* **Designing User Interfaces with Layouts**

A layout defines the structure for a user interface in your app, such as in an [activity](https://developer.android.com/guide/components/activities). All elements in the layout are built using a hierarchy of [View](https://developer.android.com/reference/android/view/View) and [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup) objects. A View usually draws something the user can see and interact with. A ViewGroup is an invisible container that defines the layout structure for View and other ViewGroup objects



Android Layout Types

There are number of Layouts provided by Android which you will use in almost all the Android applications to provide different view, look and feel.

|  |  |
| --- | --- |
| **Sr.No** | **Layout & Description** |
| 1 | [**Linear Layout**](https://www.tutorialspoint.com/android/android_linear_layout.htm)  LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally. |
| 2 | [**Relative Layout**](https://www.tutorialspoint.com/android/android_relative_layout.htm)  RelativeLayout is a view group that displays child views in relative positions. |
| 3 | [**Table Layout**](https://www.tutorialspoint.com/android/android_table_layout.htm)  TableLayout is a view that groups views into rows and columns. |
| 4 | [**Absolute Layout**](https://www.tutorialspoint.com/android/android_absolute_layout.htm)  AbsoluteLayout enables you to specify the exact location of its children. |
| 5 | [**Frame Layout**](https://www.tutorialspoint.com/android/android_frame_layout.htm)  The FrameLayout is a placeholder on screen that you can use to display a single view. |
| 6 | [**List View**](https://www.tutorialspoint.com/android/android_list_view.htm)  ListView is a view group that displays a list of scrollable items. |
| 7 | [**Grid View**](https://www.tutorialspoint.com/android/android_grid_view.htm)  GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid. |

Layout Attributes

Each layout has a set of attributes which define the visual properties of that layout. There are few common attributes among all the layouts and their are other attributes which are specific to that layout. Following are common attributes and will be applied to all the layouts:

|  |  |
| --- | --- |
| **Sr.No** | **Attribute & Description** |
| 1 | **android:id**  This is the ID which uniquely identifies the view. |
| 2 | **android:layout\_width**  This is the width of the layout. |
| 3 | **android:layout\_height**  This is the height of the layout |
| 4 | **android:layout\_marginTop**  This is the extra space on the top side of the layout. |
| 5 | **android:layout\_marginBottom**  This is the extra space on the bottom side of the layout. |
| 6 | **android:layout\_marginLeft**  This is the extra space on the left side of the layout. |
| 7 | **android:layout\_marginRight**  This is the extra space on the right side of the layout. |
| 8 | **android:layout\_gravity**  This specifies how child Views are positioned. |
| 9 | **android:layout\_weight**  This specifies how much of the extra space in the layout should be allocated to the View. |
| 10 | **android:layout\_x**  This specifies the x-coordinate of the layout. |
| 11 | **android:layout\_y**  This specifies the y-coordinate of the layout. |
| 12 | **android:layout\_width**  This is the width of the layout. |
| 13 | **android:paddingLeft**  This is the left padding filled for the layout. |
| 14 | **android:paddingRight**  This is the right padding filled for the layout. |
| 15 | **android:paddingTop**  This is the top padding filled for the layout. |
| 16 | **android:paddingBottom**  This is the bottom padding filled for the layout. |

Here width and height are the dimension of the layout/view which can be specified in terms of dp (Density-independent Pixels), sp ( Scale-independent Pixels), pt ( Points which is 1/72 of an inch), px( Pixels), mm ( Millimeters) and finally in (inches).

You can specify width and height with exact measurements but more often, you will use one of these constants to set the width or height −

* **android:layout\_width=wrap\_content** tells your view to size itself to the dimensions required by its content.
* **android:layout\_width=fill\_parent** tells your view to become as big as its parent view.

View Identification

A view object may have a unique ID assigned to it which will identify the View uniquely within the tree. The syntax for an ID, inside an XML tag is −

android:id="@+id/my\_button"

Following is a brief description of @ and + signs −

* The at-symbol (@) at the beginning of the string indicates that the XML parser should parse and expand the rest of the ID string and identify it as an ID resource.
* The plus-symbol (+) means that this is a new resource name that must be created and added to our resources.

**MainActivity.java**

package com.example.button;  
  
import android.os.Bundle;  
import android.widget.Button;  
import android.widget.Toast;  
import androidx.appcompat.app.AppCompatActivity;  
  
public class MainActivity extends AppCompatActivity {  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 Button button = findViewById(R.id.*button*);  
 button.setOnClickListener(v ->  
 Toast.*makeText*(this, "Button Clicked!", Toast.*LENGTH\_SHORT*).show()  
 );  
 }  
}

**activity\_main.xml**

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:padding="40dp">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Welcome to LinearLayout"  
 android:textSize="18sp"  
 android:textStyle="bold" />  
  
 <Button  
 android:id="@+id/button"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Click Me"  
 android:layout\_marginTop="10dp"/>  
  
</LinearLayout>

