DataBinding Definition:

- The Data Binding Library is an Android Jetpack library that allows you to bind UI components in your XML layouts to data sources in your app using a declarative format rather than programmatically, reducing boilerplate code.
- This means the relationship between your views and source code is created via the binding class.

DataBinding Setup:

- You need to configure/change your gradle.build file to include the following code lines

```
android {
    // Previously there
    // Add this next line
    dataBinding.enabled = true // <----
    // ...
```

How Binding works

- Android studio creates a binding class for every layout holds the references of view in your layout.

- Note that this binding object is generated automatically with the following rules in place:
- Layout file activity_main.xml becomes ActivityMainBinding binding object
- View IDs within a layout become variables inside the binding object (i.e binding.tvLabel)

Binding code generated by Android Studio

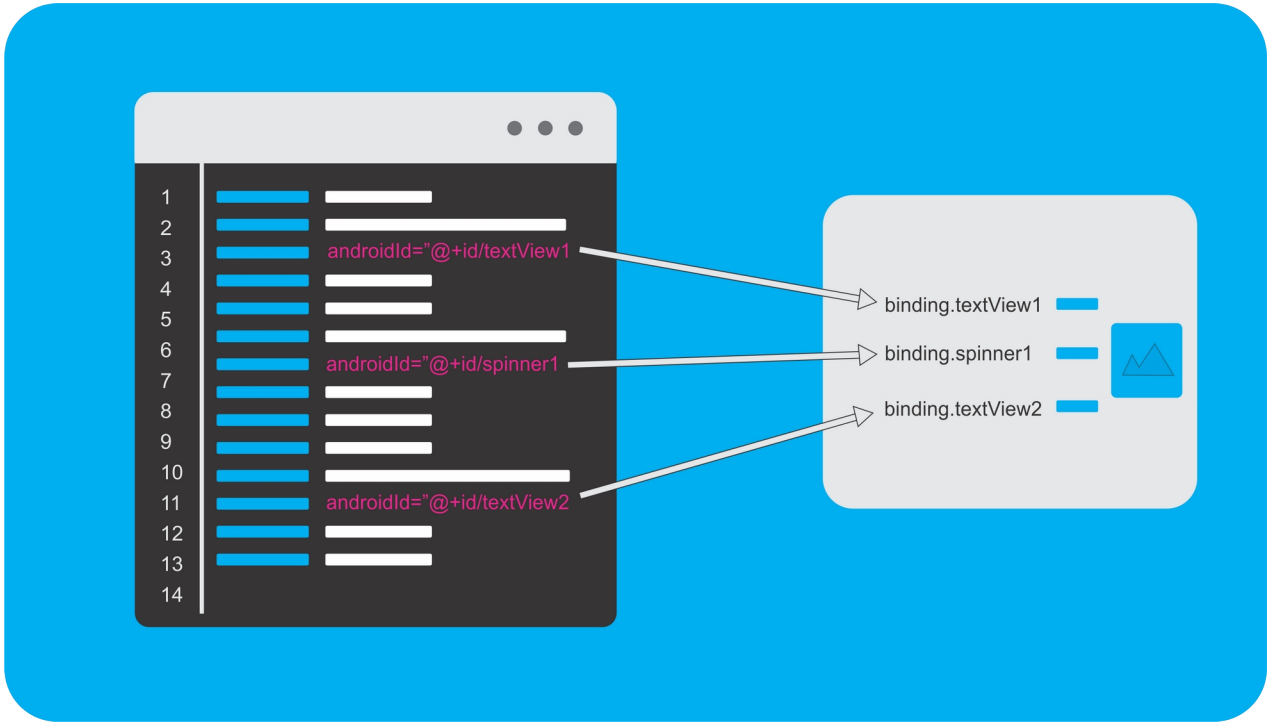
- Android Studio auto generates this code and binds your layout Class to your variable using this statement

```
private lateinit var binding: ActivityMainBinding
```

```
[ private lateinit var binding: ActivityMainBinding ]
override fun onCreate(sacedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflatte(layoutInflater)
        setContentView(binding.root)
```

ViewBinding and DataBinding

How ViewBinding and Databinding creates the your layout view references



DataBinding Setup:

- Then you need to use the layout tag as the out most tag. Then move the xmlns:android= properties to the layout tag

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
    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">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/Theme.NoteKeeper.AppBarOverlay">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/Theme.NoteKeeper.PopupOverlay" />

    </com.google.android.material.appbar.AppBarLayout>

    <include
        android:id="@+id/content_on_main"
        layout="@layout/content_main" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<layout
    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">
    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">

        <TextView
            android:id="@+id/addedNumbers"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text=""
            android:textSize="20sp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintHorizontal_bias="0.119"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_bias="0.422" />

    </androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

DataBinding Features:

- Data binding gives the ability to bind data with XML layouts. This gives access to binding expressions, BindingAdapters and two way binding.

```xml
<layout 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">

    <data>

        <variable
            name="user"
            type="com.sample.myapplication.User" />
    </data>

    <androidx.constraintlayout.widget.ConstraintLayout/>
    <!-- UI layout's root element -->
</layout>
```

DataBinding Features:

- <data>: A container for variables used inside the layout file. You will be using the variables defined inside to work with various attributes in your layout like for setting the data in your views or if you want some import statement for some expression:

```
<layout 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">

    <data>

        <variable
            name="user"
            type="com.sample.myapplication.User" />
    </data>

    <androidx.constraintlayout.widget.ConstraintLayout/>
    <!-- UI layout's root element -->
</layout>
```

One Way Binding:

- If we create a normal object class that just has variables and getter and setter.

```
public class User {
   public String firstName;
   public String lastName;
}
```

- We need to indicate that we want to load data from particular object by declaring <variable> node in a <data> section of the <layout>

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <data>
       <variable name="user" type="com.example.User"/>
    </data>
    <!-- ... rest of layout here -->
</layout>
```

One Way Binding:

- If we create a normal object class that just has variables and getter and setter.

```
public class User {
   public String firstName;
   public String lastName;
}
```

- We need to indicate that we want to load data from particular object by declaring <variable> node in a <data> section of the <layout>

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <data>
       <variable name="user" type="com.example.User"/>
    </data>
    <!-- ... rest of layout here -->
</layout>
```

One Way Binding:

- The user variable inside the data tag is a variable that can be used inside the layout. This variable can be reference by using the following syntax inside the layout @{variable.field}

```
<TextView
    android:id="@+id/tvFullName"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text='@{user.firstName + " " + user.lastName}' />
```

- We can use conditional logic and other operations as part of the binding expression language for more complex cases.

DataBinding References

- Android Data Binding: Under the Hood (Part 1)

  https://proandroiddev.com/android-data-binding-under-the-hood-part-1-33b8c7adfb7c

- Applying Data Binding for Views

  https://guides.codepath.com/android/Applying-Data-Binding-for-Views

- Use view binding to replace findViewById

  https://medium.com/androiddevelopers/use-view-binding-to-replace-findviewbyid-c83942471fc

More information on android Binding Class

- https://developer.android.com/topic/libraries/data-binding/generated-binding
-
-