1. Working with text

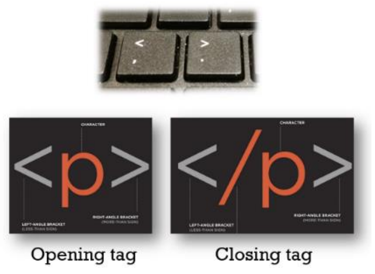
Structure

Many web apps act like electronic versions of documents such as Newspapers, which has headline, some text, and possibly some images. If the article is a long piece, there may be subheadings that split the story into separate sections or quotes from those involved. Structure helps readers understand the stories in the newspaper.

Structure is a way to build a web app to organize and make it easier for the reader to follow what he/she is reading. HTML code builds the structure or layout of a web app.

The HTML code

The HTML code is made up of characters that live inside angled brackets, which are called HTML **elements**. Elements are usually made up of two **tags:** an opening tag and a closing tag.



Each HTML element tells the browser something about the information that sits between its opening and closing tags.

Basic HTML elements

The basics HTML elements are:

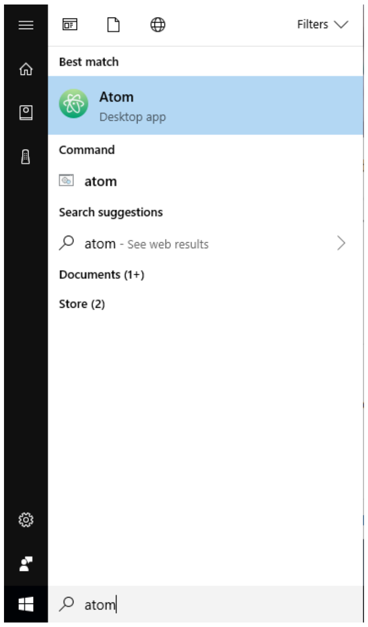
1. The opening **<html>** tag element indicates that anything between it and a closing **</html>** tag is HTML code
2. A **<head>** element contains information about the page such as title
3. The contents of the **<title>** element are either shown in the top of the browser, above where you usually type in the URL of the page you want to visit, or on the tab for that page (if your browser uses tabs to allow you to view multiple pages at the same time).
4. The **<body>** tag indicates that anything between it and the closing tag </body> should be inside the main browser window.

Class Activity 1

Activity 1 we are going to create our first app view using HTML and check our code through google Chrome browser.

Steps:

1. We start by writing the HTML code using Atom.



2. In Atom, open a new file and save it with your last name and the file extension .html. Make sure that you save the file in your local drive or computer Desktop.
3. The first line that we write in an html file is <html> tag to declare the html file. One of the great thing about Atom is that when you type a tag name, it will automatically build a basic code structure of the code. Write html and click enter and Atom will create the following code:

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>

  </body>
</html>
```

The **`<!DOCTYPE>`** declaration must be the very first thing in your HTML document, before the <html> tag and it specifies the rules for the markup language, so that the browsers render the content correctly.

The attribute **lang** indicates the language and the value **en** means English. This line specifies that the html code is based in English.

The attribute **dir** indicates the direction of the text and the value **ltr** indicates that the text will be display from left-to-right.

Metadata is data (information) about data. The **<meta>** tag provides metadata about the HTML document. Metadata will not be displayed on the page, but will be machine parsable.

Meta elements are typically used to specify page description, keywords, author of the document, last modified and other metadata

UTF-8 is the Unicode character sets from 1 to 4 bytes long. UTF-8 can represent any character in the Unicode standard. UTF-8 is backwards compatible with ASCII and the encoding for e-mail and web pages.

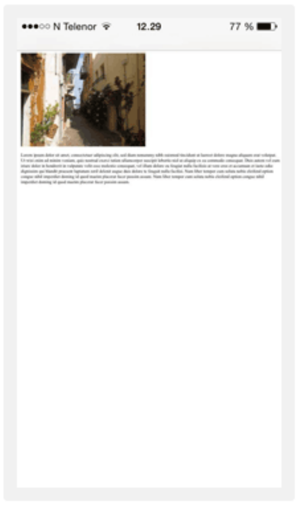
For app view development, it is an important to add the following line:

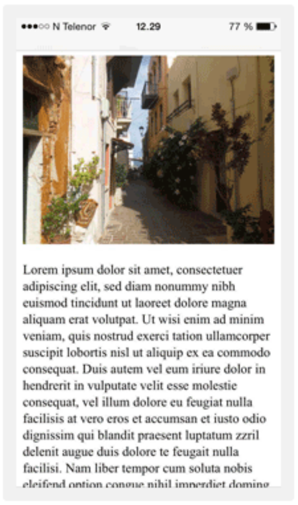
```html
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

The **viewport** is the user's visible area of a web page. It varies with the device, and will be smaller on a mobile phone than on a computer screen. viewport element gives the browser instructions on how to control the page's dimensions and scaling.

The **width=device-width** part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The **initial-scale=1.0** part sets the initial zoom level when the page is first loaded by the browser.





Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming

Without the viewport meta tag **With the viewport meta tag**

Once we have the **`<head>`** element sets, we can use some tags in the **`<body>`.** Most of the tags that we use in **`<body>`** are visible in the web app. We are try to add some heading the body.

HTML has six levels of headings:

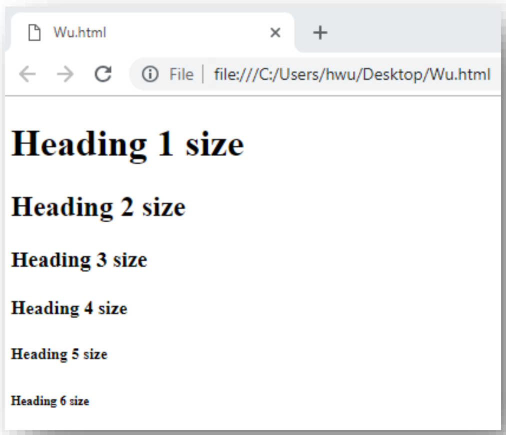
- <h1>is used for main headings
- <h2> is used for subheadings.
- If there are further sections under the subheadings then the <h3>element is used, and so on...

HTML

```
<h1>Heading 1 size</h1>
<h2>Heading 2 size</h2>
<h3>Heading 3 size</h3>
<h4>Heading 4 size</h4>
<h5>Heading 5 size</h5>
<h6>Heading 6 size</h6>
```

To see the output of the html code, we can go to the location of our html file, and double click on html icon

The html file will open in internet browser



Heading 1 size

Heading 2 size

Heading 3 size

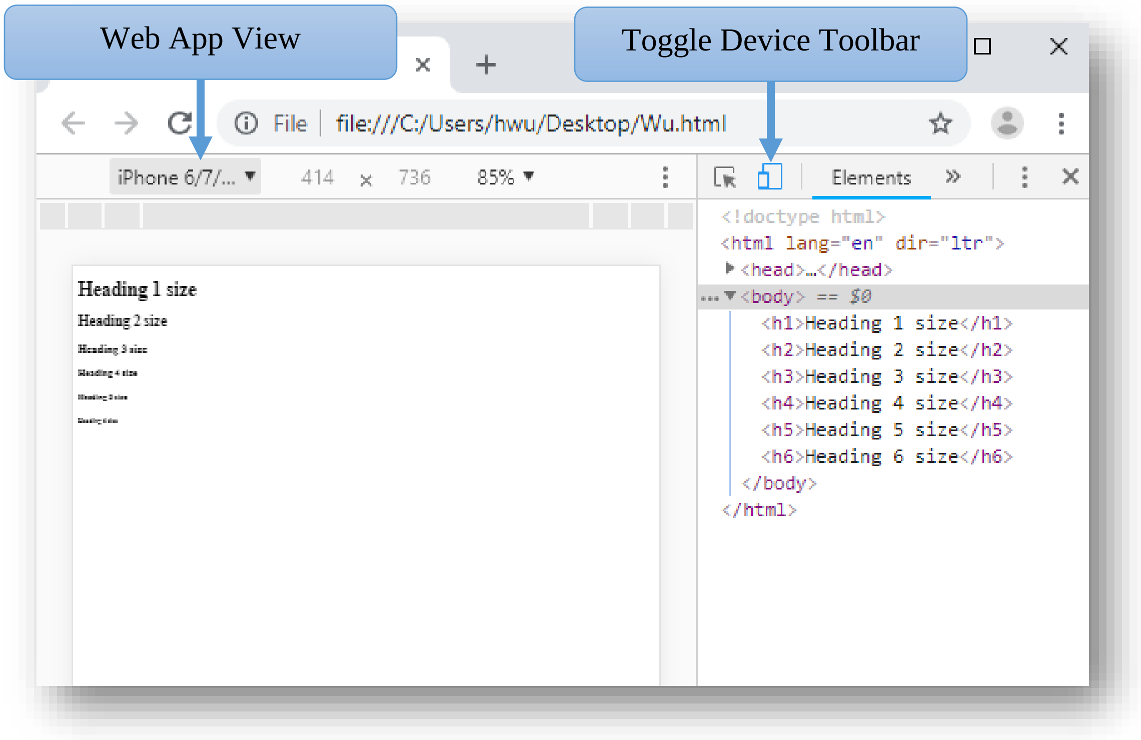
Heading 4 size

Heading 5 size

Heading 6 size

Since we are creating web app, we can use the browser development tools. To get to the development tools in Google Chrome, we just hit the function key F12 from our keyboard

In the development window, we can select the "Toggle Device Toolbar", and select the size of the app view.



```
<!doctype html>
<html lang="en" dir="ltr">
  <head>…</head>
  <body> == $0
    <h1>Heading 1 size</h1>
    <h2>Heading 2 size</h2>
    <h3>Heading 3 size</h3>
    <h4>Heading 4 size</h4>
    <h5>Heading 5 size</h5>
    <h6>Heading 6 size</h6>
  </body>
</html>
```

Now that we have learn the basic set up of an app view, we can try more html tags.

More elements

Paragraph

The <**p**> **tag** specifies a paragraph of text. It is a block-level element and it automatically have margin before and after the tag.

```
<p>Queensborough Community College - QCC</p>
```

Bold

By enclosing words in the tags <**b**> and </**b**> we can make characters appear bold.

```
<p>Inside a product description you might see some <b> key features</b>
in bold.</p>
```

Italic

By enclosing words in the tags <**i**> and </**i**> we can make characters appear italic.

```
<p>My cousin is reading <i>The Adventures of Tom Sawyer</i> for two
classes: <i><b> English </i></b> and <i><b> History</i></b>.</p>
```

Line Breaks

The browser will automatically show each new paragraph or heading on a new line. But if you wanted to add a line break inside the middle of a paragraph you can use the line break tag <**br**/>

```
<p>The Earth<br/>gets one hundred tons heavier every day <br/>due to
falling space dust. </p>
```

Horizontal Rules

To create a break between themes — such as a change of topic in a book or a new scene in a play — you can add a horizontal rule between sections using the <**hr**/>

```
<p>Venus is the only planet that rotates clockwise.</p>
<hr /> <p>Jupiter is bigger than all the other planets combined.</p>
```

Strikethrough

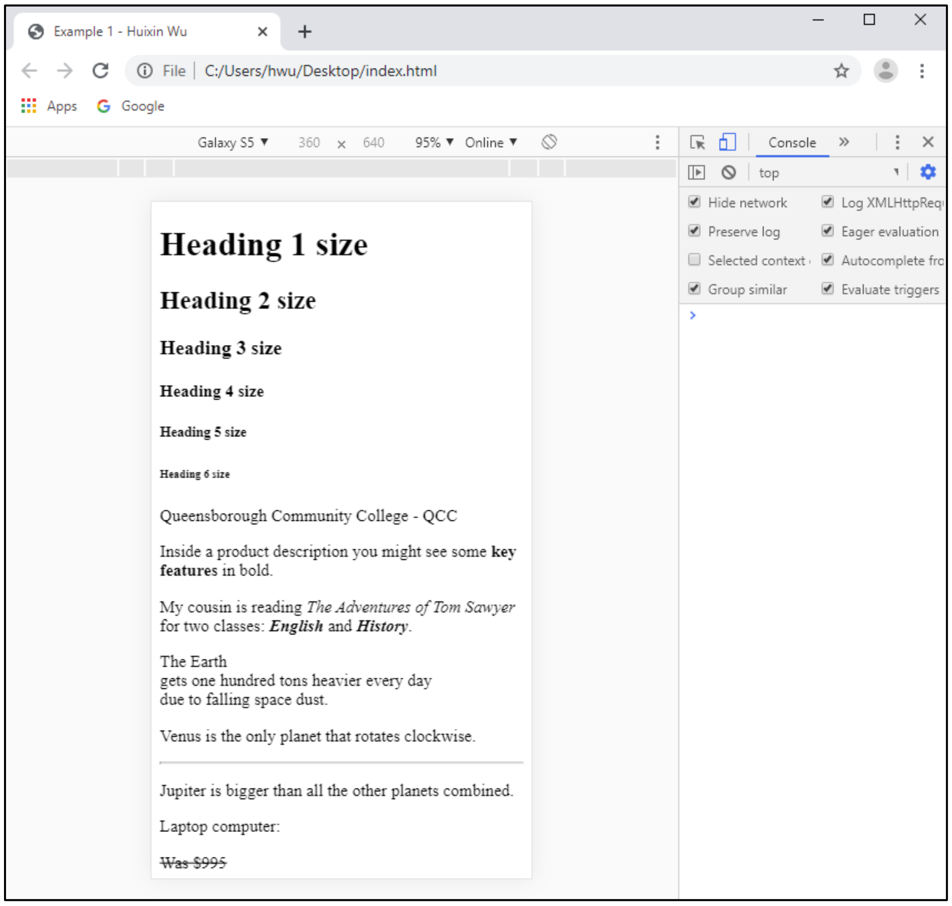
Strikethought means to cross something out by drawing a line through it.

The <**s**> element indicates something that is no longer accurate or relevant (but that should not be deleted).

Visually the content of an <**s**> element will usually be displayed with a line through the center.

```
<p>Laptop computer:</p> <p><s>Was $995</s></p> <p>Now only $375</p>
```

If we refresh the internet browser, the output will look as:



Heading 1 size

Heading 2 size

Heading 3 size

Heading 4 size

Heading 5 size

Heading 6 size

Queensborough Community College - QCC

Inside a product description you might see some **key features** in bold.

My cousin is reading *The Adventures of Tom Sawyer* for two classes: ***English*** and ***History***.

The Earth
gets one hundred tons heavier every day
due to falling space dust.

Venus is the only planet that rotates clockwise.

Jupiter is bigger than all the other planets combined.

Laptop computer:

~~Was $995~~

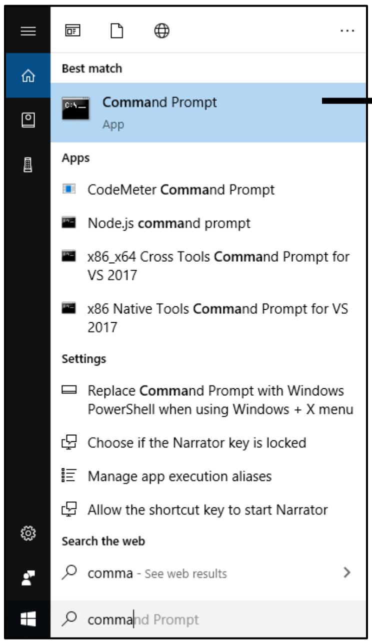
Now is time to convert a web app into a smartphone app using cordova and Android Studio.

How to convert a web app into a native app using Cordova

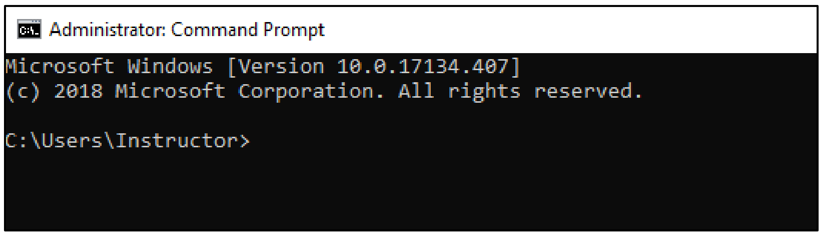
To convert a web app into a Cordova app, we have to make sure that the loading HTML file must be named *index.html*

Once you have Cordova installed and the web app ready to convert, we:

Step 1) Open the Command prompt

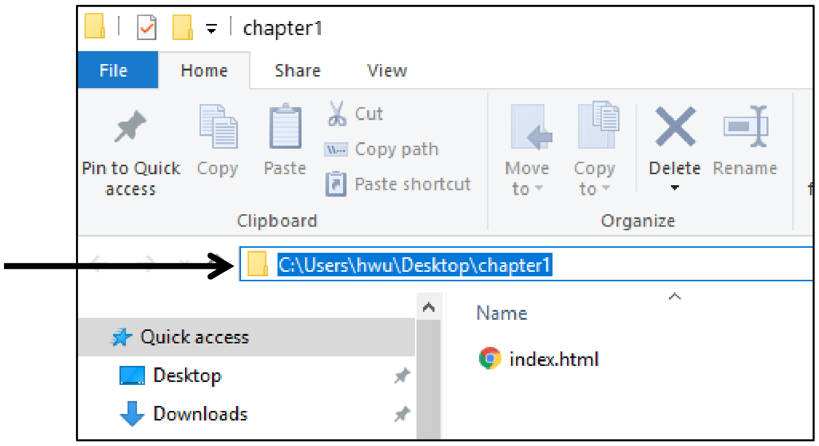


Best match
Command Prompt
App
Apps
CodeMeter Command Prompt
Node.js command prompt
x86_x64 Cross Tools Command Prompt for VS 2017
x86 Native Tools Command Prompt for VS 2017
Settings
Replace Command Prompt with Windows PowerShell when using Windows + X menu
Choose if the Narrator key is locked
Manage app execution aliases
Allow the shortcut key to start Narrator
Search the web
comma - See web results
command Prompt

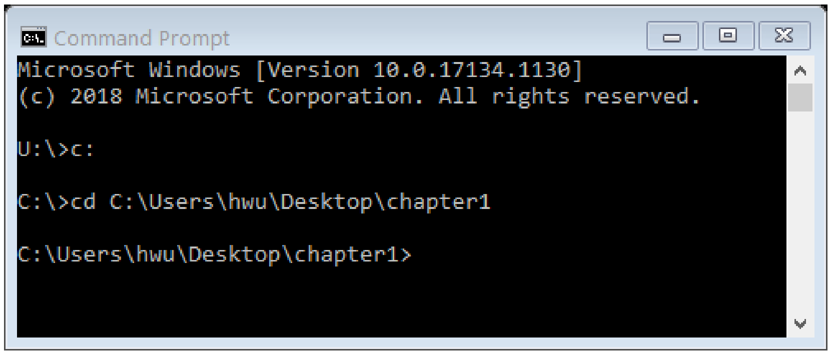


```
Microsoft Windows [Version 10.0.17134.407]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Instructor>
```

Step 2) At the command prompt, we can go to our project folder by using the *cd (current directory)* command. If we don't know our project folder location, we can open the project folder, click on the direction, and copy the location.



After it, we can go to the command prompt, type ***cd*** space, paste the location of the project folder, and click enter.

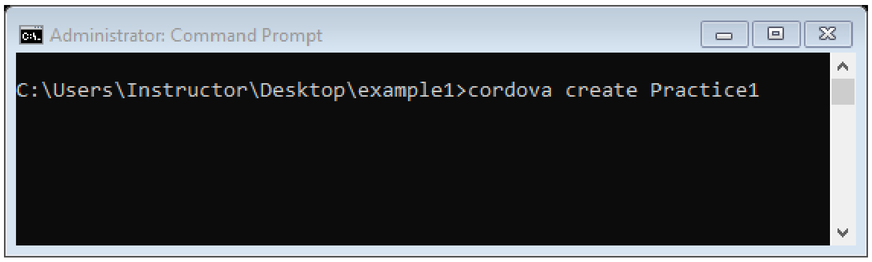


```
Microsoft Windows [Version 10.0.17134.1130]
(c) 2018 Microsoft Corporation. All rights reserved.

U:\>c:

C:\>cd C:\Users\hwu\Desktop\chapter1

C:\Users\hwu\Desktop\chapter1>
```

Step 3) Once at the folder, we need to create a Cordova project folder by using the following command line:

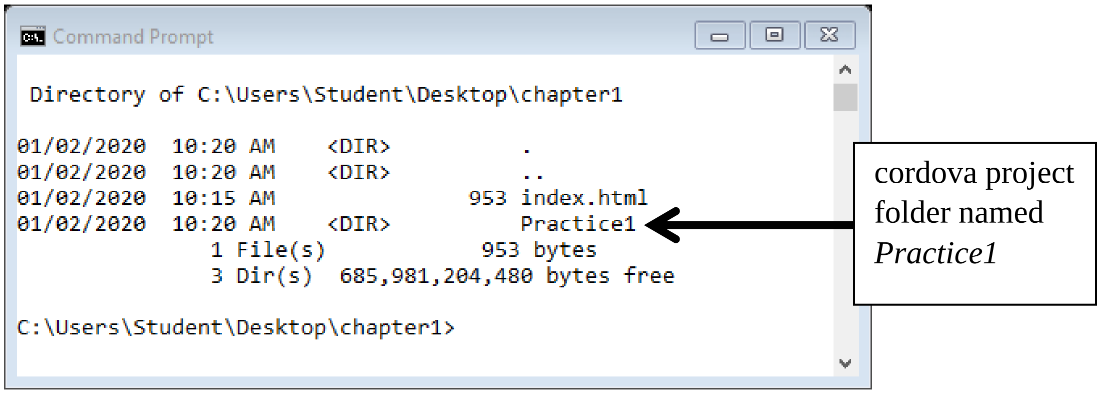
Folder name

cordova create Practice1



```
C:\Users\Instructor\Desktop\example1>cordova create Practice1
```

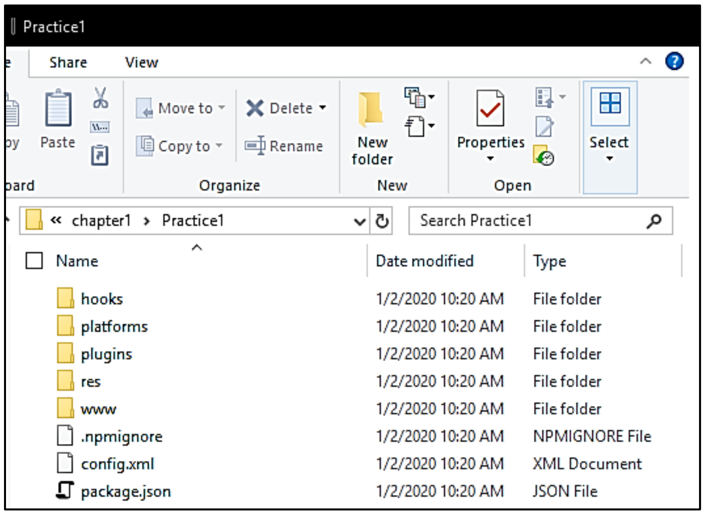
We can check if the cordova project folder is created by typing *dir*



```
 Directory of C:\Users\Student\Desktop\chapter1

01/02/2020  10:20 AM    <DIR>          .
01/02/2020  10:20 AM    <DIR>          ..
01/02/2020  10:15 AM               953 index.html
01/02/2020  10:20 AM    <DIR>          Practice1
               1 File(s)            953 bytes
               3 Dir(s)  685,981,204,480 bytes free

C:\Users\Student\Desktop\chapter1>
```

cordova project folder named *Practice1*

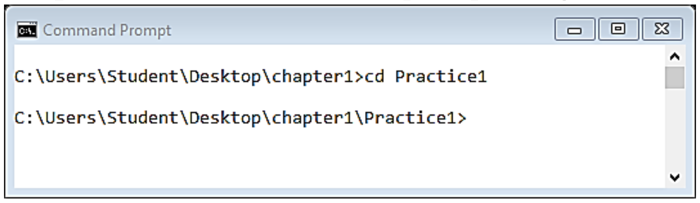
If we open the project folder from the windows interface, we can also see that the cordova project folder is already created:



Name	Date modified	Type
hooks	1/2/2020 10:20 AM	File folder
platforms	1/2/2020 10:20 AM	File folder
plugins	1/2/2020 10:20 AM	File folder
res	1/2/2020 10:20 AM	File folder
www	1/2/2020 10:20 AM	File folder
.npmignore	1/2/2020 10:20 AM	NPMIGNORE File
config.xml	1/2/2020 10:20 AM	XML Document
package.json	1/2/2020 10:20 AM	JSON File

Step 4) We can copy all the project web app files and paste them into the *www* folder inside of the cordova project folder *Practice1*.

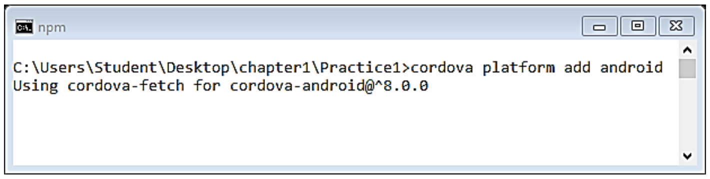
Step 5) Back to the command prompt, we go into the Practice1 folder by using typing:



```
C:\Users\Student\Desktop\chapter1>cd Practice1

C:\Users\Student\Desktop\chapter1\Practice1>
```

cd Practice1

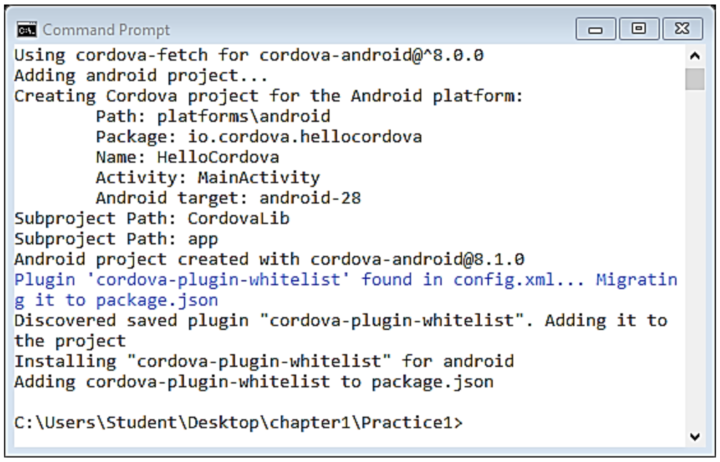
Step 6) Inside the Practice1 folder, we are going to add an android or iOS platform to our Cordova project by typing:



```
C:\Users\Student\Desktop\chapter1\Practice1>cordova platform add android
Using cordova-fetch for cordova-android@^8.0.0
```

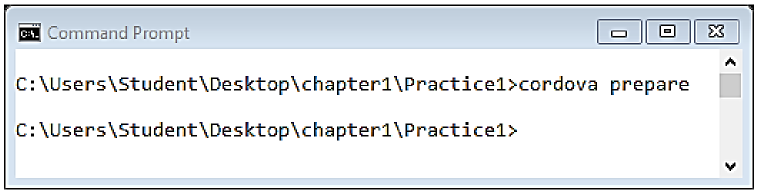
cordova platform add android

Once we clicked enter, cordova will run and create the platform.



```
Using cordova-fetch for cordova-android@^8.0.0
Adding android project...
Creating Cordova project for the Android platform:
        Path: platforms\android
        Package: io.cordova.hellocordova
        Name: HelloCordova
        Activity: MainActivity
        Android target: android-28
Subproject Path: CordovaLib
Subproject Path: app
Android project created with cordova-android@8.1.0
Plugin 'cordova-plugin-whitelist' found in config.xml... Migratin
g it to package.json
Discovered saved plugin "cordova-plugin-whitelist". Adding it to
the project
Installing "cordova-plugin-whitelist" for android
Adding cordova-plugin-whitelist to package.json

C:\Users\Student\Desktop\chapter1\Practice1>
```

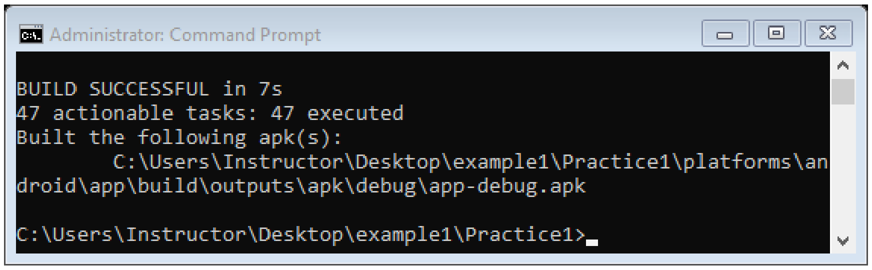
Step 7) Once the platform is created, we need to prepare and compile the cordova project Practice1. Type ***cordova prepare***



```
C:\Users\Student\Desktop\chapter1\Practice1>cordova prepare

C:\Users\Student\Desktop\chapter1\Practice1>
```

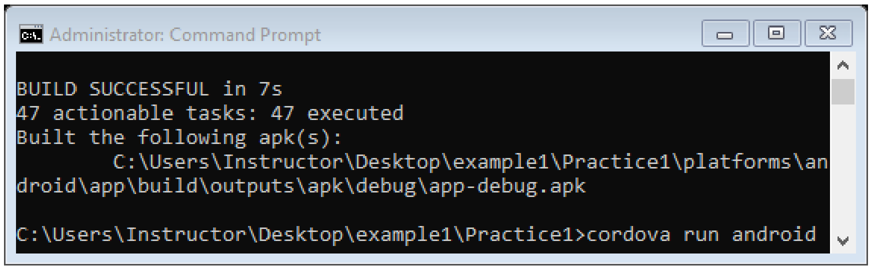
Step 8) Now we are going to compile the cordova project by typing: ***cordova compile***

Once we clicked enter, cordova will run the compiling and at the end of the compiling, the command prompt will show BUILD SUCCESSFUL



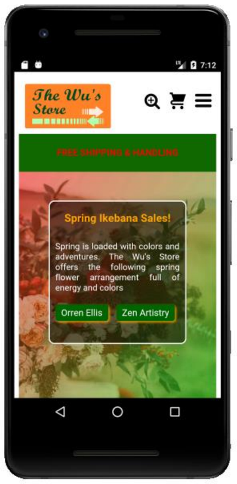
```
BUILD SUCCESSFUL in 7s
47 actionable tasks: 47 executed
Built the following apk(s):
        C:\Users\Instructor\Desktop\example1\Practice1\platforms\an
droid\app\build\outputs\apk\debug\app-debug.apk

C:\Users\Instructor\Desktop\example1\Practice1>_
```

Step 9) After compiling, we can run our project in an emulator by typing: **cordova run android**



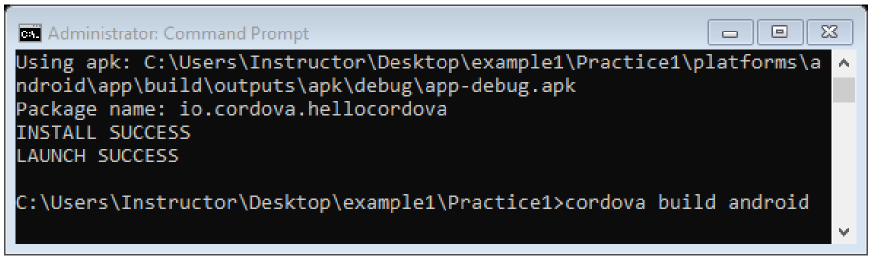
```
BUILD SUCCESSFUL in 7s
47 actionable tasks: 47 executed
Built the following apk(s):
        C:\Users\Instructor\Desktop\example1\Practice1\platforms\an
droid\app\build\outputs\apk\debug\app-debug.apk

C:\Users\Instructor\Desktop\example1\Practice1>cordova run android
```

This command line will run the emulator from Android Studio



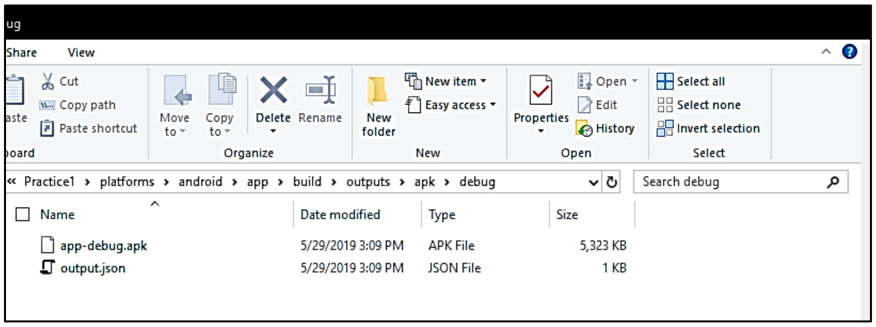
Spring is loaded with colors and adventures. The Wu's Store offers the following spring flower arrangement full of energy and colors

Step 10) We can also create an Android package .apk, by typing: **cordova build android**



```
Using apk: C:\Users\Instructor\Desktop\example1\Practice1\platforms\a
ndroid\app\build\outputs\apk\debug\app-debug.apk
Package name: io.cordova.hellocordova
INSTALL SUCCESS
LAUNCH SUCCESS

C:\Users\Instructor\Desktop\example1\Practice1>cordova build android
```

Once the .apk is build, we can find it at:
example1\Practice1\platforms\android\app\build\outputs\apk\debug



« Practice1 › platforms › android › app › build › outputs › apk › debug

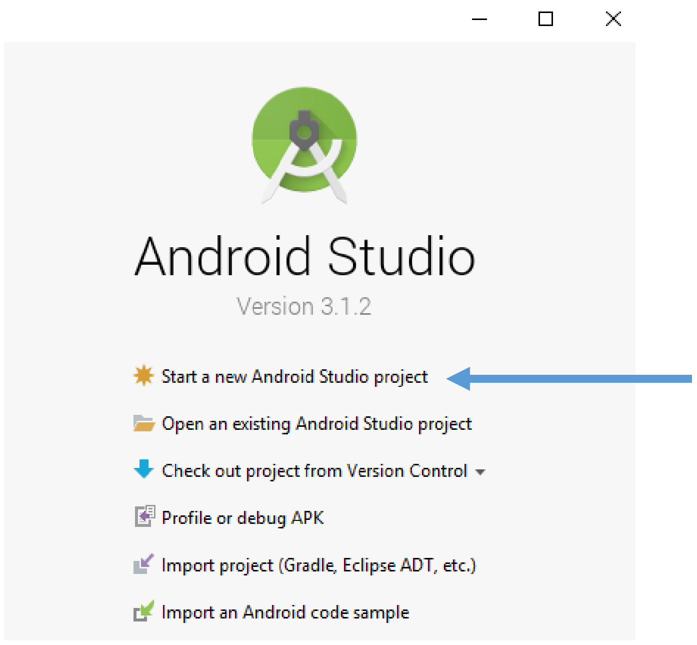
Name	Date modified	Type	Size
app-debug.apk	5/29/2019 3:09 PM	APK File	5,323 KB
output.json	5/29/2019 3:09 PM	JSON File	1 KB

For more information about the cross-platform conversion using Cordova, you can check the following YouTube video ➔ Cordova: How to Create, Build, Add Platform and Run an html file into Android and iOS emulator.

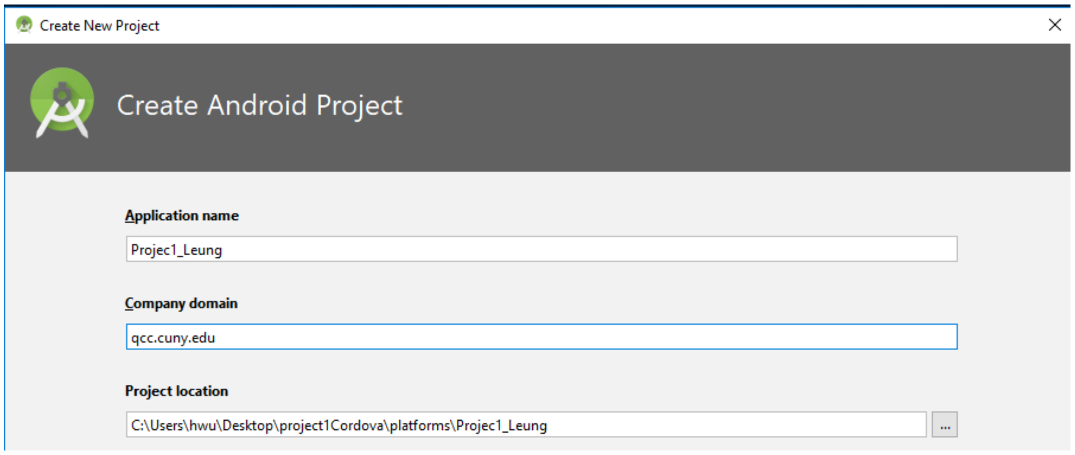
https://www.youtube.com/watch?v=cxJAiUeb5bQ

How to convert a web app to android app

Step 1) Open Android Studio and select Start a new Android Studio project


Android Studio
Version 3.1.2
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample

Step 2) If you want, you can change the Application name and the Company domain. Those two information is to Android Studio to create a java package under the Company domain name and the main java class to the Application name.

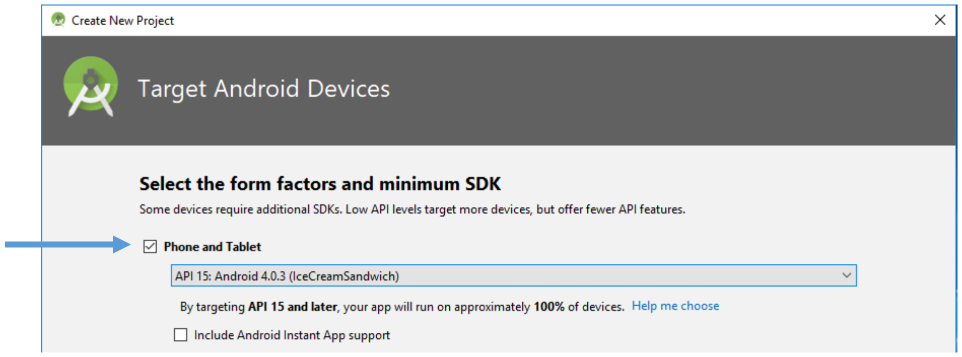


Application name
Projec1_Leung

Company domain
qcc.cuny.edu

Project location
C:\Users\hwu\Desktop\project1Cordova\platforms\Projec1_Leung

Step 3) You can also select the format of the app that you are creating. By default Android Studio has it selected as to ***Phone and Tablet***, and from the dropdown list you can select the android version of the mobile devices.


Create New Project

Target Android Devices

Select the form factors and minimum SDK

Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

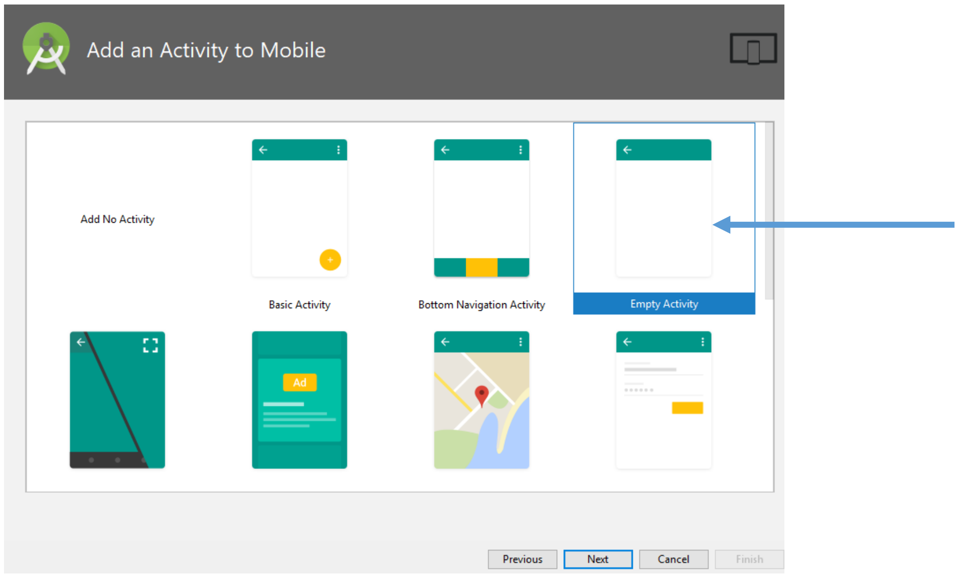
☑ Phone and Tablet

API 15: Android 4.0.3 (IceCreamSandwich)

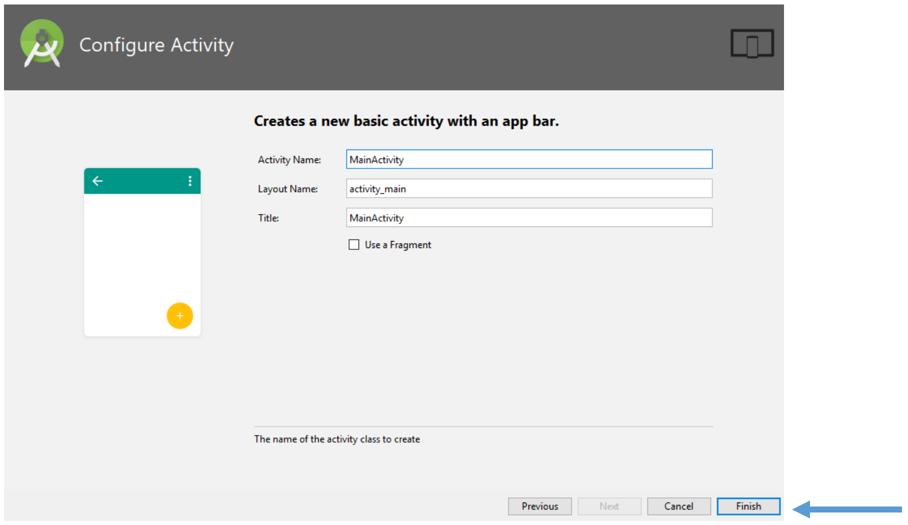
By targeting **API 15 and later**, your app will run on approximately **100%** of devices. Help me choose

☐ Include Android Instant App support

Step 4) In the Add an Activity to Mobile window, since we are converting a web app into an Android app, we can select Basic Activity or Empty Activity.



Step 5) In the Configure Activity window, the Activity Name is the name of the main java class.


Configure Activity

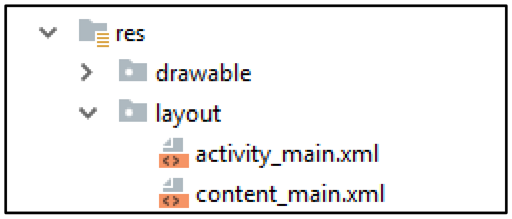
Creates a new basic activity with an app bar.

Activity Name:	MainActivity
Layout Name:	activity_main
Title:	MainActivity

☐ Use a Fragment

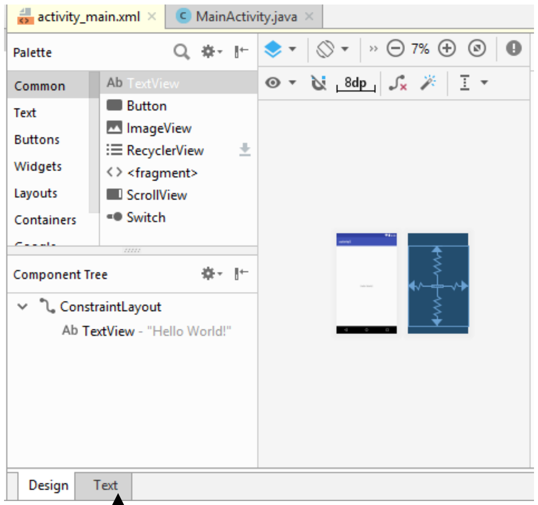
The name of the activity class to create

Step 6) Once in Android Studio, you can delete the Hello World! Text view from the emulator first. You can go to *res* folder res , open the *layout* folder and double click on *activity_main.xml*



```
res
  drawable
  layout
    activity_main.xml
    content_main.xml
```

Step 7) Click on **Text** tab. In the xml file, delete the `<TextView />` tag and create a `WebView` and id it as ***`webView`***.

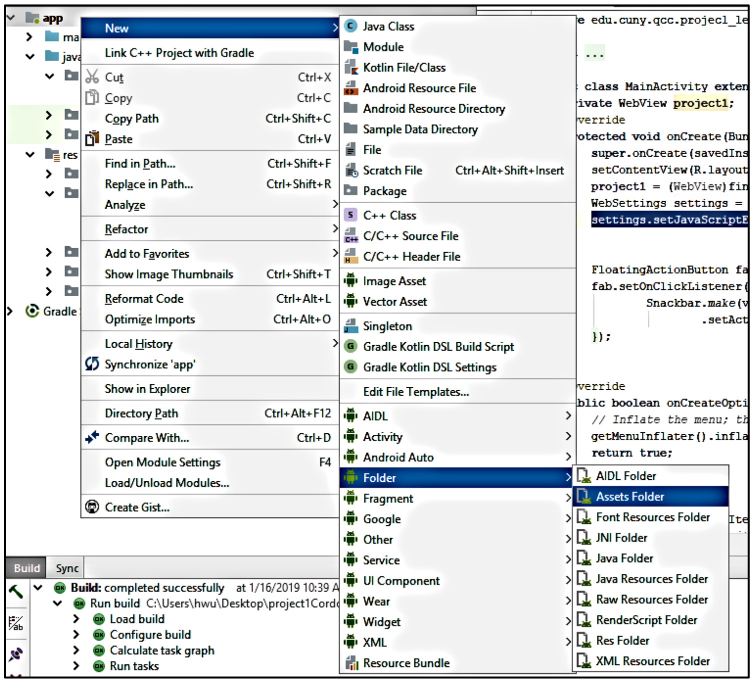


Click **Text** tab

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <WebView
        android:id="@+id/webView"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
    </WebView>

</android.support.constraint.ConstraintLayout>
```

Add a ***WebView*** to display the web app in the android window

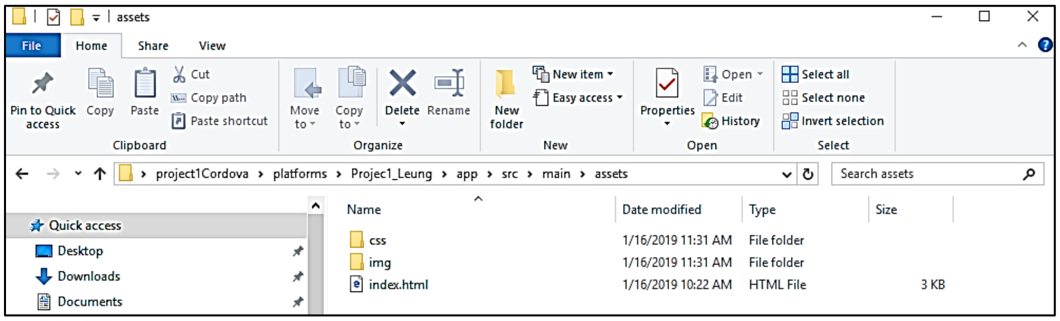
Step 8) Now, we are going to create a Assets folder where we are doing to store all html and css project files. Right click on the app icon app , select New ➔ Folder ➔ Assets Folder



```
e edu.cuny.qcc.project1_le
...
 class MainActivity exten
ivate WebView project1;
verride
otected void onCreate(Bun
   super.onCreate(savedIns
   setContentView(R.layout
   project1 = (WebView)fin
   WebSettings settings =
   settings.setJavaScriptE

   FloatingActionButton fa
   fab.setOnClickListener(
          Snackbar.make(v
                  .setAct
   });

verride
blic boolean onCreateOpti
   // Inflate the menu; th
   getMenuInflater().infla
   return true;
```

```
Build: completed successfully   at 1/16/2019 10:39 A
  Run build  C:\Users\hwu\Desktop\project1Cord
    Load build
    Configure build
    Calculate task graph
    Run tasks
```

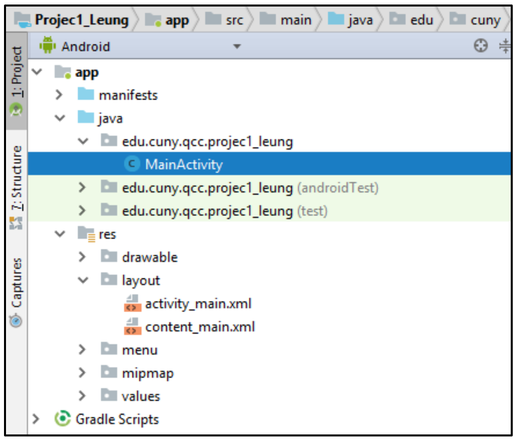
Step 9) Once the *Assets* folder is created, you can located the assets folder in your android folder, and copy and paste the entire web project into the *assets* folder



Name	Date modified	Type	Size
css	1/16/2019 11:31 AM	File folder	
img	1/16/2019 11:31 AM	File folder	
index.html	1/16/2019 10:22 AM	HTML File	3 KB

Step 10) Once the WebView is set, now we need to write a java code that we take Assets folder files and display in an Android App.

Click on arrow next to the java folder, then click on the arrow next to the Company and Activity name domain folder, and open *MainActivity* java file

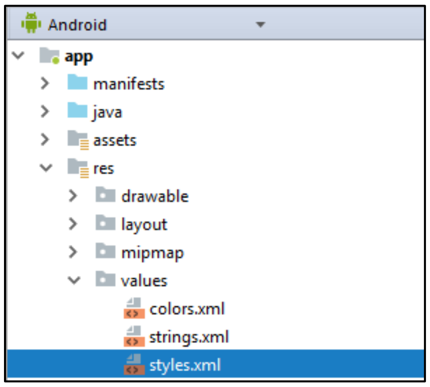


Step 11) In the *MainActivity.java* file, delete all the code from line 6 ➔ `public class` and add the following code:

```
import android.webkit.WebChromeClient;
import android.webkit.WebView;

public class MainActivity extends AppCompatActivity {
    private WebView project1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        project1 = (WebView)findViewById(R.id.webView);
        project1.getSettings().setJavaScriptEnabled(true);
        project1.loadUrl("file:///android_asset/index.html");

        // Interacting with browser
        project1.setWebChromeClient(new WebChromeClient());

    }
}
```

Remember that loadUrl constructor is used to load the HTML file, for this, we will need to add the name of the loading HTML: `project1.loadUrl("file:///android_asset/index.html");`

Step 12) In the *styles.xml* we can modify the Action Bar from the app view.

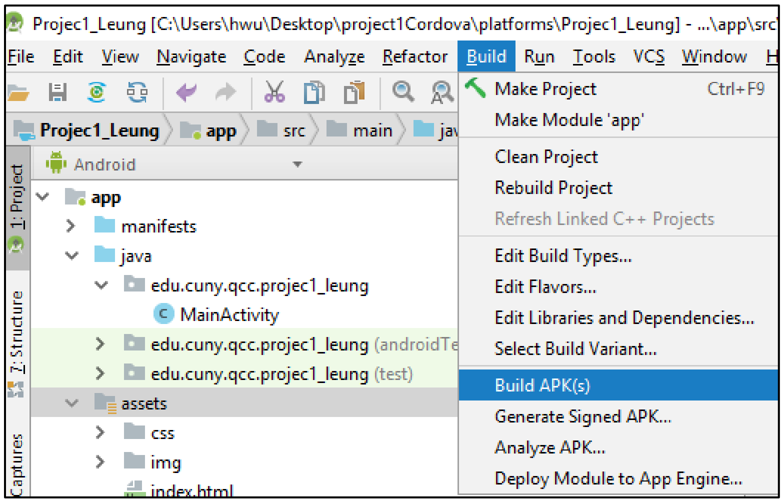


From the *styles.xml* file, we can modify the **`parent`** attribute from the **`<style>`** tag:

```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
</resources>
```

Step 13) To run the emulator and test that the web app is working properly, we can click on the *Run* icon ▶

Also, if you want to pack and build your web app into an APK file[1], go to the menu bar, click on *Build* and select *Build APK(s)*


Projec1_Leung [C:\Users\hwu\Desktop\project1Cordova\platforms\Projec1_Leung] - ...\app\src
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window H
Projec1_Leung app src main jav
Android
app
manifests
java
edu.cuny.qcc.projec1_leung
MainActivity
edu.cuny.qcc.projec1_leung (androidTe
edu.cuny.qcc.projec1_leung (test)
assets
css
img
index.html
1: Project
7: Structure
Captures
Make Project Ctrl+F9
Make Module 'app'
Clean Project
Rebuild Project
Refresh Linked C++ Projects
Edit Build Types...
Edit Flavors...
Edit Libraries and Dependencies...
Select Build Variant...
Build APK(s)
Generate Signed APK...
Analyze APK...
Deploy Module to App Engine...

For more information on how to convert a web app to an android app, you can follow this video on youtube ➔ Converting a HTML+CSS+JS Code into an Android App using Android Studio!

https://www.youtube.com/watch?v=CjiSaGWvsEU

[1] **Android** Package (**APK**) is the package file format used by the **Android** operating system for distribution and installation of mobile apps and middleware.