Android User Interface UI

An Activity interacts with the user, via a visual User Interface(***UI***) on a screen. To display a UI, you assign a View (usually a l**ayout** or **Fragment**) to an Activity.

View is a basic building block of UI In android , **view** is a small rectangular box **which** responds to user inputs. Eg: EditText , Button , CheckBox ,

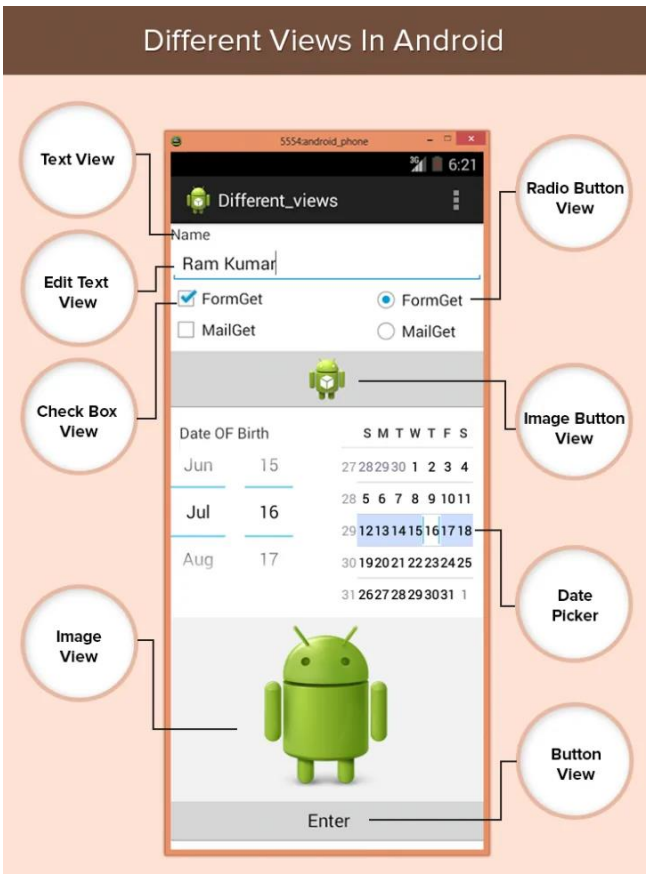
View is the base class for ***widgets***, which are used to create interactive UI components *like buttons, text fields, etc*

What is a view in android? Give an example.

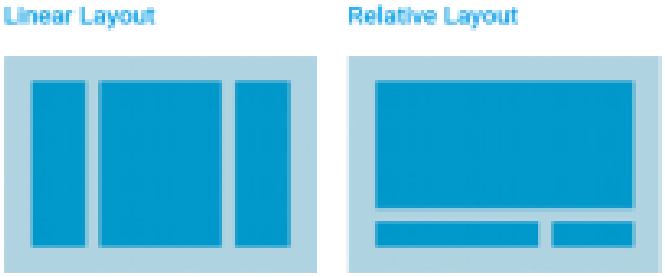
A View is an interactive UI component (***widget*** or ***control)***, such as *button* and *text field*. It controls a *rectangular area* on the screen.

Examples of view :

- TextView,
- EditText,
- Button,
- RadioButton, etc,



A ViewGroup is an *invisible container* used to *layout* the View components. Android provides many ready-to-use ViewGroups such



- as Linear Layout,
- Relative Layout,
- Table Layout
- and Grid Layout
- Each of these layouts is designed to scale to suit the host device's screen size by avoiding the use of absolute positions or predetermined pixel values.

This makes them particularly useful when designing applications that work well on a diverse set of Android hardware

A `ViewGroup` is a special view that can contain other views (called children.) The view group is the base class for layouts and views containers. This class also defines the `ViewGroup.LayoutParams` class which serves as the base class for layouts parameters.

Views and ViewGroups are organized in a single tree structure called ***view-tree***. You can create a view-tree either using programming codes or describing it in a **XML layout file**.

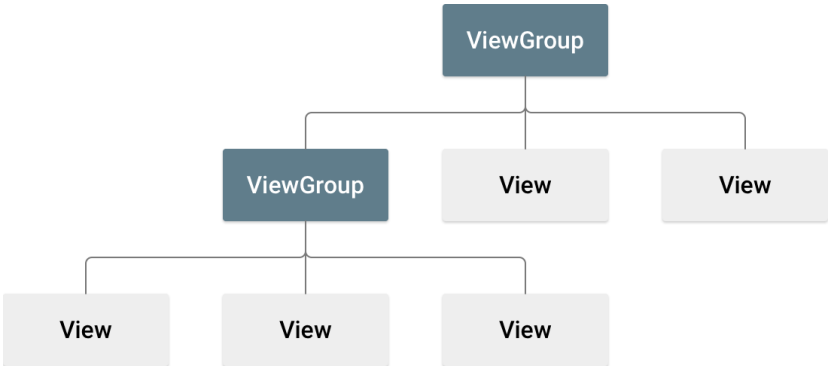


Figure 1 Tree structure for the view

XML file separates the presentation view from the controlling logic, thus it provides modularity and flexibility in your program design thus it is recommended to use XML layout file..

There are two approaches in constructing the UI:

1. **Via the XML Layout file:** For example, in "Hello-world in XML Layout", we layout all the UI components in an XML file. This approach is more flexible, with the view separated from the business logic, and therefore recommended.
2. **Via the programming codes,** as in the above Hello-world example. Use this approach only if absolutely necessary.

A ***Layout*** is a subclass of ViewGroup class. layout defines the visual structure for an Android user interface and can be created either at run time using ***View/ViewGroup objects*** or you can declare your layout using simple XML file ***main_layout.xml*** which is located in the ***res/layout folder*** of your project.

Activity and UI

A new Activity starts with an empty screen onto which you place your UI. To do so, call *setContentView*, passing in the View instance, or layout resource, to display.

Because empty screens aren't particularly, you will almost always use *setContentView* to assign an Activity's UI when overriding its *onCreate* handler.

The setContentView method accepts either a ***layout's*** resource ID or a ***single View*** instance. This lets you define your UI either in code or using the preferred technique of external layout resources.

Layout with set SetContentView example	View with SetContentView example

SetContentView(R.layout.main)
R__> means Resource
layout __>means design
main __>is the xml you have created under res->layout->main.xml
	@Override
	Protected void onCreate(Bundle savedInstanceState)
	{
	//create a textview object
	TextView textview=new TextView(this);
	//set text to textview
	Textview.setText("hello world!");
	SetContentView(textview);
	}

Whenever you want to change the current look of an Activity or when you move from one Activity to another, the new Activity must have a design to show. We call `setContentView` in onCreate with the desired design as argument.

Intents

There are two kind of intents in android :

- Explicit Intent : - An explicit intents are commonly used to launch an another activity (from current activity) within the same application.
  For example, you might start a new activity within your app in response to a user action, or start a service to download a file in the background

- Implicit Intent :- An implicit intents are commonly used to perform some action and optionally some data required for that action.
  For example, if you want to show the user a location on a map, you can use an implicit intent to request that another capable app show a specified location on a map.

In android, Intents are the objects of **android.content.Intent** types and intents are mainly useful to perform the following things.

Component	Description
Starting an Activity	By sending an **Intent** object to `startActivity()` method we can start a new Activity or existing Activity to perform required things.
Starting a Service	By sending an **Intent** object to `startService()` method we can start a new Service or send required instructions to an existing Service.
Delivering a Broadcast	By sending an **Intent** object to `sendBroadcast()` method we can deliver our messages to other app broadcast receivers.

What is intent filter, why we use it?

If we want to make our activity available to be used by other apps(in other word we want implicit type of intent to be possible for this activity), then we need to declare an intent filter in manifest file.

When you use an implicit intent, the Android system finds the appropriate component to start by comparing the **contents of the intent** to **the *intent filters*** declared in the manifest file of other apps on the device.

If the intent matches an intent filter, the system starts that component and delivers it the Intent object. If multiple intent filters are compatible, the system displays a dialog so the user can pick which app to use.

An intent filter is an expression in an app's **manifest file** that specifies the type of intents that the component would like to receive.

For instance, by declaring an intent filter for an activity, you make it possible for other apps to directly start your activity with a certain kind of intent. Likewise, if you do *not* declare any intent filters for an activity, then it can be started only with an explicit intent.

if you do *not* declare any intent filters for an activity, then it can be started only with an explicit intent. Which mean we can use it insid the apps not between two or more apps.

Using Intents to Launch Activities

Intents are used to start Activities, allowing you to create a workflow of different screens.

To create and display an Activity, call startActivity, passing in an Intent, as follows:

```
startActivity(myIntent);
```

The startActivity method finds and starts the single Activity that best matches your Intent.