Before connecting the device download the device driver from the device website in our case:

[**https://developer.samsung.com/mobile/android-usb-driver.html**](https://developer.samsung.com/mobile/android-usb-driver.html)

**to check the connection with the real device we ll use ADB(Android Debug Bridge)**

to use it:

type sdk in settings🡪copy the path of sdk-->go to terminal🡪type **(path copied)/platform-tools/adb.exe devices**

**note: change the braces in the path**

|  |
| --- |
| **Takeaways….**   * **Ctrl+d** to duplicate a line * Don’t give **hardcoded** values ,instead keep it in a **variable** defined in **res/values** folder**.** |

**Android manifest.xml**

This defines overall structure of the android app.

**Exploring AndroidManifest.xml**

🡪Here the **application label** is the name of the application that we see below the application icon.

🡪the activity tag defines an activity .

Here we define the main activity.

**Exploring mainActivity.java**

Here we have **onCreate()** which is called on the creation of the app.

**setContentView** is the graphical representation of the activity.

We have graphical representation of the activity and also a xml(**activity\_main.xml**) file that renders the graphical representation of the activity.

**Drawables**

General **concept of graphics** that android draws.

**Bitmap**

it is a type of **drawable** that provides

* **Png**
* **Jpg**
* **Gif**

**Screen density**

The **number of pixels that share a physical area** of the screen.

Therefore we **should not declare the size of image in pixels**.

Then what to use??

We go for **dp** that is **Density independent Pixels.**

**Launcher Icon**

It should be placed in **mipmap** folder rather than **drawable** folder.

Navigate to res🡪right click on res🡪new🡪image assert🡪assert type (Launcher icon)

**Using the icon**

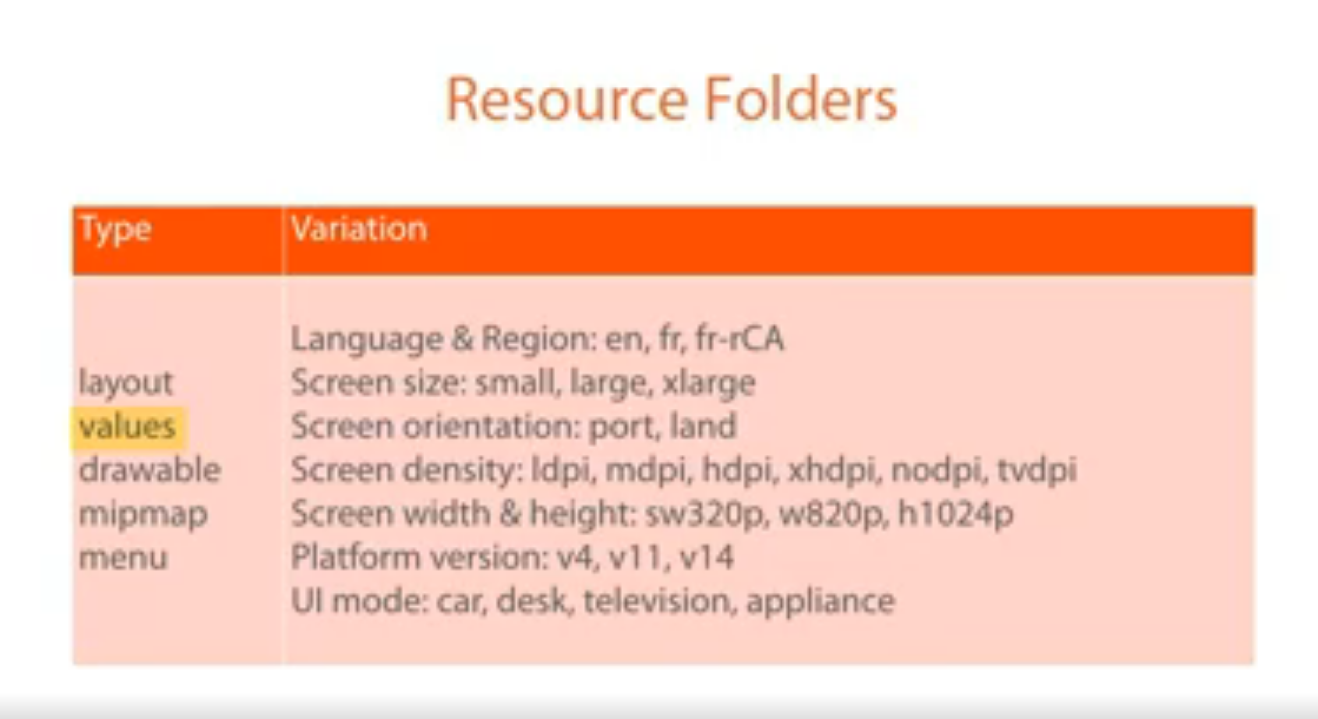
In **AndroidManifest.xml** :

In **application tag**:

Give android:icon=**”@mipmap/ic\_launcher”**

**Ref video**: https://app.pluralsight.com/course-player?clipId=67ff4048-1f4c-4a0f-b25f-433a7f212a39

**RESOURCE FOLDER(res)**

****

**To change the spec according to the screen size.**

Go to app🡪right click on it🡪create a new resource folder🡪ensure it is on values🡪choose screen width/height.

On doing this a new **value** folder will be created for the given screen size.

So how do android knows which value folder has to be loaded in which case??

While building android takes the device configuration and the basis of that it decides the value folder to be loaded.

**To change the spec according to the android version**

Go to app🡪right click on it🡪create a new resource folder🡪ensure it is on values🡪version(choose the quantifiers as version)🡪and give the value as 21 which means on android OS of versions 21 and above will be applicatbe for the style and values under this folder .

**Quantifiers:** It is the factor by which you want to classify the devices and different styles and design has to be made for these classifications.

Few style instances:

Global android:theme

v-14 android:Theme.Holo

v-21 android:theme.Material

**what to do if we want to change the text to blue color??**

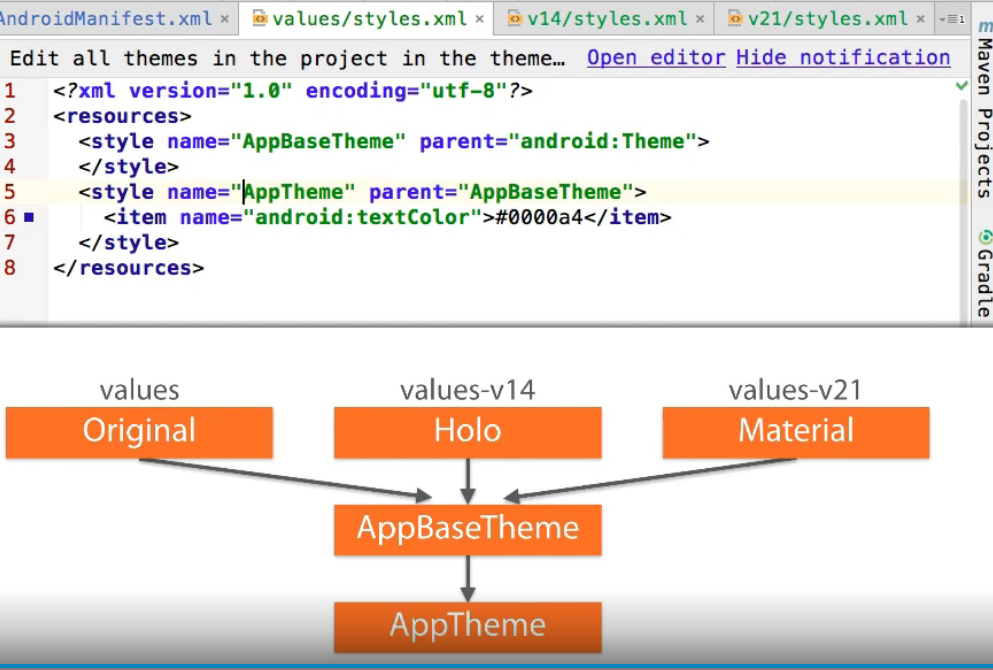
The common way is to go and change in the **style.xml** of the default **value** folder.

But since we have two more value folder (one for v14 above and for v21 above). So there the font color will not be changed.

Therefore in order to make the change in all the version we have to keep a style in default value folder.(say ”AppBaseTheme” in the below picture).

This style should have the styling code which should vary with the version.

And we should have another style that inherits the “AppBaseTheme” , this consists of the styling code that is common and should be applied irrespective of the versions.



# Android Hello World  
  
In Android the Hello World example is generated by the IDE, bundled with state-of-the-art best  
practices. Over the years it has become more and more complex, and getting rather overwhelming for  
first-timers.  
  
This repository starts from the bare bones and add one concept at a time for a gentler introduction  
to the Android environment.  
  
Start from the `master` branch, build up the app by following these branches:  
  
 \* `01-drawables`  
 \* `02-dimens`  
 \* `03-styles`  
 \* `04-menu`  
 \* `05-test`  
  
# Pluralsight Course  
  
You can also watch this Pluralsight course for a guided experience:   
https://www.pluralsight.com/courses/android-start-developing  
  
# Next Steps  
  
Explore these topics to deepen your Android knowledge.  
  
### Navigation  
  
 \* [Intent Fundamentals](https://www.pluralsight.com/courses/android-fundamentals-intents)  
  
### UI  
  
 \* [Layout Fundamentals](http://pluralsight.com/courses/android-layout-fundamentals)  
 \* [Custom Components](http://pluralsight.com/courses/android-custom-components)  
  
### Compatibility  
  
 \* [Support Multiple Device Definitions](http://developer.android.com/training/basics/supporting-devices/index.html)  
  
### Background  
  
 \* [Internet and Background Threads](https://guides.codepath.com/android/Sending-and-Managing-Network-Requests)  
  
### Persistence  
  
 \* [SharedPreferences](http://developer.android.com/training/basics/data-storage/shared-preferences.html)  
  
### Testing  
  
 \* [Testing Fundamentals](http://developer.android.com/training/testing.html)  
 \* [Unit Testing Fundamentals](http://developer.android.com/training/testing/unit-testing/index.html)  
 \* [Espresso Fundamentals](https://google.github.io/android-testing-support-library/docs/espresso/index.html)