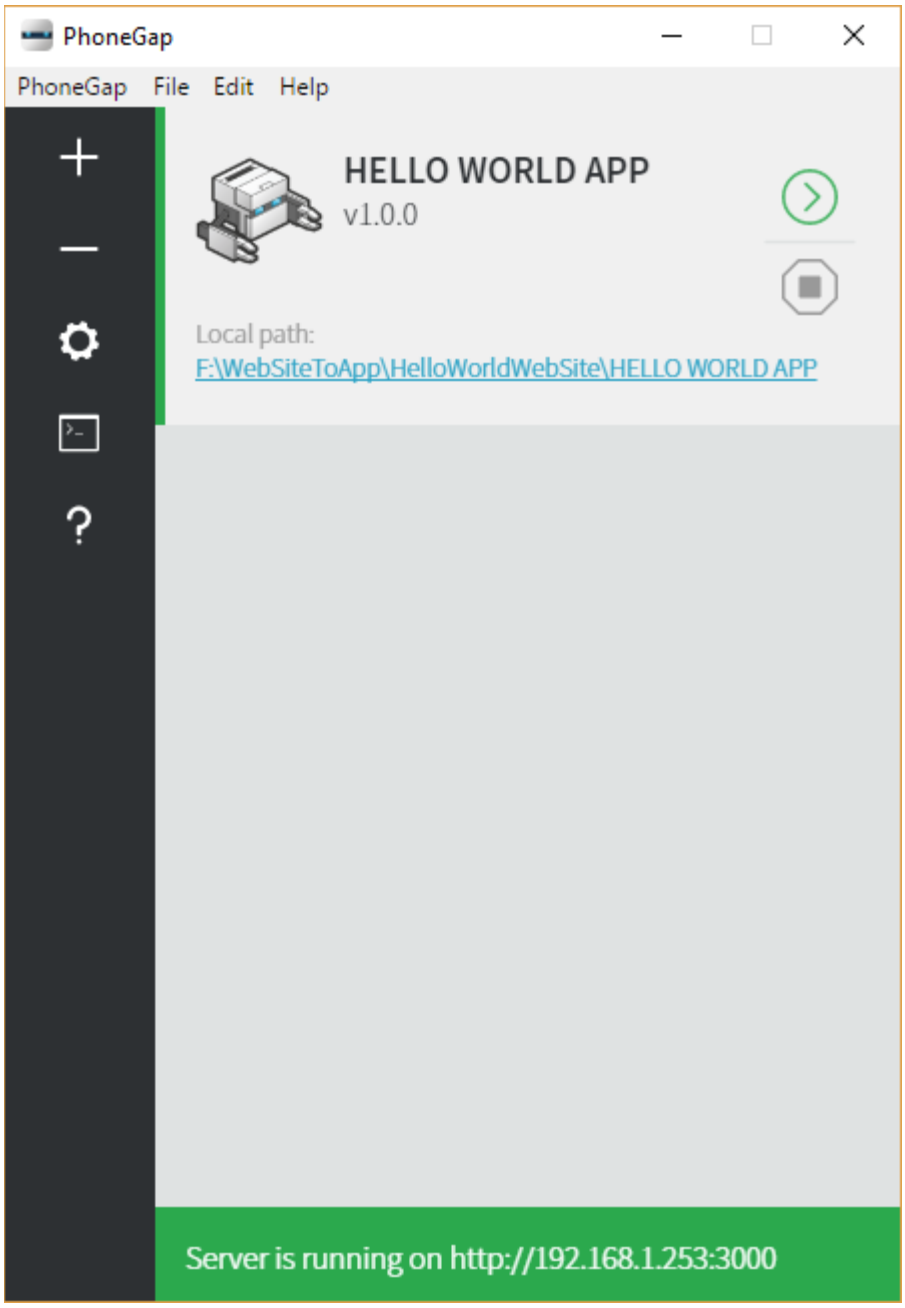
- **Click the Choose… button and select the folder where you want the app assets to be saved and then click the Select Folder button.**
- **Give the app a name** in the Name: text field (e.g., HELLO WORLD APP).
- **Give the app an optional ID** in the ID text field (e.g., com.phonegap.helloworld)

NOTE: It is common to use the reversed domain naming convention for an app name that will be used to create a **UNIQUE** app name. The ID field is known as the *package identifier* for Android and the *bundle identifier* for iOS.

- **Click the Create project button.**



5. **CHECK POINT:** You will be presented with the following screen:



NOTE: PhoneGap Desktop will automatically start a small web server to host your project and returns a server IP address and server port number that you can then enter into the **PhoneGap Developer** app running on your mobile device or in your desktop browser. You can also manually start the project that is inactive by clicking the green play (**>**) button.

The advantage of using your desktop browser it that it can speed up the initial development process if you are using frameworks like Angular or React where there are tools to perform debugging before moving your app to a device. Recently, PhoneGap began supporting the browser platform as a target automatically to help you test with the `deviceready` event

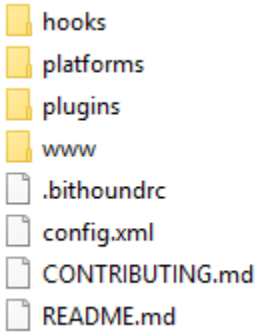
and Apache Cordova core plugins more easily in an environment you're already familiar with.

CAUTION: You phone and the mobile device need to be connected to the SAME NETWORK for the PhoneGap App to work.

6. **(OPTIONAL) Open the PhoneGap Developer app** on your mobile device that you downloaded earlier and **enter the IP address and the port number** at the bottom of the PhoneGap Desktop app (e.g., http://192.168.253:3000) into the Server Address text field and then **click the Connect button** to preview the app.

   NOTE: This app does not always work…

7. **Click the Local path link to open the files in the File Explorer to examine the folders and files** that it created.



   NOTE: The advantage of using the PhoneGap Desktop app is that it will AUTOMATICALLY create the following default file and folders for you:
   1. **Platforms** folder with platform files already created
   2. **Plugins** folder with several default plugins
   3. Config.xml file

8. **CHECK POINT: Click the www folder** to examine its content. Except for the spec folder, if you click any of these folders, you will see for the most part that the PhoneGap Desktop app created standard web content.
9. **CHECK POINT: Click the index.html file**. You will see the default app (e.g., HELLO WORLD) opens in your default browser.

PHASE 5: Add Web Site

Now that we have the default content working, replace it with our own web content in the www folder.

1. **Create or use an existing web site** (using standard HTML5, CSS3 and optionally any JavaScript framework (Bootstrap, jQuery Mobile, and AngularJS))
2. **MOVE your web content to the www folder**.

PHASE 6: Convert To App

SEE PHASE 2-6 FROM OPTION 1

OPTION 3: Use PhoneGap CLI

If you are a code geek, you may want to use the PhoneGap CLI approach.

PHASE 1: Create Web Site

Create or use an existing **responsive** web site using standard HTML5, CSS3, and any JavaScript framework (Bootstrap, jQuery Mobile, and AngularJS)

PHASE 2: Install PhoneGap CLI

The PhoneGap CLI provides a command line interface (CLI) for creating PhoneGap apps as an alternative to using the PhoneGap Desktop App for those who prefer working at the command line. It has some additional features over the PhoneGap Desktop for building, running, and packaging your PhoneGap applications on multiple platforms.

Before you can use the PhoneGap CLI, you will need to download and install node.js and git:

- node.js - a JavaScript runtime to build JavaScript code
- git - used in the background by the CLI to download assets. It comes pre-installed on some operating systems. To see if you already have it installed, type `git` from the command line.

1. Install the PhoneGap CLI via `npm` with the following command from the Terminal app (Mac) or Command Prompt (Win).

   $ **npm install -g phonegap@latest**

- The **$** symbol is used to indicate the command prompt and should not be typed.
- **npm** is the node package manager and installed with node.js. The `npm` command fetches the necessary dependencies for the PhoneGap CLI to run on your local machine. It creates a *node_modules* folder with the necessary code needed to run the CLI.
- The **-g** flag specifies that folder to be installed at the global location so it can be accessed from anywhere on your machine (defaults to */usr/local/lib/node_modules/phonegap* on Mac).
- **OS X Users:** You may need to prefix this command with `sudo` to allow installation to restricted directories and type the following instead: `$ sudo npm install -g phonegap@latest`

2. Type **phonegap** at the command prompt.

3. **CHECK POINT:** You should see a list of commands that can be used with phonegap.

```
Commands:

  help [command]       output usage information
  create <path>        create a phonegap project
  build <platforms>    build the project for a specific platform
  install <platforms>  install the project on for a specific platform
  run <platforms>      build and install the project for a specific platform
  platform [command]   update a platform version
  plugin [command]     add, remove, and list plugins
  template [command]   list available app templates
  info                 display information about the project
  serve                serve a phonegap project
  version              output version number
```

TIP: You can access the PhoneGap CLI usage text at any time by adding the keyword `help`, or the `-h` or `--h` attribute with any phonegap command (e.g., `$ phonegap create help`, `$ phonegap serve -h`).

PHASE 3: Create Stub App

You can create a stub app by creating the app, adding the platform(s) and the plugins.
TIP: Remember the acronym: **APP**

A – create app
P – add platform(s)
P – add plugin(s), if necessary

1. **Type the following command to create a stub app:**

   $ **Phonegap create HelloWorldApp com.richmediacs.helloworldapp**

   NOTE:
   - HelloWorldApp is the name of the folder that the app will be created in.
   - **com.richmediacs.helloworldapp** is the app unique ID

2. **Navigate to the folder that the folders and files were created** in (e.g., C: /Users/Cornelius/HelloWorldApp).

   NOTE: If you click the plugins or platforms folder, you will see that they are currently both empty. This will be resolved later.

3. **Drill down to the www folder** and then **click the index.html file**.
4. **CHECKPOINT:** You should see the app loads in the browser.
5. **Type cd/ and then the name of the folder** (HelloWorldApp) to change the directory to that folder.

   $ **cd/HelloWorldApp**

6. **Type the following command** to add the Android platform:

   $ **Phonegap platform add android**

   NOTE: You can do the same thing for ios by typing: Phonegap platform add ios

7. **CHECKPOINT:** If you double-click the platforms folder, you will see that it added platform content there. If you click the plugins folder, you will see that it also added some default plugins.

   NOTE: You can add additional plugins as needed. See next step.

8. **(Optional) Type the following command** to add the admobpro plugin:

   $ **Phonegap plugin add cordova-plugin-admobpro**

9. Now, build the app by type the following command:

   $ **Phonegap run android**

   NOTE:
   - You can do the same thing for ios by typing: Phonegap run ios
   - You can also zip up the content of the www folder along with the config.xml file and upload it to the PhoneGap Build Cloud service. (See OPTION 1: PHASE 2)

PHASE 4: Add Web Site

Now that we have the default content working, let's replace it with our own web content in the www folder.

- **MOVE your web content to the www folder**.

PHASE 5: Convert To App

SEE PHASE 2-6 FROM OPTION 1

OPTION 4: Use Android Studio

PHASE 1: Create Web Site

Create or use an existing **responsive** web site using standard HTML5, CSS3, and any JavaScript framework (Bootstrap, jQuery Mobile, and AngularJS)

PHASE 2: Install Android Studio

First, choose one of two OS installation options below:

Window Installation

1. **Download the latest version of Android Studio for Windows**
2. **Double-click the installer icon and follow the prompts and buttons** to install the Android Studio.

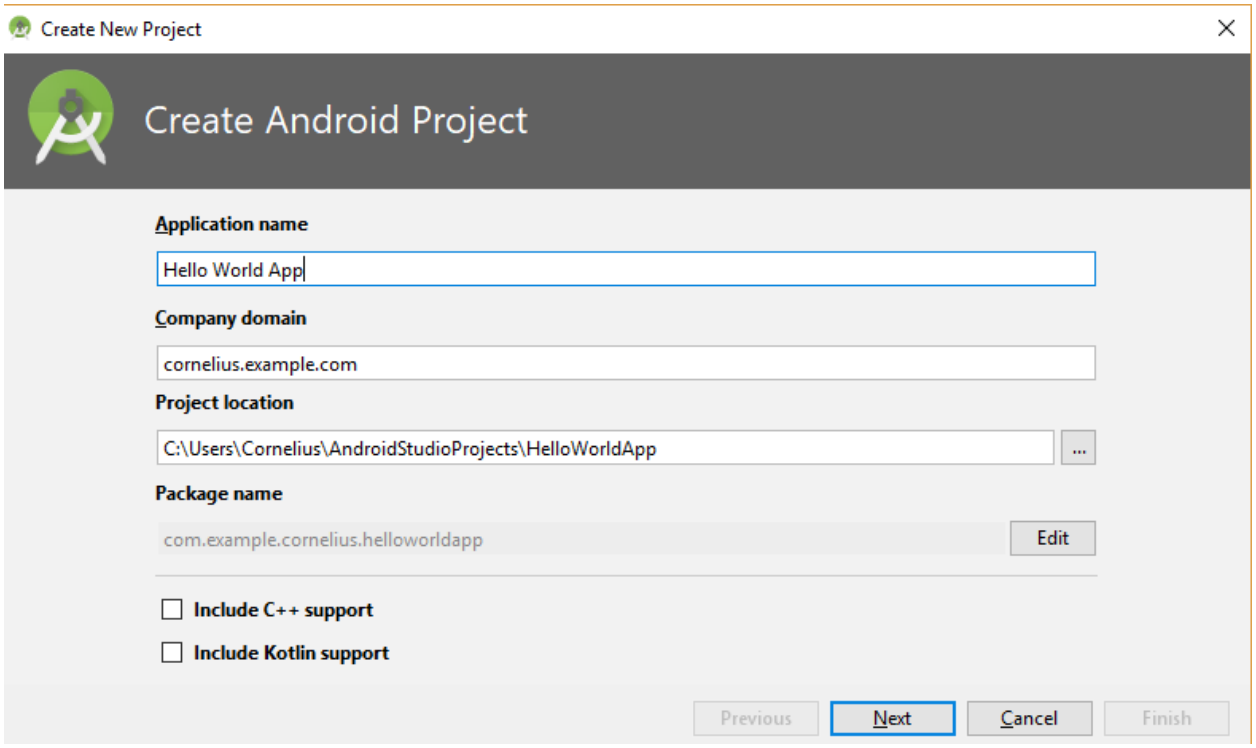
Mac Installation

1. **Download the latest version of Android Studio for Mac**
2. **Double-click the installer icon to run the installer and accept the license agreement.**
3. **Drag and drop the application into the Applications folder** on your Mac to install Android Studio

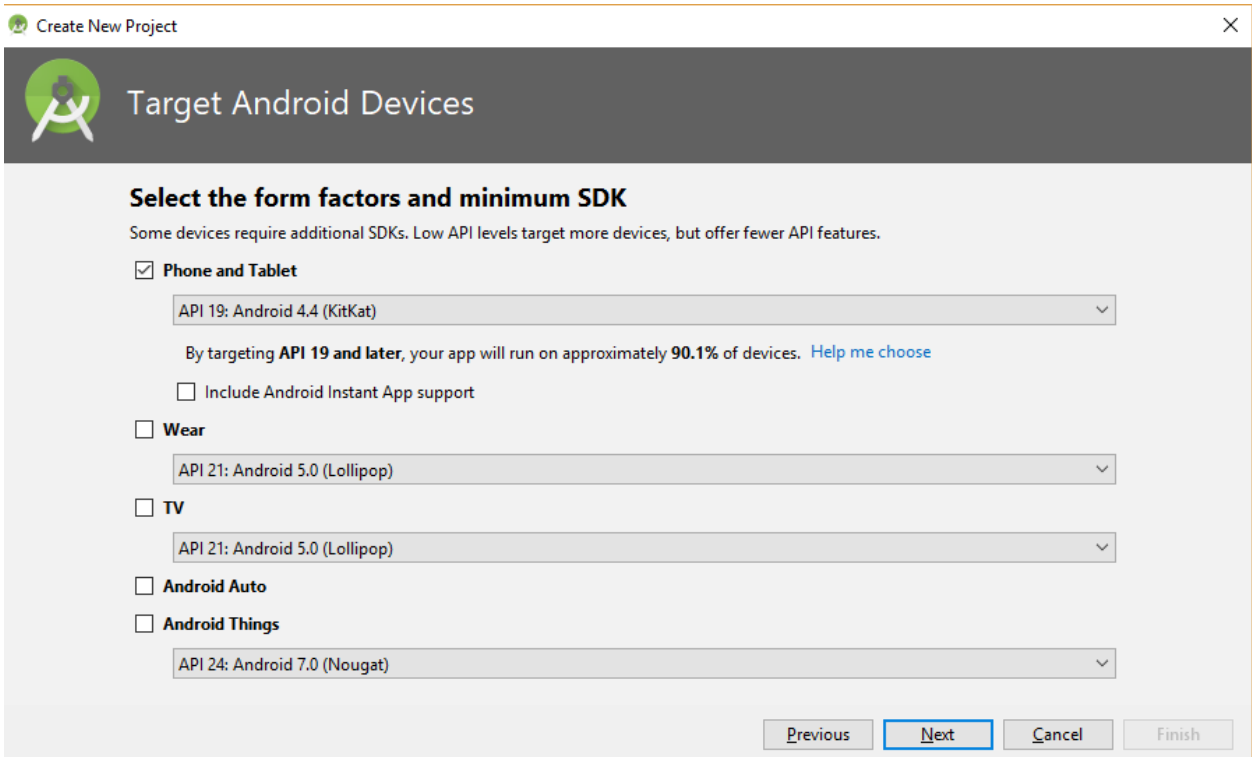
PHASE 3: Create Stub App

Now that we have Android Studio installed, let's create a stub app.

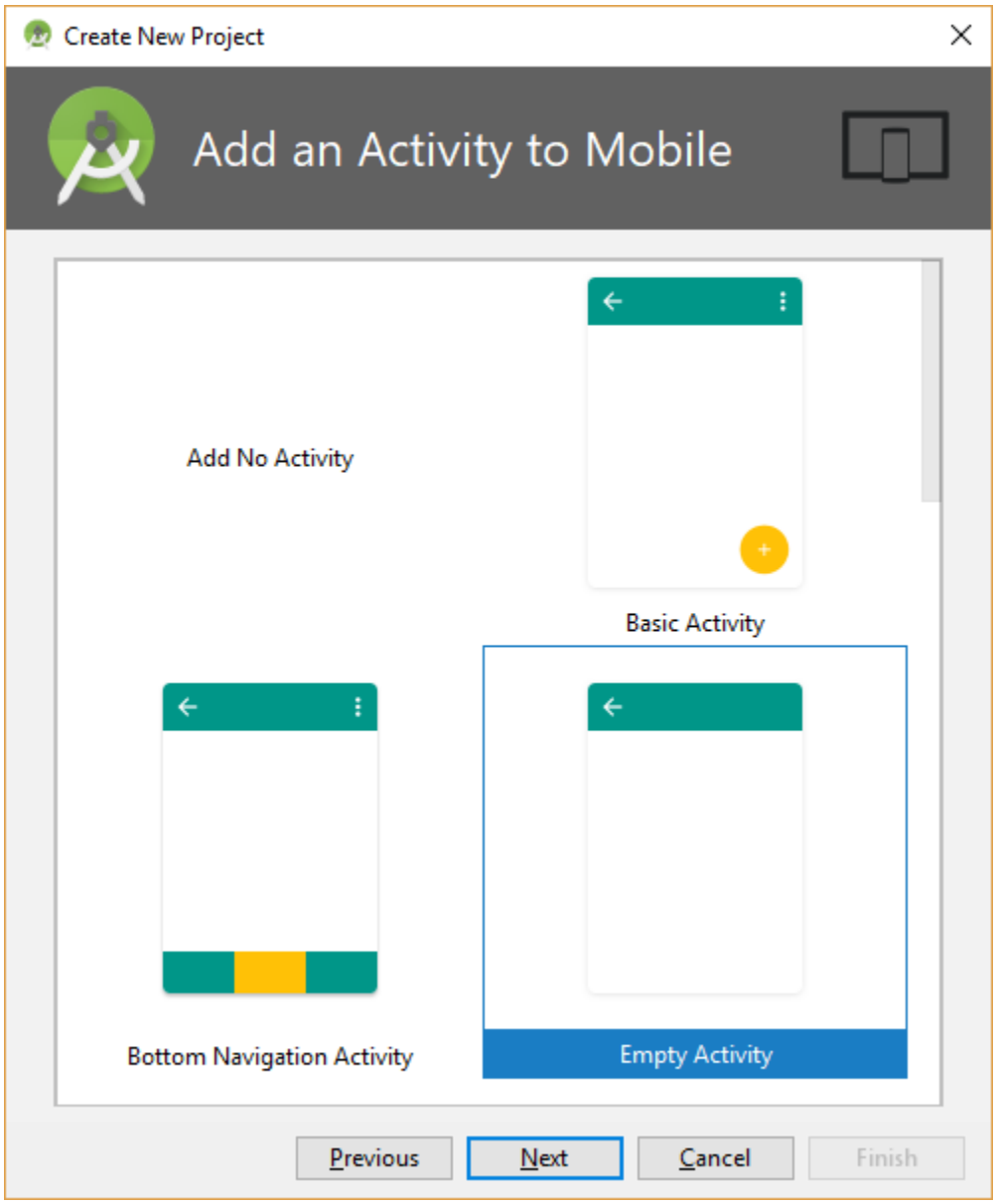
1. **Create a folder where you want the app content to be saved.**
2. **Open Android Studio** and **select File > New > New Project...**
3. In the Create New Project dialog box that appears, **add the following info** and then **click the Next button**.
    1. Application Name
    2. Company Domain
    3. Project Location



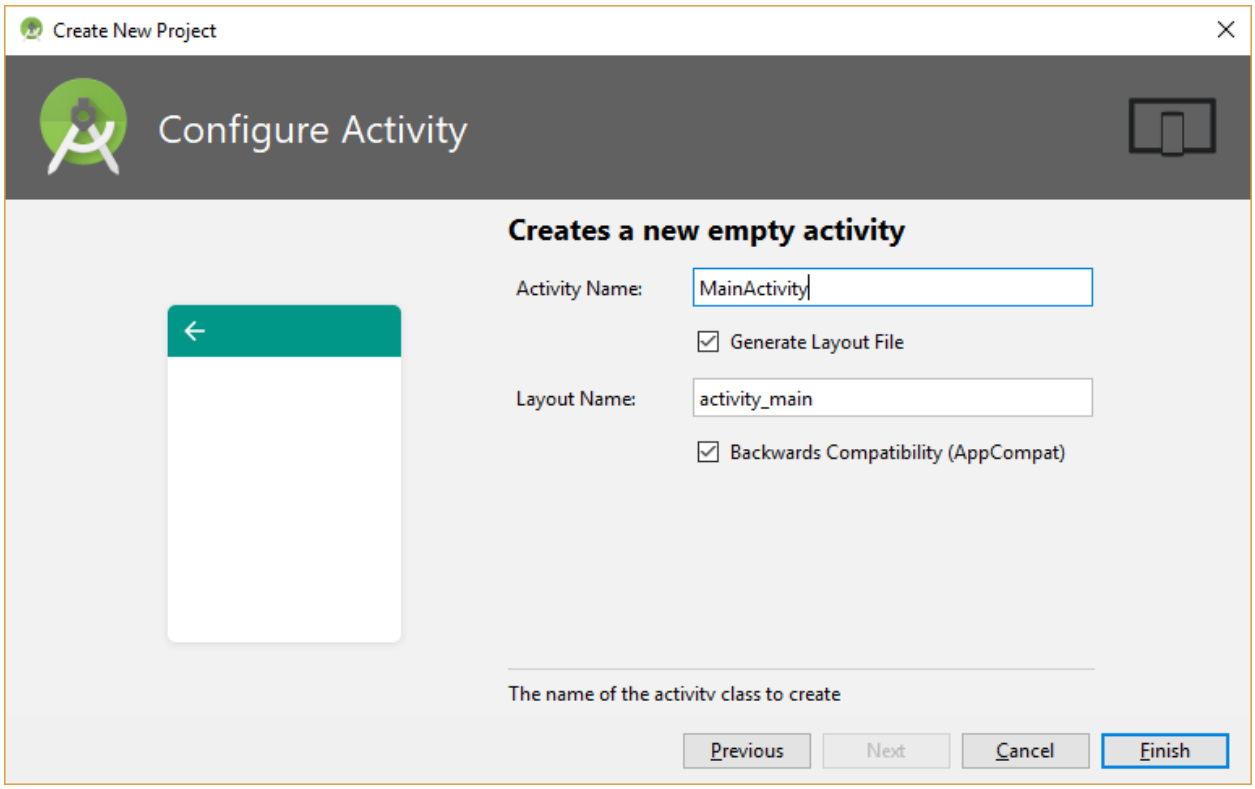
4. In the Target Android Devices dialog box that appears, **accept the defaults** for this simple project and the **click the Next button**.



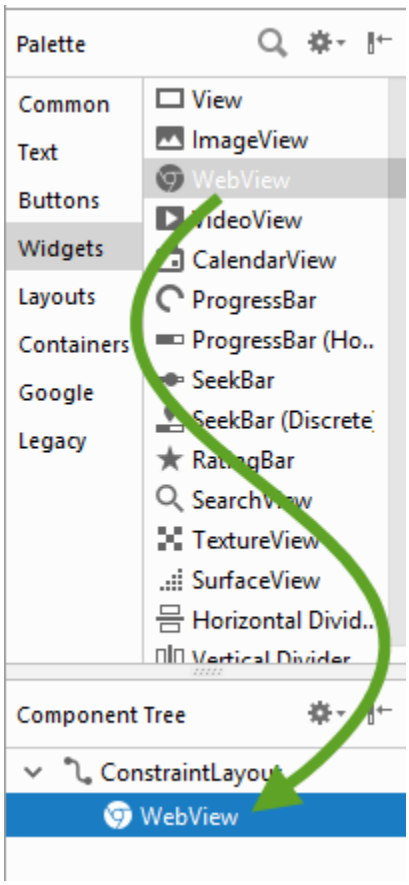
5. In the Add an Activity to Mobile dialog box that appears, **select the Empty Activity option** and then **click the Next button**.



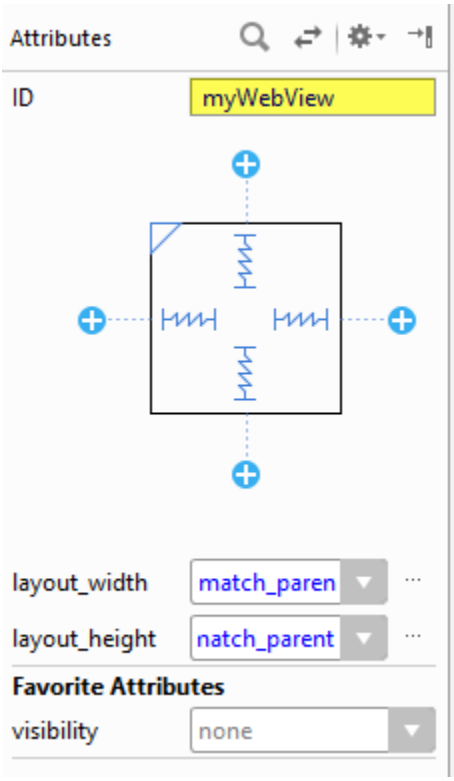
6. In the Add an Activity to Mobile dialog box that appears, **select the Empty Activity option** and then **click the Finish button**.



7. **CHECK POINT:** After Gradle finish building the app, **click the Run button** to run the app on a mobile device or in an emulator to see if it works with the default settings. You should see the app loads with the phrase Hello World.
8. In the Android panel, **click res > layout** and then **double-click the activity_main.xml file** to open it.
9. In the Design view, **select the TextView** and then **press the Delete key.**
10. From the Palette panel **select the Widgets category,** and then **drag-and-drop a WebView component the below the ConstraintLayout component in the Component Tree panel**.



11. With the WebView component selected, in the Attributes panel, **give the WebView component an ID of myWebView**.



12. **Open the MainActivity.java file** and **add the following highlighted code within the MainActivity() class:**

```
public class MainActivity extends AppCompatActivity {
   private WebView view;
   @Override
```

13. **Add the following highlighted code within the onCreate() method below the setContentView() method statement**:

```
setContentView(R.layout.activity_main);
String myURL = “file:///android_asset/www/index.html”;
WebView view = (WebView) findViewById(R.id.myWebView);
view.loadUrl(myURL);
view.getSettings().setJavaScriptEnabled(true);
view.setWebChromeClient(new WebChromeClient());
```

NOTES:
- The setJavaScriptEnabled(true); set the JavaScript to be enabled as the method implies.
- The setWebChromeClient() method is used so that the page does not load in another browser window.

- Notice the triple forward slashes (file:///). However, this could be any web url (e.g., http://www.mycompanyname.com/) instead of a file name. However, you **may** have to give your app internet permission in the AndroidManifest.xml file by writing the following highlighted code: (VERIFY IF THIS IS STILL NEEDED)

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.cornelius.helloworldapp" >
   <uses-permission android:name="android.permission.INTERNET"/>
```

14. **Click inside the first instance of the red word WebView** and then **press ALT+ENTER.**
15. **CHECK POINT:** Pressing ALT+ENTER will import the WebView at the top of the file.

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.webkit.WebView;
```

16. **Right-click the app folder, select New > Folder > Assets Folder, and then click the Finish button**.

    NOTE: This will create an empty assets folder.

17. **Right-click on the newly created assets folder, select Directory, type www into the text field of the New Directory dialog box, and then click the OK button**.

    NOTE: This will create a www directory.

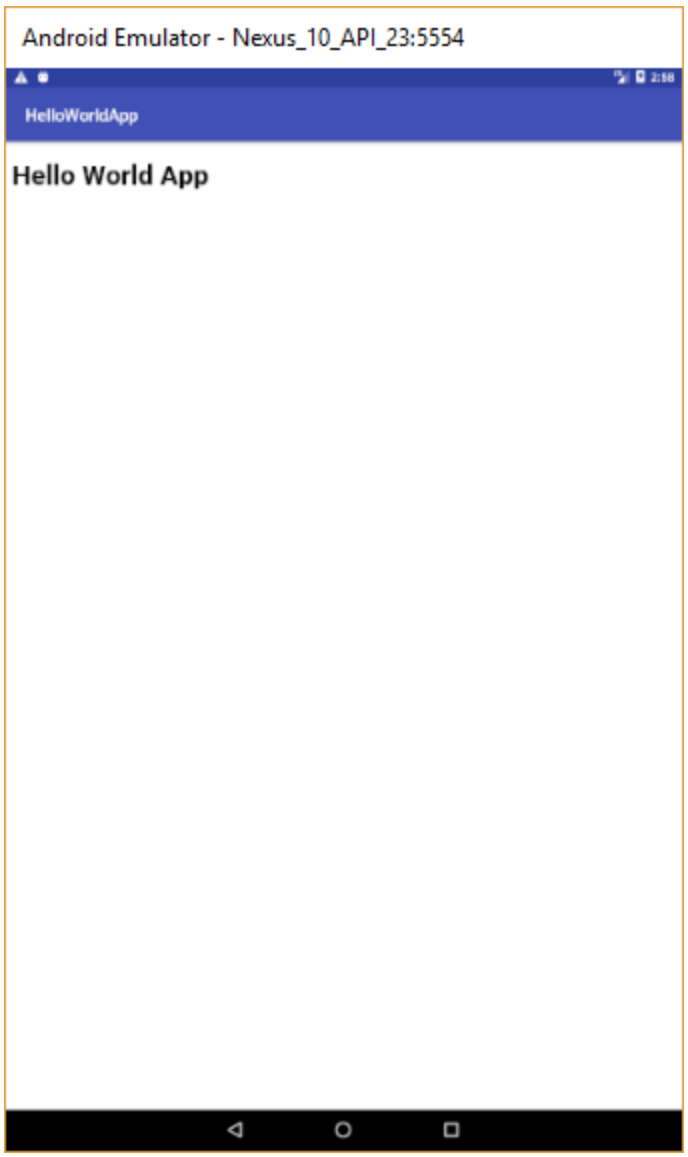
18. **Right-click the newly created www directory, select New > File, type index.html into the text field of the New File dialog box, and then click the OK button**.

    NOTE: This will create the index.html file.

19. **Write the following HTML code into the index.html** file that appear in one of the tabs.

```
<!DOCTYPE html>
<html>
<body>
  <h1>Hello World App</h1>
</body>
</html>
```

20. **CHECK POINT: Click the Run button** to run the app on a mobile device or in an emulator to see if it works with the default settings. You should see the app loads with the phrase Hello World App as a title.



21. **Write the following highlighted HTML code into the index.html** file.

```
<!DOCTYPE html>
<html>
<head>
  <script src="app.js"></script>
</head>
<body>
```

```
<h1>Hello World App</h1>

<label for="name">Enter Name:</label>
<input id="name" value="" placeholder="Enter your name"/>

<button onclick = "myMsg()">Display Message</button>
<br><br>
<button><a href="page2.html">Page 2</a></button>

</body>
</html>
```

22. **Right-click the www folder AGAIN, select New > File, type app.js into the New File dialog box that appears, and then click the OK button**.

23. **Type the following JavaScript code into the app.js file** that appears:

```
function myMsg()
{
  alert("Hello, " + document.getElementById("name").value + "!!");
};
```

24. **Right-click the www folder AGAIN, select New > File, type page2.html into the New File dialog box, and then click the OK button**.

25. **Write the following HTML code into the page2.html file** that appears.

```
<html>
<body>
<h1>Page 2</h1>
<button><a href="index.html">Go to Home Page</a></button>
</body>
</html>
```

26. **Click the Run button, select a device either from the Connected Devices (if you have a device connected to your computer) or Available Emulators, and then click the OK button**.

27. **CHECKPOINT:** You should see the app load on your device or emulator. Test it to see if it is working correctly by:
    1. Entering a name in the Enter Name text field and then Clicking the Display Message button to see an alert with a message display.

2. Clicking the Page 2 button to see Page 2 displayed and then clicking the Go To Home Page button to go back to the home page.

   NOTE: If you navigate to Page 2 and then click the Back button on the emulator or your mobile device, you may be taken to a previous loaded app and not the current app home page. This will be resolved in the upcoming steps.

28. In the MainActivity.xml file **BELOW** the closing curly brace of the MainActivity class, **type the following highlighted word (onBack) and then press the ENTER key**:

```
    view.setWebChromeClient(new WebChromeClient());
}
onBack
```

29. **CHECK POINT:** You should see that the word expanded to the following highlighted code:

```
    view.setWebChromeClient(new WebChromeClient());
}

@Override
public void onBackPressed() {
    super.onBackPressed();
}
```

30. **Modify the newly created code to the following highlighted if/else statement**:

```
@Override
public void onBackPressed() {

if(view.canGoBack())
{
  view.goBack();
}
else
{
super.onBackPressed();
}

}
```

31. Also, if you want to remove the app title, a**dd the following highlighted code:**

    protected void onCreate(Bundle savedInstanceState) {
      requestWindowFeature(Window.*FEATURE_NO_TITLE*);
      super.onCreate(savedInstanceState);

32. If necessary, you may need to change the extension:

    From:
    public class MainActivity extends **AppCompatActivity** {

    To:
    public class MainActivity extends **Activity** {

33. **Click inside the red word Activity and press ALT+ENTER.**
34. **Click the Run button, select a device either from the Connected Devices (if you have a device connected to your computer) or Available Emulators, and then click the OK button**. You should see that the title has been removed.

PHASE 4: Add Web Site

Now that we have the default content working, replace it with your own web content in the **www** folder.

1. **Delete the HTML files** (index.html and page2.html) **and the JavaScript file** (app.js).
2. **MOVE your web content to the www folder**.

   TIP: It may be easier to move the content from outside of Android Studio and then return.

3. **CHECK POINT:** Click the Run button to run the app on a mobile device or in an emulator to see if it works as expected.

SEE PHASE 2-6 FROM OPTION 1 or use Android Studio to create a debug and release version of your app.

BONUS: (OPTIONAL) Create App and View It in a Browser Within AS

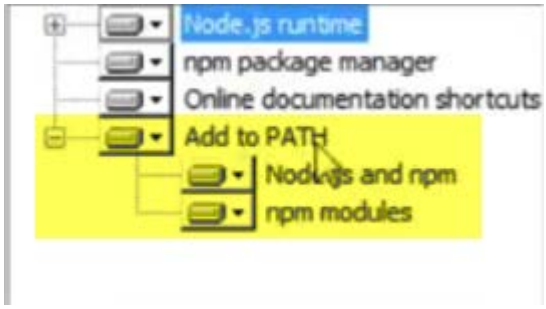
As an alternative to the previous method, you can download a Cordova/PhoneGap plugin and create a stub app at the command prompt, and then open it up and view it not only in Android Studio with a mobile device or emulate, but also within a browser.

Install node.js and Android Studio

1. **Download and run node.js** if you do not already have it installed.

   NOTE: Ensure when installing that the Add to PATH option is set for both Node.js and npm

and npm modules.



2. **From the Start menu, right-click Computer, select Properties, click the Advanced system settings, and then click the Environment Variables button.**
3. **Ensure that the JAVA_HOME is set to the proper folder** (e.g., C:\Program Files\Java\jdk1.8.0_144), if not, add it by clicking the New…. Button and enter:
    a. Variable name: **JAVA_HOME**
    b. Variable value: **C:\Program Files\Java\jdk1.8.0_144** or the latest version
    CAUTION: Ensure that you do not use the **bin** folder.
4. **Click the OK button** to commit the changes.
5. **Open command prompt**, **type the following statement, and then press the Enter key** to install Cordova:

    npm install -g cordova

Create project

1. **Create a new folder** on your desktop or wherever you want (e.g., HelloWorldApp) and then **copy the path to the clipboard.**
2. **Type the following to create a new cordova project** and then **add a space.**

    cordova create

3. **Paste the path** (e.g., C:\Users\Cornelius\Desktop\HelloWorldApp) and then **add a space**.

    cordova create **C:\Users\Cornelius\Desktop\HelloWorldApp**

    NOTE: Replace the path name with your own system path name.

4. **Type the package name** (e.g., com.richmediacs.HelloWordApp) and **add a space**.

    cordova create C:\Users\Cornelius\Desktop\HelloWorldApp **com.richmediacs.HelloWordApp**

NOTE: Replace the package name with your own package name.

5. **Type the app name** (e.g., com.richmediacs.HelloWordApp)

   cordova create C:\Users\Cornelius\Desktop\HelloWorldApp com.richmediacs.HelloWordApp **HelloWorldApp**

   NOTE: Replace the app name with your own app name.

6. Press the **ENTER** key to create the project.
7. **CHECK POINT:** Go to your project folder (e.g., C:\Users\Cornelius\Desktop\HelloWorldApp)) to see the folders and files that were created.

Install Cordova/Phonegap in Android Studio

1. Open Android Studio for any previous project, **select Files, and then Settings from the menu**.
2. **Click the Plugin** option and in the Settings dialog box that appears, **click the install JetBrain plugin… button**.
3. **In the Search field, type PhoneGap/Cordova Plugin and then click the Install button.**
4. **Click the Restart Android Studio button.**
5. **Click Apply, OK, and then the Restart buttons.**

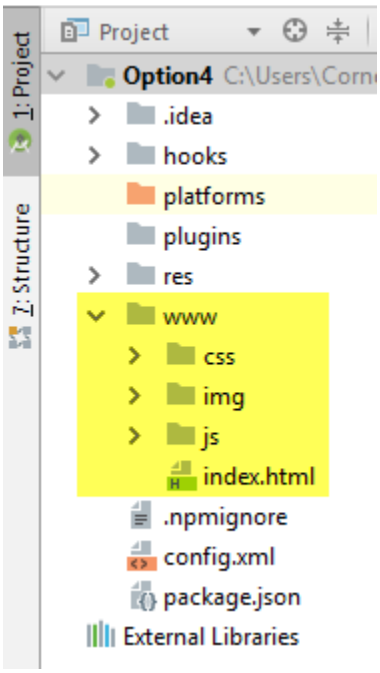
Create App in Android Studio

1. **Once Android Studio restart, select File > Open and navigate to the project folder that was created earlier (e.g., HelloWorldApp), and then click the OK button.**

   NOTE: Notice that the platforms folder is currently empty.

2. In the dialog box that appears, **click the This Window button**.
3. **Change the view from Android to Project.**

   NOTE: Since this is not a native Android app, you will have to change the view to see the file structure.

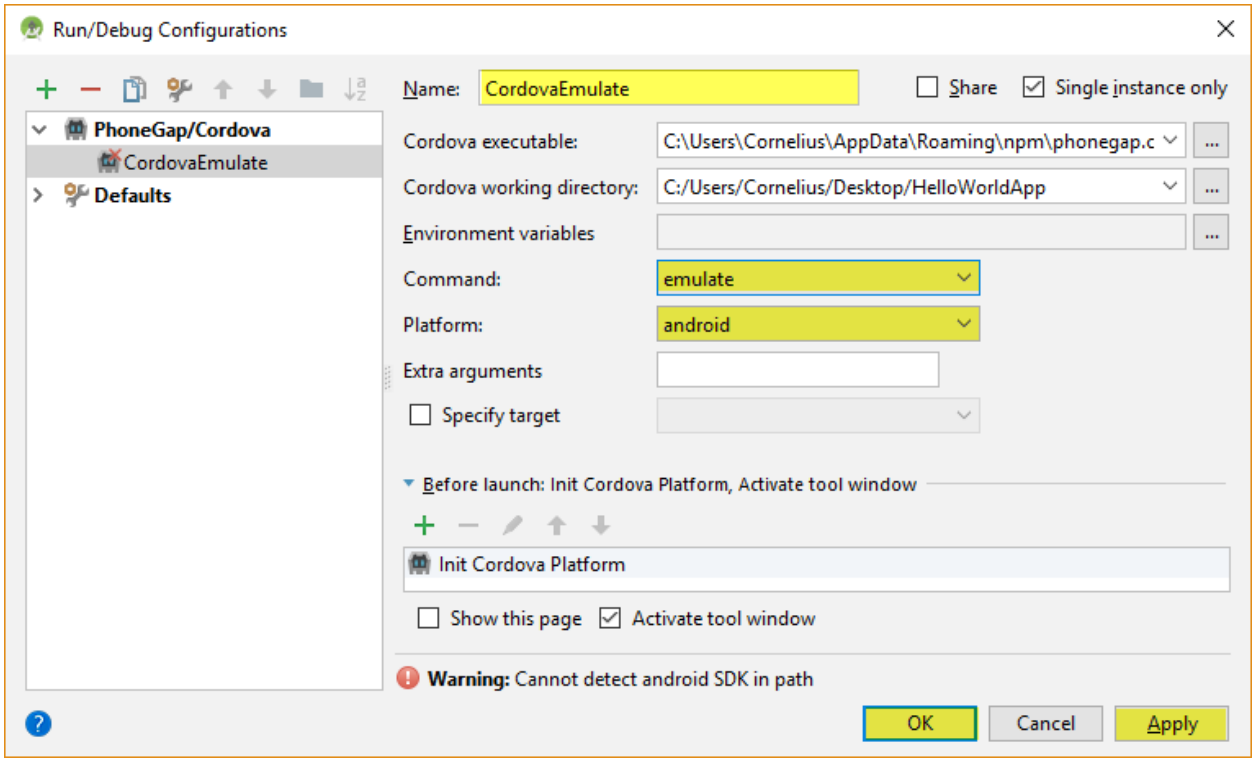


4. **Click the Edit Configuration… button** (to the right of the Run button)

   NOTE: Since this is not an Android project, we need to use this plugin.

5. In the dialog box that appears, **click the plus button (+)** and **select the PhoneGap/Cordova configuration from the list**.

6. In the Run/Debug Configurations dialog box dialog box that appears:
   - **Give the configuration a name** (e.g., CordovaEmulate).
   - **From the Command drop-down, select run** for mobile devices (or **emulate** for virtual devices).
   - **From the Platform drop-down, select android**.
   - **Click Apply** and then **click the OK button**.



7. **Click the Run button.**
8. **CHECK POINT:** If you view the platform folder, you will see that the **android** platform has been added. You should see the app running on your mobile device or emulator.
9. **Make any minor changes to the index.html file below the following tag**.

   Connecting to device
   <h1>Hello World</h1>

10. **Run the app again.**
11. **CHECK POINT:** You should see the change you made displayed in the app.
12. **Click the Edit Configuration… button again** and **change the Platform drop-down to browser**.
13. **Click Apply and then click the OK button**.
14. **Click the Run button again**.
15. **CHECK POINT:** This time you should see the app opens in a browser. If you look in the platforms folder, you will see another folder called browser with content in it.

Resources

Splash Screen Generators

- Go Ape Tools - Generate the many sizes of icons and splash screens (launch images) your app will require in order to get your app published to all of the major app ...
- PhoneGap Icon & Splashscreen generator

- AlexDisler/cordova-splash - Create a splash screen once in the root folder of your Cordova project and use cordova-splash to automatically crop and copy it for all the platforms your project supports (currenty works with iOS, Android and Windows 10).
- Generating Splash Screens and Application Icons for NativeScript ...
- splash-screen-generator - npm - Easily generate *splash screens* to use in your PWA's.
- Abiro PhoneGap Image Generator - Upload a preferably square app icon and an optional *splash screen* background in a high resolution bit image format, but preferably PNG.
- TiCons - Generate all icon & splash screens for your Titanium app ...
- Automating Icons and Splash Screens | The Official Ionic Blog - An app icon and *splash screen* (launch image) are important parts of any app, yet ...
- iOS App icon generator - make app icons and xCode launch images. Compatible with iOS11 and Android. Just drag & drop into your project. Free app icon *generator* and *splash screen* resizer. Now supports Apple Watch.
- Automatically Generate Splash Screens and Icons with Ionic CLI ... - In this video tutorial, I will walk you through how to create the initial icon and *splash screen*, and how to use the ionic resources command.

How to Convert A Website into a Mobile App Using Android Studio

- How to Convert a Website into Android Application using Android Studio
- How to Convert a Website into Android App using AS
- How to Convert Web into App on Android Studio 2.3.1 in 2017 and add Admob Ads full Tutorial

How to use HTML, CSS and JS and convert it to an app

- ******** How to build Android App with HTML5/CSS/JavaScript
- **** Install Cordova, Import to Android studio to Deploy using PhoneGap Cordova Plugin
- **** How to display HTML Files On Android Applications Locally With Android Studio