Android Kotlin Fundamentals Notes 01.3: Image resources and compatibility

App resources:

* Your app's resources can include images and icons, standard colors used in the app, strings, and XML layouts. All of those resources are stored in the res folder.
* The drawable resources folder is where you should put all the image resources for your app.

Using vector drawables in image views:

* Vector drawables are images described in XML format. Vector drawables are more flexible than bitmap images (such as PNG files) because they can be scaled to any size or resolution.
* To add a drawable to your app's layout, use an <ImageView> element. The source of the image is in the android:src attribute. To refer to the drawable resource folder, use @drawable, for example, "@drawable/image\_name".
* Use the ImageView view in your MainActivity code for the image. You can use setImageResource() to change the view's image to a different resource. Use R.drawable to refer to specific drawables, for example setImageResource(R.drawable.image\_name).

The lateinit keyword:

* Minimize the calls to findViewById() in your code by declaring fields to hold those views, and initializing the fields in onCreate(). Use the lateinit keyword for the field to avoid needing to declare it nullable.

The tools namespace for design-time attributes:

* Use the tools:src attribute in the <ImageView> element in your layout to display an image in only Android Studio's preview or design editor. You can then use an empty image for android:src for the final app.
* Use the tools namespace in the Android layout file to create placeholder content or hints for layout in Android Studio. Data declared by tools attributes is not used in the final app.

API levels:

* Each Android OS has an official version number and name (for example Android 9.0, "Pie") and an API level (API 28). Use the API levels in your app's Gradle files to indicate the versions of Android your app supports.
* The compileSdkVersion parameter in the build.gradle file specifies the Android API level that Gradle should use to compile your app.
* The targetSdkVersion parameter specifies the most recent API level that you have tested your app against. In many cases this parameter has the same value as compileSdkVersion.
* The minSdkVersion parameter specifies the oldest API level your app can run on.

Android Jetpack:

* Android Jetpack is a collection of libraries, developed by Google, that offers backward-compatible classes and helpful functions for supporting older versions of Android. Jetpack replaces and expands on the set of libraries formerly known as the Android Support Library.
* Classes imported from the androidx package refer to the Jetpack libraries. Dependencies to Jetpack in your build.gradle file also start with androidx.

Backward compatibility for vector drawables:

* Vector drawables are only natively supported in versions of Android higher than API 21. In older versions, Gradle generates PNG images for those drawables when your app is built.
* You can specify that the Android Support Library should be used for vector drawables in older API versions with the vectorDrawables.useSupportLibrary = true configuration parameter in the build.gradle file.
* Once you've enabled the support library for vector drawables, use the app:srcCompat attribute in the <ImageView> element (instead of android:src) to specify the vector drawable source for that image.

The app namespace:

* The app namespace in your XML layout file is for attributes that come from either your custom code or from libraries, not from the core Android framework.