**2. Android UI Layouts**

**(Linear, Relative, Frame, Table, ListView, GridView, WebView)**

In android, **Layout** is used to define the user interface for an app or [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle) and it will hold the UI elements that will appear to the user.

The user interface in android app is made with a collection of View and ViewGroup objects. Generally, the android apps will contain one or more activities and each activity is a one screen of app. The activities will contain a multiple UI components and those UI components are the instances of View and ViewGroup subclasses.

The View is a base class for all UI components in android and it is used to create an interactive UI components such as [TextView](https://www.tutlane.com/tutorial/android/android-textview-with-examples" \o "Android TextView with Examples" \t "_blank), [EditText](https://www.tutlane.com/tutorial/android/android-edittext-with-examples), [Checkbox](https://www.tutlane.com/tutorial/android/android-checkbox-with-examples), [Radio Button](https://www.tutlane.com/tutorial/android/android-radiobutton-with-examples), etc. and it responsible for event handling and drawing.

The ViewGroup is a subclass of View and it acts as a base class for **layouts** and **layouts parameters**. The ViewGroupwill provide an invisible containers to hold other Views or ViewGroups and to define the layout properties.

In android, we can define a layouts in two ways, those are

* Declare UI elements in XML
* Instantiate layout elements at runtime

The android framework allows us to use either or both of these methods to define our application’s UI.

**Declare UI Elements in XML**

In android, we can create a layouts same like web pages in HTML by using default [Views and ViewGroups](https://www.tutlane.com/tutorial/android/android-view-and-viewgroup-with-examples) in XML file. The layout file must contain only one root element, which must be a View or ViewGroup object. Once we define root element, then we can add additional layout objects or widgets as a child elements to build View hierarchy that defines our layout.

Following is the example of defining a layout in XML file (**activity\_main.xml**) using [LinearLayout](https://www.tutlane.com/tutorial/android/android-linearlayout-with-examples" \o "Android Linear Layout with Examples" \t "_blank) to hold a [TextView](https://www.tutlane.com/tutorial/android/android-textview-with-examples" \o "Android TextView with Examples" \t "_blank), [EditText](https://www.tutlane.com/tutorial/android/android-edittext-with-examples" \o "Android EditText with Examples" \t "_blank) and [Button](https://www.tutlane.com/tutorial/android/android-button-with-examples).

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"

    android:layout\_width="match\_parent"  
    android:layout\_height="match\_parent">  
    <TextView  
        android:id="@+id/fstTxt"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:text="Enter Name"  
         />  
    <EditText  
        android:id="@+id/name"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:ems="10">  
    </EditText>  
    <Button  
        android:id="@+id/getName"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:text="Get Name" />  
</LinearLayout>

We need to create a layout files in **/res/layout** project directory, then only the layout files will compile properly.

**Load XML Layout File from an Activity**

Once we are done with creation of layout, we need to load the XML layout resource from our [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle) **onCreate()** callback method like as shown below

protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity\_main);    
}

If we observe above code we are calling our layout using **setContentView** method in the form of **R.layout.layout\_file\_name**. Here our xml file name is **activity\_main.xml** so we used file name **activity\_main**.

Generally, during the launch of our [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle), **onCreate()** callback method will be called by android framework to get the required layout for an [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle).

**Instantiate Layout Elements at Runtime**

If we want to instantiate layout elements at runtime, we need to create own custom View and ViewGroup objects programmatically with required layouts.

Following is the example of creating a layout using [LinearLayout](https://www.tutlane.com/tutorial/android/android-linearlayout-with-examples) to hold a [TextView](https://www.tutlane.com/tutorial/android/android-textview-with-examples" \o "Android TextView with Examples" \t "_blank), [EditText](https://www.tutlane.com/tutorial/android/android-edittext-with-examples" \o "Android EditText with Examples" \t "_blank) and [Button](https://www.tutlane.com/tutorial/android/android-button-with-examples) in an [activity](https://www.tutlane.com/tutorial/android/android-activity-lifecycle) using custom View and ViewGroup objects programmatically.

public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        TextView textView1 = new TextView(this);  
        textView1.setText("Name:");  
        EditText editText1 = new EditText(this);  
        editText1.setText("Enter Name");  
        Button button1 = new Button(this);  
        button1.setText("Add Name");  
        LinearLayout linearLayout = new LinearLayout(this);  
        linearLayout.addView(textView1);  
        linearLayout.addView(editText1);  
        linearLayout.addView(button1);  
        setContentView(linearLayout);  
    }  
}

By creating a custom View and ViewGroup programmatically, we can define a layouts based on our requirements in android applications.

**Width and Height**

When we define a layout using XML file we need to set width and height for every View and ViewGroup element using **layout\_width** and **layout\_height** attributes.

Following is the example of setting width and height for View and ViewGroup elements in XML layout file.

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"

    android:layout\_width="match\_parent"  
    android:layout\_height="match\_parent">  
    <TextView  
        android:id="@+id/fstTxt"  
        android:layout\_width="wrap\_content"  
        android:layout\_height="wrap\_content"  
        android:text="Enter Name" />  
</LinearLayout>

If we observe above example, we used different values to set **layout\_width** and **layout\_height**, those are

* match\_parent
* wrap\_content

If we set value match\_parent, then the View or ViewGroup will try to match with parent width or height.

If we set value wrap\_content, then the View or ViewGroup will try to adjust its width or height based on the content.

**Android Layout Attributes**

In android, like **layout\_width** and **layout\_height** we have a different type of attributes available for View and ViewGroupobjects to define the appearance of layouts based on our requirements.

| **Attribute** | **Description** |
| --- | --- |
| android:id | It is used to uniquely identify the view and ViewGroups |
| android:layout\_width | It is used to define the width for View and ViewGroup elements in layout |
| android:layout\_height | It is used to define the height for View and ViewGroup elements in layout |
| android:layout\_marginLeft | It is used to define the extra space in left side for View and ViewGroup elements in layout |
| android:layout\_marginRight | It is used to define the extra space in right side for View and ViewGroup elements in layout |
| android:layout\_marginTop | It is used to define the extra space on top for View and ViewGroup elements in layout |
| android:layout\_marginBottom | It is used to define the extra space in bottom side for View and ViewGroup elements in layout |
| android:paddingLeft | It is used to define the left side padding for View and ViewGroup elements in layout files |
| android:paddingRight | It is used to define the right side padding for View and ViewGroup elements in layout files |
| android:paddingTop | It is used to define padding for View and ViewGroup elements in layout files on top side |
| android:paddingBottom | It is used to define the bottom side padding for View and ViewGroup elements in layout files |
| android:layout\_gravity | It is used to define how child Views are positioned |
| **ems** is a typography term,[is the art and technique of arranging type to make written language legible, readable, and appealing when displayed. Point is the smallest unit of measure in typography (controls text size, etc).](https://en.wikipedia.org/wiki/Typography)The name em was originally a reference to the width of the **capital M**. It sets the width of a TextView/EditText to fit a text of n 'M' letters regardless of the actual text extension and text size. | |

Following are the some of common layout attributes used in android application.

**Android Layout Types**

We have a different type of layouts available in android to implement user interface for our android applications with different designs based on our requirements.

Following are the commonly used layouts in android applications to implement required designs.

* [Linear Layout](https://www.tutlane.com/tutorial/android/android-linearlayout-with-examples)
* [Relative Layout](https://www.tutlane.com/tutorial/android/android-relativelayout-with-examples)
* [Frame Layout](https://www.tutlane.com/tutorial/android/android-framelayout-with-examples)
* [Table Layout](https://www.tutlane.com/tutorial/android/android-tablelayout-with-examples)
* [Web View](https://www.tutlane.com/tutorial/android/android-webview-with-examples)
* [List View](https://www.tutlane.com/tutorial/android/android-listview-with-examples)
* [Grid View](https://www.tutlane.com/tutorial/android/android-gridview-with-examples)

**Android Linear Layout**

In android, [LinearLayout](https://www.tutlane.com/tutorial/android/android-linearlayout-with-examples" \o "Android Linear Layout with Examples" \t "_blank) is a ViewGroup subclass which is used to render all child View instances one by one either in horizontal direction or vertical direction based on the orientation property.

**Android Relative Layout**

In android, [RelativeLayout](https://www.tutlane.com/tutorial/android/android-relativelayout-with-examples" \o "Android Relative Layout with Examples" \t "_blank) is a ViewGroup which is used to specify the position of child View instances relative to each other (Child A to the left of Child B) or relative to the parent (Aligned to the top of parent).

**Android Frame Layout**

In android, [FrameLayout](https://www.tutlane.com/tutorial/android/android-framelayout-with-examples" \o "Android Frame Layout with Examples" \t "_blank) is a ViewGroup subclass which is used to specify the position of View instances it contains on the top of each other to display only single View inside the FrameLayout.

**Android Table Layout**

In android, [TableLayout](https://www.tutlane.com/tutorial/android/android-tablelayout-with-examples" \o "Android Table Layout with Examples" \t "_blank) is a ViewGroup subclass which is used to display the child View elements in rows and columns.

**Android Web View**

In android, [WebView](https://www.tutlane.com/tutorial/android/android-webview-with-examples" \o "Android WebView with Examples" \t "_blank) is a browser which is used to display the web pages as a part of our activity layout.

**Android List View**

In android, [ListView](https://www.tutlane.com/tutorial/android/android-listview-with-examples" \o "Android List View with Examples" \t "_blank) is a ViewGroup which is used to display scrollable single column list of items.



**Android Grid View**

In android, [GridView](https://www.tutlane.com/tutorial/android/android-gridview-with-examples" \o "Android Gridview with Examples" \t "_blank) is a ViewGroup which is used to display items in a scrollable grid of columns and rows.

**Questions**

1. What are the different types of android layouts – explain their features and activities.